Forecasting daily month-ahead TTF gas prices using a combination of preprocessing and machine learning techniques.

by

T.P.N.M. Vertregt

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 16th, 2023 at 11:00 AM.

Student number:	4589467	
Project duration:	February 1, 2023 – August 16, 2023	
Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft
	Mr. ir. drs. V. N. S. R. Dwarka,	TU Delft
	Dr. ir. G. N. J. C. Bierkens,	TU Delft
	Dr. I. Borovykh,	Northpool

This thesis is confidential and cannot be made public until August 16, 2025.

TU Delft

Abstract

Although energy commodity price forecasting has been around for quite some time, up until recently, especially in Europe, it mostly concerned other energy commodities than gas. That is why in this work, a forecasting model is presented for the single day forecast of the daily VWAP price of the TTF month-ahead gas contract from 2020/01/02 until 2023/08/03. A model combining machine learning and data preprocessing is proposed. First a decomposition of the gas price is produced by a combination of Variational Mode Decomposition (VMD) and Independent Component Analysis (ICA). With these decompositions an initial volatility regime split is chosen because of the unusual characteristics of the dataset. During the forecasting period the Russo-Ukrainian war started, and the economy was recovering from the COVID-19 crisis, causing the gas price to surge to levels previously unknown. Using the decompositions as input, a final price prediction has been made using a Gated Recurrent Unit Neural Network (GRUNN) and Support Vector Regression (SVR), amongst others. The proposed model is compared to a selection of benchmarks, one of which is the naive forecast. To conclude, a selection of exogenous variables is added to the model to improve the performance. Gas storage and the UK NBP gas price are chosen for their specific characteristics. The best performance the model exhibits is between 0.38% MAPE for the low volatility regime, and 1.4% MAPE for the high volatility regime.

Acknowledgements

I would like to express my gratitude towards Northpool, for creating the opportunity to write my thesis fulltime. In particular, Igor, Maria, and the entire algorithmic trading team. I could always run to you when I had a question, and you would help me instantly, which is something I deeply appreciate. Besides this, I learned a lot about everything trading related, and had a very fun time doing it! Next, I would like to thank Vandana and Kees for guiding me through the thesis process. Especially Vandana, thank you for taking me as your thesis student, even though I already started the project. I really appreciate you helping me getting my thesis back on the rails initially, after a turbulent week. I really enjoyed the process of writing the thesis, and I think we can be proud of the work we've produced.

Contents

1 Introduction

Although energy commodity price forecasting has been around for quite some time, up until recently, especially in Europe, it mostly concerned other energy commodities than gas. This has to do with the fact that, for a long time, Europe relied heavily on gas supplied by large pipelines from countries such as Russia, Norway and Holland (Hamie et al. (2020)). The prices of these supplies were often set in contracts with long durations, which were linked to the oil price, causing little interest in forecasting gas prices. Only since the 2000s gas hubs[1] started to emerge in Europe, which opened up the European gas markets to the import of Liquefied Natural Gas (LNG) from all over the world. By 2013, most of the long term pipeline contracts where shifted to hub indexation. The introduction of exchange traded LNG in Europe caused gas prices to decouple from the oil price, subsequently meriting research into forecasting the gas price independently from the oil price.

Many hubs in continental Europe exist, most of which are virtual hubs (Shi (2016)), as they offer more flexible trade arrangements and are more open to participants, in particular to financial players. The first gas hub on European soil was built in 1998 in Zeebrugge, Belgium, connecting mainland Europe to the National Balancing Point (NBP), the main gas hub in the UK. Every European country has their own gas hub. To name a few: Title Transfer Facility (TTF, NL); Trading Region France (TRF, FR); Trading Hub Europe (THE, DE); Punto di Scambio Virtuale (PSV, ITA). Some of these trading hubs have opened very recently, such as TRF (2018) and THE (2021). This research is concerned with the TTF gas price traded in Rotterdam, the Netherlands. The TTF is consistently ranked as the best trading hub in continental Europe[2]. TTF has the most market participants, the widest range of products, with the highest volumes in every category, and is now widely regarded as a global price reference (Patrick Heather (2020)).

Futures

The most common way to trade gas on a gashub is through a futures contract. A futures contract is a standardised legal agreement between two parties to buy or sell gas for a predetermined price during a specific period in the future. Some common contracts traded on the TTF are the Day-Ahead (DA) and Month-Ahead contracts (MA). The day-ahead contract obliges the selling party to deliver the gas during the following day, while for the month-ahead contract the gas has to be delivered during the subsequent month. More details about the monthly contracts traded on the TTF are given in section 3. Usually, the day-ahead contracts are used to meet short-term demand. The month-ahead contracts are used more often for speculation by financial parties. Many other contracts exist besides the DA and MA, such as week-ahead, 2-,3-,4-,... month-ahead contracts. If we are talking about a transaction of the April-2023 contract for example, then it is meant that the gas will be delivered during the course of April 2023, for the specified price. Of course, if a party buys the contract before the first of April 2023, then it becomes responsible for the acquisition of the gas as well. As we will see in section 3, only contracts with a specific delivery day or month are traded on the exchanges. This means that we need to make some adjustments to obtain a single, continuous price series for the month-ahead contract. When talking about trading futures, the term "spot price" informally refers to all contract with a delivery date in the near future (typically 2 days from now).

1.1 Context

The research described in this thesis has been made possible by Northpool. Northpool is a proprietary trading firm active on the energy commodity markets. Founded in 2013 by Roald van Noort, Northpool mainly trades power, gas and carbon contracts. With the use of data, robust weather models, predictive analyses and fundamental market knowledge, Northpool calculates the amount of electricity and gas required, thereby determining the price of electricity or gas. With the growth of intermittent renewable electricity generation, energy trading plays a bigger role in maintaining the fine balance between supply and demand. Alongside trading, Northpool also provides ancillary services to the system operators, to help balance the electricity grid.

[1] Gas hubs are at the heart of gas infrastructure networks, and are used as a central pricing point. In a way, they can be seen as an exchange for gas (derivatives).

[2] Review of the Gas Hub Assessment, European Federation of Energy Traders (EFET), 2021

Northpool has a database of very recent, high-resolution TTF month-ahead gas pricing data available, with no forecasting model yet employed. By high-resolution we mean that all trades are individually recorded, with exact timestamps. The idea is to combine trader knowledge with data preprocessing and machine learning techniques, producing a model which is not only accurate, but explainable as well. The model is inspired by the approach proposed by E et al. (2019).

The novel aspect of this thesis is be two-fold. First, the database containing high-resolution (i.e. real-time) gas pricing data presents a novel domain on which a combination of models has been employed. The dataset is unique in the sense that it contains prices with dynamics not previously seen before. Recent events heavily impacting the gas price are included, such as the COVID-19 crisis, and the Russian-Ukrainian conflict. This caused the price to show levels the market had never seen before. An important part of the research is going to be how to best handle these extended periods of time with elevated volatility and different price dynamics. The dataset and its characteristics are discussed in section 3.4.

The second novel aspect of the thesis is the combination of exogenous variables, data preprocessing and machine learning methods, applied to the EU TTF price. To address the issue of the volatile prices, we will be researching if the inclusion of exogenous variables helps us in forecasting. Up until now, data preprocessing techniques and machine learning methods have mostly been performed on gas pricing data alone, without considering exogenous variables. E et al. (2019) and Tang et al. (2018) combine preprocessing methods with various modelling techniques, but they do not include exogenous variables. Some examples of papers that combine preprocessing, machine learning (ML) and exogenous variables are Nguyen and Nabney (2010) and Abrishami and Varahrami (2011). Since these papers are published over 10 years ago, they handle modestly sized datasets, and relatively outdated (preprocessing) methods.

Besides these contributions, the proposed model will be applied to European gas price data. None of the models described in the literature review presented by Lu et al. (2021) model European gas prices, because most of the European gas hubs have opened relatively recently. One of the few papers modelling European (TTF) gas prices is published by Berrisch and Ziel (2022). They present a traditional time series method. Another paper which describes a forecasting method for the EU TTF price is presented by Ram et al. (2019). They describe the use of a simple Artificial Neural Network (ANN). We did not come across papers publishing an approach for the EU TTF price forecast combining data preprocessing with machine learning and exogenous variables.

The document is set up as follows: section 2 discusses previous publications concerning gas price forecasting. It starts by discussing the different data preprocessing techniques in section 2.1. Then, Section 2.2, 2.3 and 2.4 discuss the different modelling approaches. In section 2.5 the performance of some of the best performing gas forecasting models are presented. In section 3.7 the proposed model is presented, where we will take a deep dive into the theoretical background of the models and the preprocessing techniques. After we have discussed the theory of the forecasting approach, we present the research questions in section 3.8. Then, the proposed models are compared to a number of benchmarks in section 4.5, by presenting their performances. To conclude, the exogenous variables are presented in section 4.6 which have been included in the models, and again, a performance comparison is made.

2 Literature review

Many papers on forecasting the natural gas price (and other energy commodities such as oil and coal) exist. In the past ten years alone over 170 papers have been published, which have been summarised in a literature review by Lu et al. (2021). The main goal of this literature review is to make informed decisions on the techniques to include when building the forecasting model.

One of these decisions is the use of preprocessing methods. Preprocessing the input data can help improve the accuracy of the forecast, as shown by Nguyen and Nabney (2010), Tang et al. (2018) and E et al. (2019). Besides this, it can help us inspect the quality of the data, and extract extra information. Hopefully, the effects of the COVID-19 crisis and the Russian-Ukrainian conflict are already visible during preprocessing. Papers describing the use of preprocessing the input data in light of energy commodity forecasting are discussed in the next section. After selecting the preprocessing method, we need to make a decision on the modelling approach. For the sake of clarity we will group the literature review of the modelling approach into 3 categories: traditional methods, machine learning methods, and hybrid methods. Hybrid methods combine the best of both worlds, and have been shown to outperform their individual counterparts on some occasions (as shown by Chen et al. (2020), E et al. (2019), Wang et al. (2020)).

A decision which has to be made later on in the modelling process is the use of exogenous variables. The use of exogenous variables causes the model to become more complex. Besides complicating the model, another problem with the use of exogenous variables is the fact that in order to use them for forecasting, they have to be forecast themselves. Despite these drawbacks, they can improve the forecasting performance since they add extra information which could lead to increased accuracy.

Before discussing the literature review, it is useful to define some common performance measures used in most papers discussed in the literature review. Performance measures, as the name suggest, are used to compare performance between models. We will focus on 4 measures: the Root Mean Squared Error (RMSE), the Mean Absolute

Percentage Error (MAPE), the Mean Absolute Error (MAE), and the Mean Squared Error (MSE):

$$\begin{aligned}
\text{Root Mean Squared Error (RMSE)} &= \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \\
\text{Mean Absolute Percentage Error (MAPE)} &= \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \\
\text{Mean Absolute Error (MAE)} &= \frac{\sum_{i=1}^{N}|\hat{x}_i - x_i|}{N} \\
\text{Mean Squared Error (MSE)} &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2
\end{aligned} \tag{1}$$

Where $x_i,\ i \in \{1, ..., N\}$ is the actual series to be forecast, and where $\hat{x}_i,\ i \in \{1, .., N\}$ is the forecast itself. As far as they are presented in the papers reviewed, these performance measures are mentioned in table 1 to compare performances. When measuring performance, it is important to keep in mind what shortcomings certain performance measures inherently possess. It is important to note, for example, that the MAPE is the only scale invariant performance measure. On the other hand, it is not suitable to measure errors for series close to zero.

The RMSE, MSE and MAE are not scale invariant, meaning they are all dependent on the magnitude of the target variable being predicted. The larger the magnitude of the target variable, the larger the RMSE, MSE and MAE will likely be. This makes it hard to compare models not publishing a MAPE, since the prediction models are often assessed on different data sets.

2.1 Preprocessing of data

Some papers describing the use of preprocessing techniques for energy price prediction include E et al. (2019), Qin et al. (2019) and Tang et al. (2018). E et al. (2019) use Independent Component Analysis (ICA) together with Variational Mode Decomposition (VMD). Tang et al. (2018) and Qin et al. (2019) both use (Ensemble) Empirical Mode Decomposition ((E)EMD). Since at least one of these methods will be used in the research, a detailed explanation of these methods is described in section 3.6. This section will treat the papers using other preprocessing techniques.

The main idea behind preprocessing the data is to separate a time series[3] into different components, that together form the original series. These components might be easier to forecast, or they might contain information that helps the forecasting. One example of a paper which compares multiple data preprocessing techniques is written by Nguyen and Nabney (2010). They compare wavelet transforms (WT), extended Kalman filters (EKF) and particle filters (PF). The first method they discuss is the wavelet transform, which is used to split the target signal into multiple components. Nguyen and Nabney (2010) provide two approaches with the wavelet transform. In the so-called multicomponent forecast they forecast the components individually to combine them later on. The other approach they call the direct forecast, in this approach they use the components of the WT as input variables for a single forecasting model to directly predict the target variable.

The second preprocessing technique presented in the paper is filtering, by means of particle filtering and extended Kalman filtering. This is not preprocessing in the sense that it happens before the forecasting. It is rather used to update model parameters in an adaptive model whenever new values of the time series are observed. The filtering estimates weights of the models by treating them as states of a non-linear dynamic system. The EKF does this by approximating the unknown dynamics by linearising the non-linear functions defining the system, using a first order Taylor expansion. However, if the functions are strongly non-linear, the approximations made by the EKF are not good enough. This is where the PF comes in. PF is an alternative method which avoids the negative effects of linearisation, since it is a sampling based method.

2.2 Forecasting: traditional approach

Traditional models on predicting gas prices mainly come in 2 forms: time series models and econometric models. Econometric models rely on the causal relationship between the price and its driving factors (mainly supply and demand). Time series analysis on the other hand, does not make any causal assumptions about economic behaviour in forecasting a price, but merely uses the statistical properties of the data itself.

2.2.1 Econometric model

One of the first attempts at modelling the economics of exhaustable resources (including gas) has been presented by Hotelling (1931). Examples of econometric models on gas price forecasting are presented by Bopp (1990) and Hsieh

[3] A time series is a sequence of numerical values linked to a timestamp. A simple example of a time series is the stock price of any publicly listed company.

(1990). The main assumption of an econometric price model is that price is a consequence of supply and demand. If we model supply and demand, then price directly follows. In their work, Reiter and Economides (1999) suggests the following approach for predicting the American gas spot price using the econometric approach, described in system (2), where $P_{g(t)}$ is the price of gas, $q_{d(t)}$ is the gas demand, and $q_{s(t)}$ is the gas supply. $q_{ind(t)}, q_{com(t)}, q_{el(t)}, q_{res(t)}$ denote the industrial, commercial, electric utility and residential demand respectively. $q_{pro(t)}, q_{imp(t)}, q_{st(t)}$ represent the gas production, imports and the amount in storage, $p_{sf(t)}$ is the price of substitution fuels, a_i and b_i are parameters, and a_i^* and b_i^* are normalised parameters, $u(t)$ and $v(t)$ are error terms, and finally, t represents a point in time.

$$\begin{cases} q_{d(t)} &= a_1^* \cdot q_{ind(t)} + a_2^* \cdot q_{com(t)} + a_3^* \cdot q_{el(t)} + a_4^* \cdot q_{res(t)} + a_5^* \cdot P_{g(t)} + a_6^* \cdot P_{sf(t)} + u(t) \\ q_{s(t)} &= b_1^* \cdot q_{pro(t)} + b_2^* \cdot q_{imp(t)} + b_3^* \cdot q_{sto(t)} + b_4^* \cdot P_{g(t)} + v(t) \\ q_{s(t)} &= q_{(d(t)} \end{cases} \tag{2}$$

Making use of the equality in the system, and taking the first derivative with respect to time yields the following expression:

$$\begin{aligned} \Delta_{t-1}^t P_g = \alpha_1^* \Delta_{t-1}^t q_{pro} + \alpha_2^* \Delta_{t-1}^t q_{imp} + \alpha_3^* \Delta_{t-1}^t q_{sto} + \alpha_4^* \Delta_{t-1}^t q_{ind} \\ + \alpha_5^* \Delta_{t-1}^t q_{el} + \alpha_6^* \Delta_{t-1}^t q_{com} + \alpha_7^* \Delta_{t-1}^t q_{res} + w_t \end{aligned} \tag{3}$$

The data in Reiter and Economides (1999) is non-stationary, a standard regression model can not be used on non-stationary data. They dealt with this problem by taking the first difference of the data with respect to time, as shown above. Some of the variables are not of interest to short-term price modelling. Examples are the production, imports, industrial and electric utility consumption, which are only interesting for long-term price forecasting. The price of substitute fuels is not relevant for the short-term price prediction of gas, since the switch of fuels requires equipment adaption, and thus costs. Since neither residential nor commercial gas consumption data is available for short-term use, another way of modelling these variables need to be found. Luckily, it is well studied that the consumption strongly correlates with the average temperature in the area, so this is used as a proxy for residential and commercial consumption. The only variable which has not been discussed yet is the gas storage data, which is published on a weekly basis, Reiter and Economides (1999) omit this variable in their analysis.

Short term gas prices have a very local nature, due to, amongst other things, the weather. This is why Reiter and Economides (1999) consider prices in 3 different locations and model them individually: Henry hub, Houston-Ship-Channel and the city-gate price in New York City. From equation (3) the generalised model based on average temperature is derived. The target variable is the two-day price difference,$SP_t - SP_{t-2}$, since this is most relevant for day-trading gas contracts. The following two expressions are derived for the generalised model:

$$(SP_t - SP_{t-2}) = C + \alpha * (AT_{t+1}^{HN} - AT_{t-1}^{HN}) + u_1 \tag{4}$$

$$(SP_t - SP_{t-2}) = C + \alpha * (AT_{t}^{NC} - AT_{t-2}^{NC}) + u_2. \tag{5}$$

AT^{NC} and AT^{HN} denote the average daily temperatures in the New York-Chicago region and in the Houston-New Orleans region respectively. u_1, u_2 are independent error terms. Equation (4) applies to the Henry Hub price and the Houston-Shipping-Channel price, and (5) refers to the New York city gate price. The difference in the expression has to do with different weather situations in the two areas. Weather systems that determine the temperature in Chicago often move to New York City 2 days later. In the Houston-New Orleans case the gas which is sold today, will not flow until tomorrow, so the next day temperature is a significant factor. After relating the temperatures to the short term gas price, in lack of other variables with daily observations, Reiter and Economides (1999) relate the futures price to the short term price. The correlations between the variables in Reiter and Economides (1999) have been researched using the program E-views. Finally, they arrive at the following (normalised) expressions for the 2-day price difference for gas:

$$(SP_t - SP_{t-2}) = -0.36 \cdot (AT_{t+1} - AT_{t-1}) + 0,46 \cdot (FP_{t-1} - FP_{t-2}) + u_1 \tag{6}$$

$$(SP_t - SP_{t-2}) = -0.37 \cdot (AT_{t+1} - AT_{t-1}) + 0,45 \cdot (FP_{t-1} - FP_{t-2}) + u_1 \tag{7}$$

$$(SP_t - SP_{t-2}) = -0.40 \cdot (AT_{t+1} - AT_{t-1}) + 0,37 \cdot (FP_{t-1} - FP_{t-3}) + u_1 \tag{8}$$

Where (6),(7) and (8) refer to the Henry Hub, Houston Ship Channel and New York city gate prices respectively.

2.2.2 Time series model

Some common time-series models include the Autoregressive (AR(p)), the Moving Average (MA(q)), the Autoregressive Moving Average model (ARMA(p,q)) and the (Seasonal) Autoregressive Integrated Moving Average model ((S)ARIMA(p,d,q)). These models make use of specific properties of the time series itself such as autocorrelation, trends or seasonality. In contrast to the econometric models, time series models do not use fundamental economic

research to make the forecasts. An overview of widely used models is shown below. $X_1, ..., X_n$ denote the (chronologically ordered) samples in the time series, and $1 \leq t \leq n$ is the point in time we want to forecast.

$$X_t = \mu + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \varepsilon_t \qquad \text{MA(q)} \tag{9}$$

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \varepsilon_t \qquad \text{AR(p)} \tag{10}$$

$$X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \sum_{i=1}^{p} \phi_i X_{t-i} \qquad \text{ARMA(p,q)} \tag{11}$$

$$X_t - \alpha_1 X_{t-1} - ... - \alpha_p X_{t-p} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + ... + \theta_q \varepsilon_{t-q} \qquad \text{ARIMA(p,d,q)} \tag{12}$$

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 \qquad \text{ARCH(q)} \tag{13}$$

The MA(q) process assumes that the time series is a combination of a long-term average μ, and q lagged, white-noise error terms, ε_{t-i}, ε_t. The AR(p) model assumes the price at time t is a linear combination of p previous prices together with a single error term ε_t. The ARMA model is simply a combination of the moving average model MA(q) and the autoregressive model AR(p). The ARIMA(p,d,q) model is defined implicitly in equation (12). The variables in the ARIMA model denote the number of autoregressive terms (p), the number of moving average terms (q) and the degree (d) of differencing performed on the original time series to obtain stationarity. The autoregressive conditional heteroskedasticity (ARCH) process models the volatility of the error in a time series. If the error is assumed to be ARMA, then the ARCH process is called a GARCH (generalised ARCH) process. These methods are fairly straightforward, and many software packages exist which execute them directly for the user.

As an example of applying the ARIMA model to forecast gas prices in Iran/US, Mishra (2012) makes a nice comparison between the parametric approach (ARIMA), and a non-parametric approach (ACE). More complex models are readily available, since a lot of work has been done on forecasting gas prices, both for the short- and long-term contracts. Berrisch and Ziel (2022) present a very good performing time series approach in their publication. They model two contracts, the day-ahead contract and the month-ahead contract.

2.2.2.1 Berrisch and Ziel day-ahead model

Berrisch and Ziel (2022) propose two probabilistic models for predicting the TTF month-ahead and day-ahead gas price, using data between 2011 and 2020. The paper is a nice example, since they present rigorous data and model validation. They are very experienced in the gas markets and they perform extensive analysis on their exogenous variables. Moreover, they benchmark their models not only against traditional methods such as an ARIMA model and a Vector Autoregressive (VAR) model, but also against one of the best performing ML methods known at that time, the random forest model described by Herrera et al. (2019). The models presented by Berrisch and Ziel (2022) outperform all benchmark models by at least 9% for the day-ahead price, and by 13% for the month-ahead price.

Berrisch and Ziel (2022) present the following model for the day-ahead gas price:

$$y_t = l_{t-1} + \underbrace{\frac{\psi_1}{5} \sum_{s=t-A-2}^{t-A+2} y_t}_{\text{Seasonal}} + \underbrace{\zeta_1 \text{Coal}_{t-1}}_{\text{Coal}} + \underbrace{\zeta_2 \text{EUA}_{t-1}}_{\text{EUA}} + \underbrace{\zeta_3 \text{PWR}_{t-1}}_{\text{Power}} + d_t \tag{14}$$

$$l_t = \underbrace{l_{t-1} + \lambda d_t}_{\text{Exp. Smoothing}} \tag{15}$$

$$d_t = \underbrace{\sum_{i=1}^{p} \phi_i d_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}}_{\text{ARMA Errors}} + \varepsilon_t \tag{16}$$

$$\varepsilon_t \sim \text{SST}(0, \sigma_t, \nu, \tau) \tag{17}$$

$$\sigma_t = \begin{cases} \delta \tilde{\sigma}_t, & \text{Monday} \\ \tilde{\sigma}, & \text{Else} \end{cases} \tag{18}$$

$$\tilde{\sigma}_t = \underbrace{\underbrace{\omega + \alpha |\varepsilon_{t-1}| + \beta \sigma_{t-1}}_{\text{Absolute Value GARCH}} + \underbrace{\gamma |\varepsilon_{t-1}| \mathbb{1}_{(\varepsilon_{t-1} < 0)}}_{\text{Leverage}}}_{\text{TGARCH}} \tag{19}$$

This model is a state space model (SSM) , since we assume that the observed variables (y_t) are a direct consequence of the unobserved state variables: l_{t-1}, the seasonality ψ_1 and the exogenous variables for coal, EUA and power, where the dynamics of the state variables are defined. The exogenous variables used are the month-ahead Rotterdam coal price, the German day-ahead and month-ahead power base and power peak prices, and the daily spot prices of European Emission Allowances (EUA). The autoregressive structure of the time series is modelled using exponential smoothing, with λ as the smoothing factor (a linear exponential smoothing model is a special type of ARIMA model). The seasonality is modelled as a weekly moving average, lagged by one year (A), looking 2 days ahead and 2 days back, since the usual week contains 5 observations. As mentioned earlier, since the error terms d_t are modelled as ARMA processes, this means they can model the volatility of the errors with a GARCH process. In fact, they used a Threshold GARCH process (TGARCH). The TGARCH process reduces the influence of large errors, which occur quite often because of the heavy tailed distribution of the gas prices. Berrisch and Ziel (2022) state volatility in the gas markets is higher on Mondays, so they increase the volatility on Mondays by leveraging the estimated standard deviation. Lastly, they model the error terms, ε_t using a skewed-student-t distribution, which features heavier tails than the normal distribution. They argue this distribution suits the data better, noting that it allows for skewness. A strong suit of this paper is that they present the significance of each of their model components, which substantiates the choices they made when designing the model.

2.2.2.2 Berrisch and Ziel month-ahead model

Since Berrisch and Ziel (2022) found autoregressive models did not outperform simple random walk models for the month-ahead price forecast, they built an extended random walk model to forecast the month-ahead price. The model for the month-ahead gas price is defined as:

$$y_t = \phi_0 + \underbrace{\Phi_t}_{\text{Risk Rollover}} + \psi_1 \bar{y}^M_{t-1Y} + \zeta_1 \text{EUA}_{t-1} + \zeta_2 \text{Oil}_{\text{t-1}} + \zeta_3 \tilde{T}_{t-1} + \varepsilon_t \tag{20}$$

$$\Phi_t = \begin{cases} \phi_1 y_{t-1,2MA}, & \text{First Trading Day per Month} \\ \underbrace{\phi_1 y_{t-1}}_{\text{Rollover}} + \underbrace{\eta \frac{D^{LTD}_t}{D^M_t}}_{\text{Risk Deduction}}, & \text{Else} \end{cases} \tag{21}$$

$$\varepsilon_t \sim \text{SST}(0, \sigma_t, \nu, \tau) \tag{22}$$

$$\sigma_t = \begin{cases} \delta \tilde{\sigma}_t, & \text{First Trading Day per Month} \\ \tilde{\sigma}, & \text{Else} \end{cases} \tag{23}$$

$$\tilde{\sigma}_t = \omega + \alpha |\varepsilon_{t-1}| + \beta \sigma_{t-1} + \gamma |\varepsilon| \mathbb{1}_{(\epsilon_{t-1}<0)} \tag{24}$$

Traders can rollover month-ahead contracts when they are close to their delivery date, essentially extending the lifetime of the product with one month. As such, Berrisch and Ziel (2022) used the price of the two-month-ahead product $y_{t-1,2MA}$ on the first day of the trading month (when most rollovers occur) as a predictor for the month-ahead price. Additionally, they add a seasonal component, by adding the monthly average lagged by one year $\bar{y}^M_{t-1Y}$. By including the decay in risk premia and seasonality, they address the monthly structure of the product (i.e. the rollover) better than with Fourier terms.

An important factor here is that they (both) are very experienced in the gas market, which gives them a significant advantage in modelling the time series compared to specialised mathematicians. Combining specialised trader experience with mathematical modelling is something which we will try to incorporate in our model as well. Furthermore, they include a set of exogenous factors such as the lagged prices of oil Oil_{t-1}, the EU Emission Allowances EUA_{t-1} and the lagged and smoothed German temperature $\tilde{T}_{t-1}$. The temperature is exponentially smoothed using a smoothing parameter of 0.95: $\tilde{T}_t = (0.95)\tilde{T}_{t-1} + (0.05)T_t$. As in the day-ahead model, the volatility is modelled using a TGARCH process, the only difference is that the conditional standard deviation is only elevated on the first trading day of every month, instead of on Mondays. The errors are student-t distributed, as with the day-ahead model.

2.3 Forecasting: machine learning approach

Unless stated otherwise, all papers discussed in this section model the US gas price. This is why these papers will be discussed less in depth than the Berrisch and Ziel (2022) paper. Nevertheless, it will give a short overview of a subset of methods employed in the past years concerning the application of machine learning for gas price forecasting. The performance metrics of the models are only discussed in terms of MAPE. If no performance is mentioned, no MAPE was presented in the paper. Since many papers have been written about different neural network approaches, these papers will be discussed separately in section 2.3.1. Section 2.3.2 will discuss the other machine learning methods

employed, such as random forests and support vector machines (SVM). A deep dive into SVM will be presented in section 3.7.3.

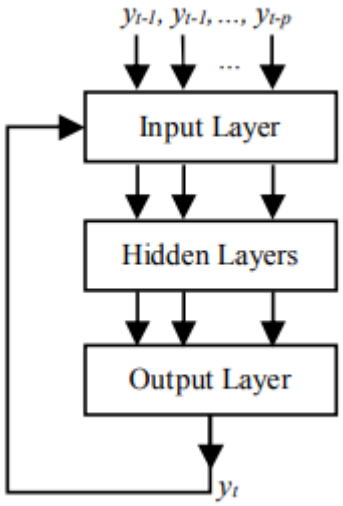
2.3.1 Artificial Neural Networks

The number of papers describing new neural network architectures are bountiful, there is a whole zoo of different architectures, all with their specific uses. For a visual overview, fig. 14 gives a non-exhaustive overview of the different architectures described up until 2019. The figure is probably outdated already, since new architectures are invented often. An early example which uses Artificial Neural Networks (ANN) to forecast natural gas prices has been written by Reiter and Economides (1999). The main motivation to write their paper was that, at the time, all known gas price prediction models tried to forecast a long-term mean or equilibrium price. While this is useful for long-term strategies, it fails to address the volatile gas price on a short-term basis. Reiter and Economides (1999) tested their models in a trade simulation and found that the ANN outperformed the econometric model in every scenario.

A relevant paper for this thesis is Ram et al. (2019), where the prediction of the gas price in several EU gas hubs (such as TTF, NCG and PECG) using a plain ANN is described. Unfortunately, it is a short paper only publishing an R^2 performance measure, without a lot of in-depth analysis of the ANN or the data. Nevertheless, it is the only example in the literature where an ML-method is used to forecast European gas prices.

In Nguyen and Nabney (2010) they combine several data preprocessing techniques such as Particle Filters, (Extended) Kalman Filters and Wavelet Transfroms with adaptive Multilayer Perceptrons (MLP) and Radial Basis Function Neural Networks to model the NBP gas price in the UK. Nguyen and Nabney (2010) also include a number of exogenous variables such as economic parameters and meteorological information. They find the best results, out of the 36 models tested, by using an adaptive MLP, and that preprocessing the data improves performance for most models.

Another example of a simple ANN being used to forecast gas prices is presented in Salehnia et al. (2013). They used the Gamma test for the first time as a non-parametric, non-linear, smooth modelling tool to choose the best input combinations for the model, before calibrating and testing the models. They compare Dynamic Local Linear Regression (DLLR), Local Linear Regression (LLR) and ANNs to each other. Salehnia et al. (2013) observe that the ANN has a superior performance compared to LLR and DLLR. They conclude that although the MSE of the DLLR forecast is lower than that of the ANN, the price prediction is not particularly impressive and is quite noisy. The ANN model presents a high accuracy for the gas spot price trend in different timescales. Its ability to predict price shocks is not considerable though, as was assumed from the start. In 2015, Jin and Kim (2015) proposed to preprocess their data using the Discrete Wavelet Decomposition (DWD), after which they employ a standard ANN to do the forecasting. Jin and Kim (2015) then compare the results of the DWD-ANN to a regular ARIMA model and a DWD-ARIMA model. They find that the performance of the ANN and the ARIMA increases after preprocessing the data. Also, the DWD-ANN approach outperforms both the regular ARIMA and the DWD-ARIMA approach. Besides these findings, they conclude that including individual components from the DWD increases performance slightly. The best MAPE they achieve is 3.29%. For a full overview of performances published in the papers we refer to Lu et al. (2021) and table 1. More recent work has been presented by Siddiqui (2019), who claims to have improved on Salehnia et al. (2013) by 58% when considering MSE. Siddiqui (2019) proposes an Autoregressive Neural Network (ARNN), to capture the autoregressive structure present in time series, as described in classical time series models as well. The ARNN feeds part of the output of the architecture back into the input layer, as shown in fig. 1.



Source: Siddiqui (2019)
Figure 1: The autoregressive neural network architecture as proposed by Siddiqui (2019), (part of) the output is fed back into the network.

Even though the data sets used in Siddiqui (2019) and Salehnia et al. (2013) overlap, it is hard to say something useful about the improvement in MSE of 58% proposed by Siddiqui (2019), since we know little about the model parameter settings, or other details in the implementation of both models.

Other works on predicting the US gas price using ANNs are presented by Tang et al. (2019), who uses internet searches and news sentiment as exogenous variables to predict natural gas prices, and Tang et al. (2018), where they compare several randomization algorithms such as Extreme Learning Machines, Random Kitchen Sinks, and Random Vector Functional Links to speed up training when combined with computationally heavy data preprocessing techniques such as EEMD. For a more complete overview of ANN models applied to gas price prediction, we refer to Lu et al. (2021).

2.3.2 Other methods

Besides neural networks, there is a broad range of other methods considered as machine learning. Two of the most commonly used are the support vector machine and the random forest method. These methods will be discussed in this section. The random forest method combines decision trees in an optimal way to minimise a certain loss function. Support vector machines apply the kernel trick to fit a hyperplane to maximally separate two sets of samples.

2.3.2.1 Random Forests

One example using random forests to predict gas prices is written by Herrera et al. (2019). In their paper, Herrera et al. (2019), discuss 3 different models. For the first model, they combine 5 traditional statistical models into one hybrid model, using ARIMA, ETS, LOESS, TBATS and a Theta model (the abbreviations can be found in the nomenclature). They combine the five models using weights. The weights are determined by performing a non-rolling time series cross validation to minimise the RMSE. Besides this hybrid model, they employ a feed-forward multilayer perceptron and a random forest (RF) model. The RF model makes use of the 'divide and conquer' paradigm, where a problem is split into smaller, simpler problems. It does this by repeating 3 steps: sampling fractions of the data, build a randomised tree predictor and aggregate these predictors by averaging. Breiman (2001) emphasises random forests are an effective tool in prediction, and one of their main advantages is that they don't have a tendency to overfit, as some of the other ML methods (such as ANNs) do. However, the problem both the MLP and the RF model have is that there is no optimal way of determining their ideal parameter setup. This means a parameter grid with combinations of parameters has to be systematically tested, which can prove to be quite time-consuming. In case of the MLP, Herrera et al. (2019) tested 1400 combinations of parameters (number of nodes, number of layers etc.), with which they performed a forecast and measured performance. For the random forest approach, a thousand different parameter settings were tested to find a good setting. The ideal architecture of the network forecasting the US gas price was a setup with 2 hidden layers with respectively 10 and 5 nodes. For the random forest method, 500 random trees turned out to be optimal.

To measure the performance of their models, and compare them to each other, Herrera et al. (2019) use the RMSE and MAPE. Herrera et al. (2019) use the modified Diebold Mariano test to quantify the statistical significance of the predictive accuracy of the different models. Additionally to the three models described earlier, a fourth benchmark

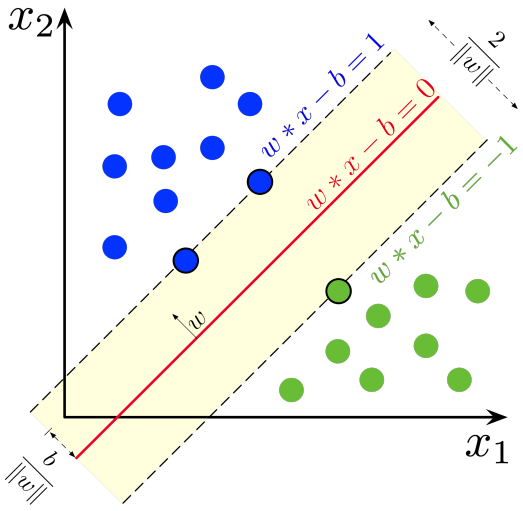
model is added: the no-change model, which is a random walk process with drift 0. When comparing the results, they find the random forest model outperforms all benchmark models.

Other works describing the use of random forest include Su et al. (2019a), where a least-squares boosting (LS-Boost) approach to random forests is proposed for gas prediction. The main methods to improve the regression performance of random forests is to apply bagging or boosting. Both methods apply to ensemble random forests, where a big number of decision trees is generated and combined to decrease variance and bias of the method. Boosting is the notion of turning weak learners (i.e. a predictor with slightly better performance than random guessing) into strong learners (i.e. a predictor with much better performance than random guessing). To quantify the performance of the learners, and to identify the weak learners, many different loss functions can be used. LSBoost uses a least squares loss function. Another example of boosted random forests is described in Su et al. (2019b), where they use a Gradient Boosting Machine (GBM), which uses gradient descent based loss functions to measure and improve the performance of the learners.

A second way to improve the regression performance of random forests is by bagging (bootstrap aggregation). It employs a different strategy than boosting, which is more straightforward. It averages the regressions of all the generated decision trees to form a more balanced result. Mouchtaris et al. (2021) evaluate the use of bagged trees for natural gas spot price forecasting in their paper, where they use oil prices and stock indices as exogenous variables to improve performance. They compare the bagged trees to Support Vector Machines and Gaussian Process Regression, and find the bagged trees and linear SVM have superior performance compared to the other methods.

2.3.2.2 Support Vector Machines

Support vector machines can be used for classification, but also for regression. The main idea of SVM is to maximise linear separability between two sample groups, with marked samples belonging to one of two groups. To do this, SVM (non-linearly) maps the training samples, stored in the vector $\mathbf{x}$, to points in higher-order space. The mapping of training samples to a higher-dimensional space is called the kernel trick. Once the kernel trick has been done, SVMs maximise the space between the two groups by fitting a hyperplane which maximally separates the two sample groups. It achieves this by adjusting the weights $\mathbf{w}$ and bias b, which define the hyperplane. If this hyperplane $\mathbf{w}^\mathsf{T}\mathbf{x} + b$ maximises $\frac{2}{||\mathbf{w}||}$, the space between the two training groups, it is called the maximum margin hyperplane. For a visual interpretation of the maximum margin hyperplane see fig. 2. This paradigm can be applied to regression problems, which is described in more detail in section 3.7.3.



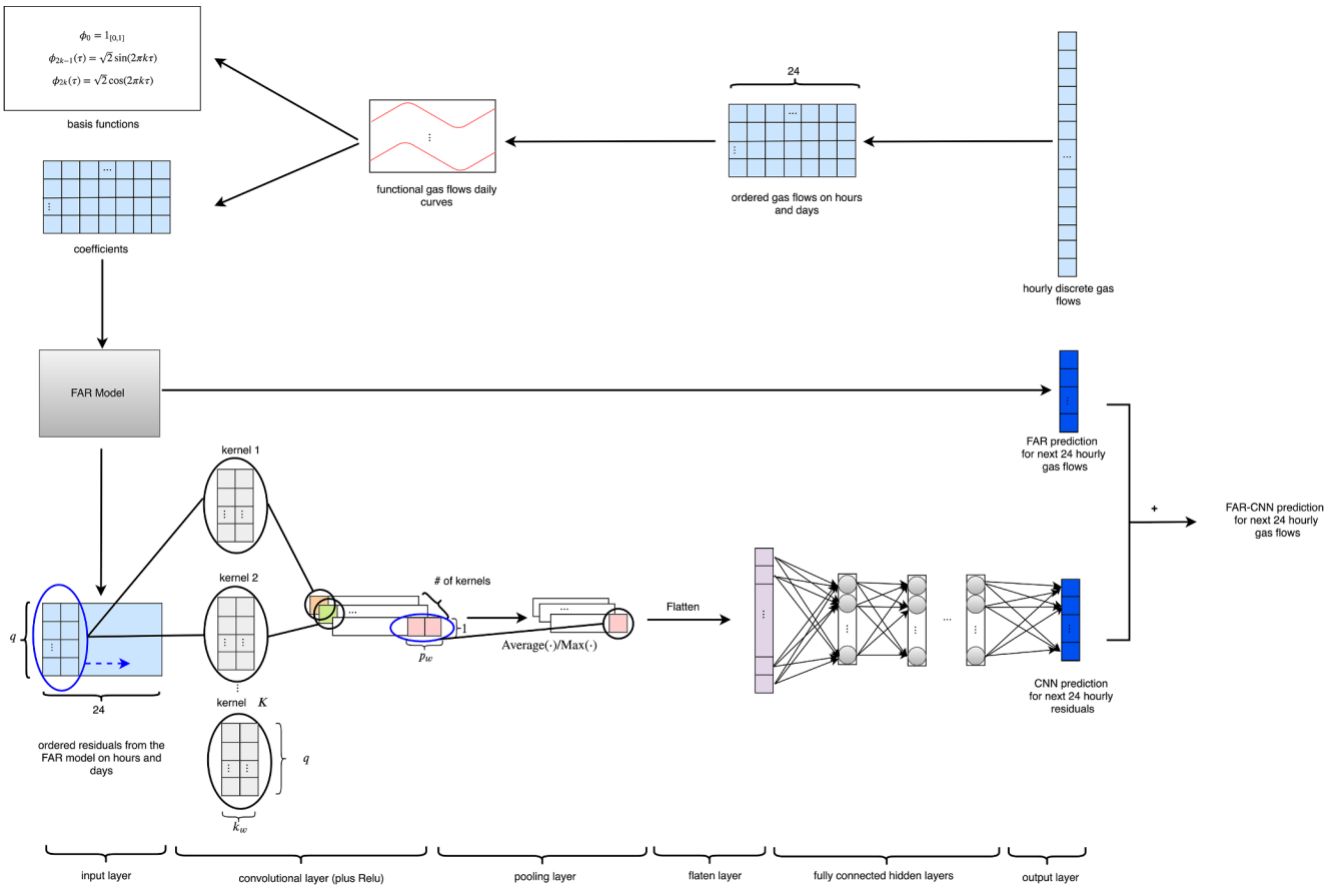
Source: Larhmam

Figure 2: Visual representation of the maximum margin hyperplane in 2 dimensions. The green training examples belong to one group, and the blue training examples to another.

2.4 Forecasting: hybrid approach

Some work has been done on hybrid models, which combine traditional models with machine learning techniques. One approach is to model the linear components in the pricing data with traditional (time series) methods. The residual non-linear effects are then handled by the machine learning models. A different approach is to employ multiple (machine learning) methods and optimise the way in which they are being combined to fit the data best. One danger of this last approach is that, although it gives great performance, it goes at the cost of generalisability of the model. We will discuss models employing both approaches.

One example of the first approach is the work of Chen et al. (2020), who propose to use a hybrid model combining a traditional Functional Autoregressive model (FAR) together with a Convolutional Neural network (CNN) to forecast demand in 92 gas distribution nodes in Germany. Their FAR-CNN model outperforms all benchmark models (FAR, AR, ANN) on a general level. FAR-CNN is only outperformed by a simple AR model for border distribution nodes. Since gas flow is very stable in these nodes, a simple model is sufficient. The idea of their method is that the linear correlations are modelled using the autoregressive model. The residuals left behind by this approach still contain non-linear effects which have not been exploited yet. The residuals are being fed into the CNN to model the last non-linear effects as to improve performance even further. A schematic overview of this approach is shown in fig. 3.



Source: Chen et al. (2020)

Figure 3: The hybrid FAR-CNN method, where the residuals of the FAR form the input of the CNN.

An example of the second approach, where multiple models are optimally combined, is the work done by Naderi et al. (2019). They combine SVM, ANN, ARIMA and Genetic Programming (GP) using a meta-heuristic bat algorithm to ensure the combination is optimal. The bat algorithm is based on echolocation used by bats to find its prey by changing pulsation rate and loudness. For more information we refer to Yang (2010). Naderi et al. (2019) achieve a good MAPE score of 1.49%, although Wang et al. (2020) suggest that the model is too closely tied to its data (i.e. it is overfitting). This is a risk when optimising weights in a hybrid model. The weights are optimised in a way that best fits the data, however this does not guarantee that the model will perform as well on unseen data. Wang et al. (2020) themselves propose a hybrid approach using Support Vector Regression (SVR), improved pattern similarity search (IPSS), and LSTM. They combine the models using a weighted average, giving more weight to predictions which are closer to the real sequence. They show that the hybrid approach outperforms the individual counterparts, and their best MAPE score equals 5.04%. There are also papers describing a more simple approach to combining the models into a single, hybrid model. Karabiber and Xydis (2020) simply combine a number of methods by taking their average for example. They pick the best performing combinations such as ARIMA combined with a single layer neural network, and this way they achieve a MAPE of 3.8%. In their case most of the hybrid models perform better than their individual counterparts as well, apart from one occasion.

2.5 Performance comparison of discussed methods

Table 1 shows a selection of the papers from the literature review, together with a selection of the performance measures discussed.

Paper	Model	RMSE	MAPE	MAE	MSE
Berrisch and Ziel (2022)	Custom probabilistic	1.0843	-	0.3863	-
Mouchtaris et al. (2021)	Bagged trees	0.057	-	-	-
Karabiber and Xydis (2020)	Custom hybrid	77521	3.8%	58706	-
Wang et al. (2020)	LSTM-SVR-IPSS-model	0.174	5.04%	-	0.0301
Chen et al. (2020)	FAR-CNN	-	11.90%	-	-
Su et al. (2019a)	LSBoost	0.6615	-	0.4493	0.4376
Su et al. (2019b)	ANN	0.7247	11.17%	0.5115	0.5363
Herrera et al. (2019)	RF	0.4151	10.60%	-	-
Naderi et al. (2019)	LSSVM, GP, ANN, ARIMA hybrid	0.06	1.49%	-	-
Siddiqui (2019)	ARNN	-	-	-	0.026
Čeperić et al. (2017)	SVR with FS	27.82	4.30%	-	-
Salehnia et al. (2013)	ANN	-	-	-	0.3366
Nguyen and Nabney (2010)	Multicomponent adaptive GARCH	-	1.82%	2.066	-
E et al. (2019)	ICA-VMD-GRUNN-SVR	0.1009	0.17%	-	-
Qin et al. (2019)	EEMD-LLR	0.035	1.24%	-	-
Tang et al. (2018)	EEMD-ELM	-	0.5%	-	-

Source: Lu et al. (2021), own additions

Table 1: Overview of modelling approaches and performance measures of a selection of recent papers on gas price and demand forecasting.

For an exhaustive overview of published papers, we refer to Lu et al. (2021), which reviews 171 energy commodity price forecasting papers in the 10 years leading up to 2021, of which 19 concerning gas price prediction models.

For now we will focus on comparing MAPE scores to each other, since this is the only scale-invariant measure, as mentioned earlier. Doing so, we observe that the ICA-VMD-GRUNN-SVR approach proposed by E et al. (2019) presents the best score. This does not necessarily mean it is also the best model for forecasting the gas price. It could be tailored specifically to a given dataset, and may even overfit a little bit, compromising its performance on unseen data.

However, the reason we'll be employing the ICA-VMD-GRUNN-SVR is that it combines some of the latest methods in preprocessing and forecasting. Hopefully, ICA will prove especially useful to explain the independent influence of every exogenous (or endogenous) factor on the gas price, which will be discussed in section 3.6. It may also help in explaining the characteristics of our non-representative dataset, which will be discussed in more detail in section 3.4. VMD has been developed quite recently as an alternative to (E)EMD, both methods will be described in section 3.6 as well. The Gated Recurrent Unit Neural Network (GRUNN) is a special form of Long Short Term Memory (LSTM). LSTM has proven to be a useful tool for forecasting stock movements, because of the dynamics of pricing time series. GRUNN is a simplification of LSTM, which is faster and less cumbersome to train. Finally, SVR has proven to be a solid forecaster of gas prices in earlier work such as in the work of Čeperić et al. (2017). A general notion which has emerged in the last years is that hybrid models are able to outperform their individual counterparts (Chen et al. (2020), Wang et al. (2020)), which, in theory, makes the combination of SVR and GRUNN even stronger. Besides this fact, numerous papers have proven that preprocessing data could lead to better performance in terms of forecasting accuracy.

3 Research proposal

3.1 Introduction

The raw dataset used in this research is a set of 2,015,902 transaction records registered between 02/01/2020 and 08/03/2023. The dataset is provided by the Intercontinental Exchange (ICE) Endex Energy Exchange for Gas and Power. Only the relevant columns were used: timestamp, quantity, price and contract symbol. A description together with the data types is provided in table 2.

Column name	Data type	Description
Timestamp	DateTime	Date and time (up until second) when transaction took place
Quantity	int64	Quantity of contracts bought or sold
Price	float64	Price paid
Contract symbol	string	Identifier to mark which kind of contract has been traded

Table 2: Data columns of raw dataset

The contract symbol has a specific format, which is shown in 25.

$$\underbrace{\text{TFM}}_{\substack{\text{Physical}\\\text{code}}}\ \overbrace{\text{M}}^{\substack{\text{Letter denoting delivery month}\\\text{of contract}}}\ 00\ \underbrace{\text{YY}}_{\substack{\text{Two digits denoting}\\\text{expiry year of contract}}}\ ! \tag{25}$$

For example, TFM FMF0020! denotes the future contract with delivery during January 2020. About the delivery, the exchange states:

> "Delivery is made equally each hour throughout the delivery period from 06:00 (CET) on the first day of the month until 06:00 (CET) on the first day of the next month."[4]

On the trading period they mention the following:

> "Trading will cease at the close of business two UK Business Days prior to the first calendar day of the delivery month."[4]

All of the letters denoting the delivery months are mentioned in table 3.

F	January	N	July
G	February	Q	August
H	March	U	September
J	April	V	October
K	May	X	November
M	June	Z	December

Table 3: Delivery months with corresponding letters.

All contracts are traded for a specific delivery month in the future. To analyse the 'month-ahead' gas price, we need to do some filtering and adjustments. We define the 'month-ahead' contract as the contract where the delivery month is the month after the month in which the contract is traded. If we want to find all the transactions in 2020 concerning month-ahead contracts, we consider all transactions registered in January 2020 concerning the TFM FMG0020! contract (delivery in February). We append these transactions to all the transactions in February 2020 concerning the TFM FMH0020! contract. If we continue this process for the entire year of 2021, we get a subset of the original dataset, with only transactions relating to the month-ahead contract. The last transactions in this dataset will be concerning the TFM FMF0021! contract (with delivery in January 2021), since it is the month-ahead contract in December 2020. It is important to note that contracts with different delivery months rarely trade at the same level. They are subject to different dynamics and have a different delivery month. This means we can not simply splice them together, since a gap would occur at the change of the month. That is why, before filtering the transactions, we need to adjust the transactions of all the previous contracts based on the price of the most recent month-ahead contract. To do this, we first introduce an important concept, the volume-weighted average price.

[4] Intercontinental Exchange (ICE) Endex Energy Exchange for Gas and Power

3.2 Volume-Weighted Average Price

An important metric when dealing with any type of tradeable good, is the Volume-Weighted Average Price (VWAP). It is defined as:

$$\text{VWAP} = \frac{\sum_{i=0}^{n} p_i \cdot q_i}{\sum_{i=0}^{n} q_i}. \tag{26}$$

Where q_i denotes the quantity of transaction i, and p_i the price. In our case, the quantity q_i denotes the number of standard contracts traded per transaction. n denotes the total number of transactions which occurred during the time span for which the VWAP is calculated. The daily VWAP of contracts with subsequent delivery months is plotted in fig. 4, where the prices of the contracts is recorded in euro per megawatt-hour (MWh). The daily VWAP denotes the VWAP taken over all transactions occurring in a single trading day.

Figure 4: Daily VWAP calculated for each month-ahead contract

All of the contracts plotted in fig. 4 are month-ahead contracts, but with different delivery months. The time series is definitely not continuous, which is a prerequisite for analysing the time series. Especially between the price series in August, September and October 2020, there are large differences between the subsequent month-ahead contract prices. To solve this issue, we need to adjust the prices of the transactions in the month-ahead contracts in a meaningful way.

3.3 Panama Backwards Adjustment

There are different ways to make the price series continuous. We have chosen to use backwards Panama rolling. Backwards means that the price of the contract with the most recent delivery month will be taken as the unadjusted price, and that all prices of contracts with older delivery months will be adjusted to create a continuous price series, matching the price of the contract with the most recent delivery month.

Suppose we want to adjust the month-ahead price of January 2020 (the 'old' contract, with delivery month February 2020) such that it forms a continuous price series with the month-ahead price of February 2020 (the 'new' contract, with delivery month March 2020). First, the so-called roll date is defined. The roll date is the date that the new contract price will be in the adjusted price series. The roll date is the day prior to the last day of trading. To adjust the price of the old contract to match the price of the new contract, a 'roll gap' is calculated. The roll gap is defined as the difference between the new and old VWAP, calculated over the second to last hour of trading prior to the roll date (16.00-17.00 CET).

$$\text{Roll gap} = \text{VWAP}_{\text{new}} - \text{VWAP}_{\text{old}}. \tag{27}$$

To adjust the price of the old contract, the roll gap is added to it. All dates are schematically shown in table 4 for February 2021.

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
21	22	23	24	25	26	27
28		VWAPs calculated 16.00-17.00	Roll date	Last day of trading	Last business-day	

Table 4: All relevant dates in February 2021 for backwards Panama adjustments

Suppose we want to create a continuous time series out of m different contracts (for m months). We thus have m collections of transaction data, where m is the most recent pricing series available. Define them as $T_i = \{(p_1^i, q_1^i, t_1), ..., (p_{n_i}^i, q_{n_i}^i, t_{n_i})\}$, $i \in \{1, ..., m\}$, where n_i is the number of transactions in month i, p_1^i is the price of transaction 1 in month i, q_1^i is the quantity of transaction 1 in month i, and t_1 is the time transaction 1 occurred. The final output of the Panama rolling should be the union of all the adjusted transaction sets: $T_1^A \cup ... \cup T_m^A$.

Algorithm 1: Backwards Panama rolling

Data: A collection of month-ahead contract transactions: $\{T_1, ..., T_m\}$

Result: A single set containing all the adjusted transactions: $\{T_1^A, ..., T_m^A\}$

function `PanamaAdjustment`$(T_1, ..., T_m)$**:**

  for $i = m - 1 : 1$ **do**

    for $k = 1 : n_i$ **do**

$$p_k^i = p_k^i + \textsc{RollGap}(T_i, T_{i+1}) \tag{28}$$

    end

  end

  `/* Appending adjusted series to create single continuous series */`

  return $T_1^A \cup ... \cup T_m^A$

end function

function `RollGap`(T_i, T_{i+1})**:**

  `/* Calculates the roll gap for two subsequent contracts */`

  return $\text{VWAP}(T_{i+1}) - \text{VWAP}(T_i)$

end function

function `VWAP`(T_i)**:**

  `/* Calculates the VWAP for a given contract transaction data set */`

  for t_i between 16.00-17.00 on roll date **do**

$$\text{WeightedPrice} \leftarrow \sum_{i=0}^{n} p_i \cdot q_i \tag{29}$$

$$\text{TotalVolume} \leftarrow \sum_{i=0}^{n} q_i \tag{30}$$

  end

  return WeightedPrice/TotalVolume

end function

If we plot the same month-ahead prices as in fig. 4, but now add the continuous Panama adjusted series in the plot, we get the following image:

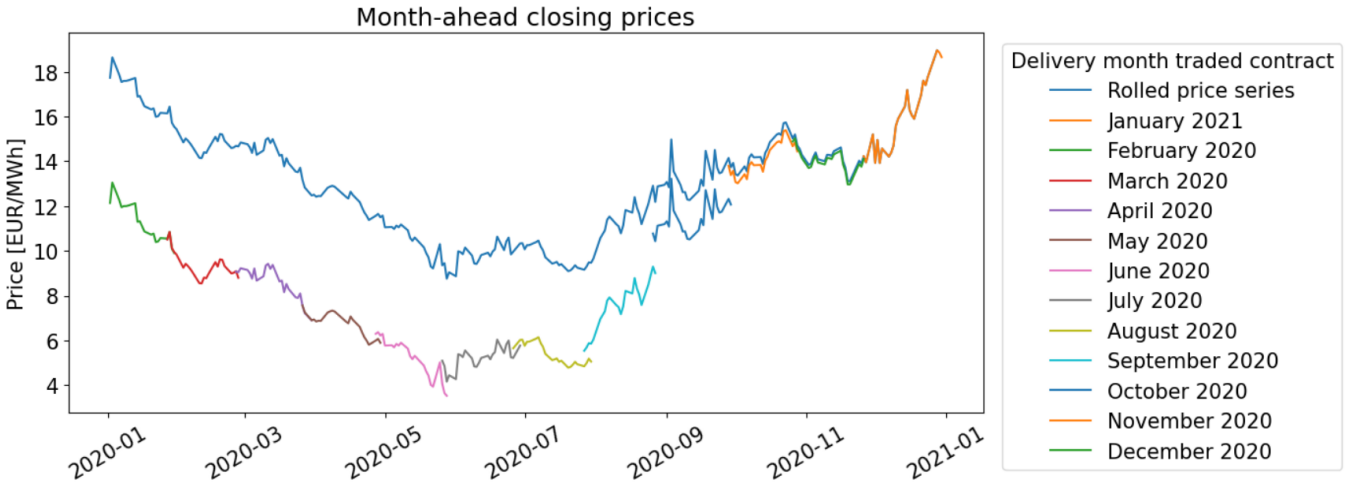


Figure 5: The non-rolled original contract prices vs. the Panama adjusted continuous price series.

Notice that the adjustment is not constant, but depends on the roll gap between the most recently adjust contract series and the contract series which is being adjusted at that moment. Notice also that the Rolled price series presented in fig. 5 is the daily VWAP calculated over the adjusted contract prices.

3.4 Data exploration

Besides the daily VWAP of the transaction data, it is interesting to look at the daily close prices. The close price is the price of the last transaction on the trading day. It does not consider the quantity of that transaction. Comparing it to the VWAP will show some of the differences in characteristics of the two pricing approaches. In fig. 6 we see that sometimes the close price is higher than the VWAP, and sometimes it is lower.

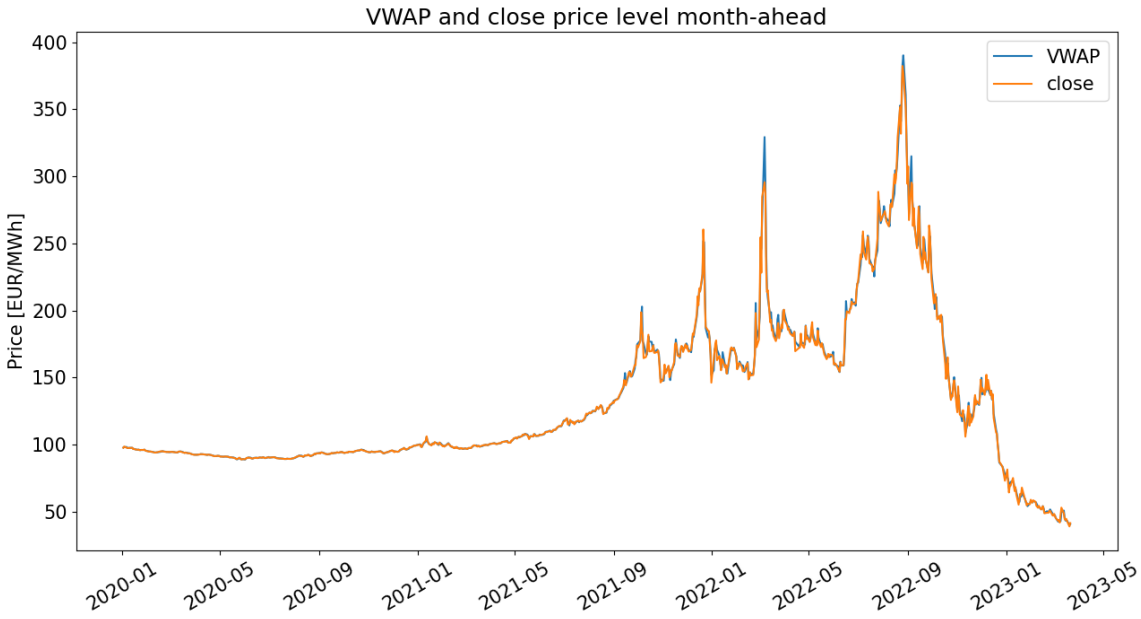
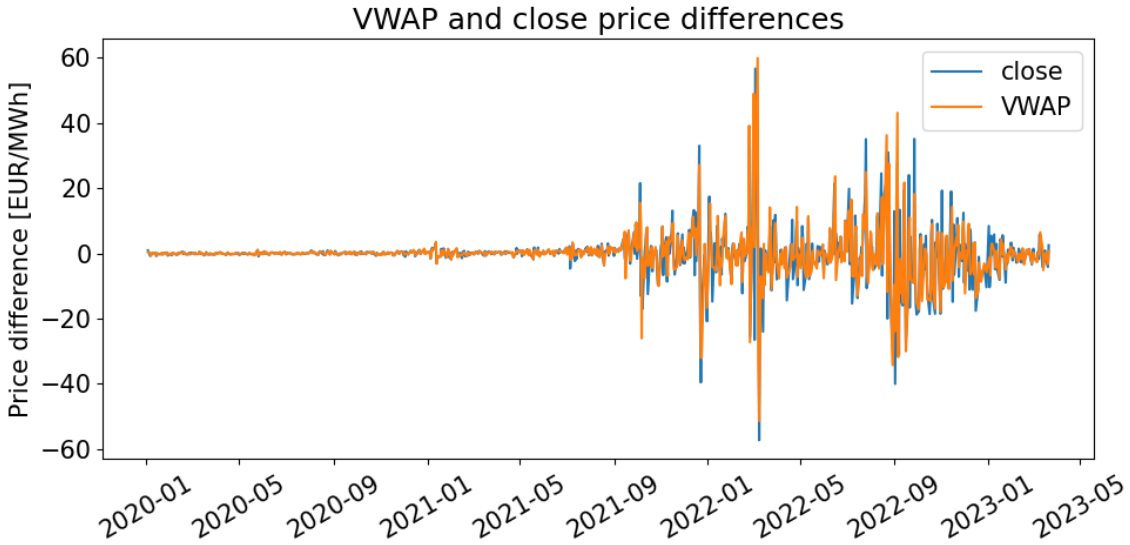


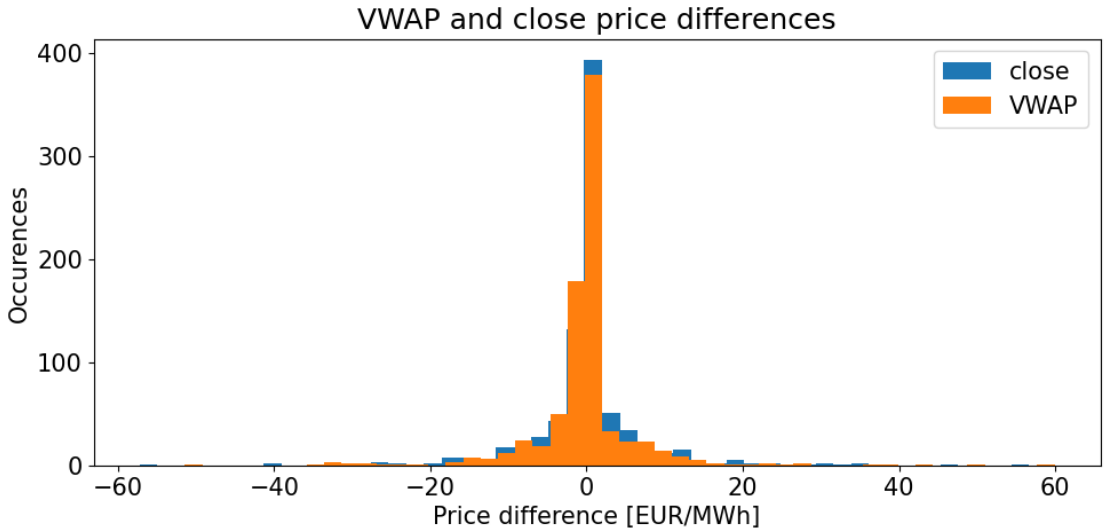
Figure 6: Price levels for both daily VWAP and close prices.

As we can see from fig. 6, the prices of month-ahead gas have been very volatile in the past couple of years. There seems to be two volatility regimes present in the data. A volatility regime is a period in which the volatility is either higher or lower than during a different period of time. The difference in volatility between the first and second period also becomes visible when looking at fig. 7a. This will be difficult to model, and may prevent us from achieving

results similar to those in other papers using older data. At the same time, it provides an opportunity to research the best ways to address these characteristics in the dataset.



(a) EOD and VWAP price differences between subsequent days.



(b) Distribution of price differences of subsequent days.

Figure 7: Distribution and levels of differences of VWAP and EOD subsequent prices

If we look at the differences between subsequent days in fig. 7a, the volatility of the prices clearly clusters. There is a period of low volatility in 2020 and part of 2021, until about September. After September 2021, volatility sharply increases, mainly due to the recovery of the economy after COVID-19 causing issues on the supply side. The peak in February 2022 has to do with the Russian-Ukrainian war, which had a big impact on the gas market.

If we look at the histogram of the price differences in fig. 7b, both distributions resemble some form of normal distribution, although they seem more centred around the mean. A couple of key metrics about the data are summarised in table 5. Some examples to take away from the table: the maximum daily VWAP in the dataset is equal to 149.38 euro; the maximum price difference between subsequent close prices is 56.65. The standard-deviation of the close prices is higher than the standard-deviation of the VWAP prices, which quantifies the fact that the close prices are more volatile. Also, we performed an Augmented Dickey-Fuller test, to test whether the series is stationary or not. Stationarity is very welcome when analysing a time series. The level series are not stationary, hence the high p-value (> 0.05). The high p-value indicates that we can not reject the null-hypothesis, which is the hypothesis that the time series is non-stationary. Taking the first difference of the price series is a common way to create a series which is stationary, of which the p-values for the difference series are a testimony.

Metric	VWAP level	VWAP differences	EOD level	EOD differences
mean	149.38	-0.09	149.08	-0.09
std	62.43	8.56	62.12	9.06
median	141.90	0.09	143.39	0.195
τ	–	0.96	–	1.07
min	39.95	-51.41	39	-57.25
max	390.22	59.89	382.27	56.65
1st quartile	101.49	-1.98	101.42	-2.50
3rd quartile	179.39	1.51	178.62	2.01
p-value ADF	0.54	≈ 0	0.56	≈ 0

Table 5: Indicative metrics VWAP price level and differences

Earlier, we denoted the kurtosis of the distribution shown in fig. 7b. In fig. 7b the distribution seems more concentrated around the mean than a standard normal distribution. This is further exemplified by fig. 8. Because of the large outliers the probability density function of both price series is leptokurtic. This means the distribution of the prices has fatter tails and is more centred around the mean than a standard normal distribution.

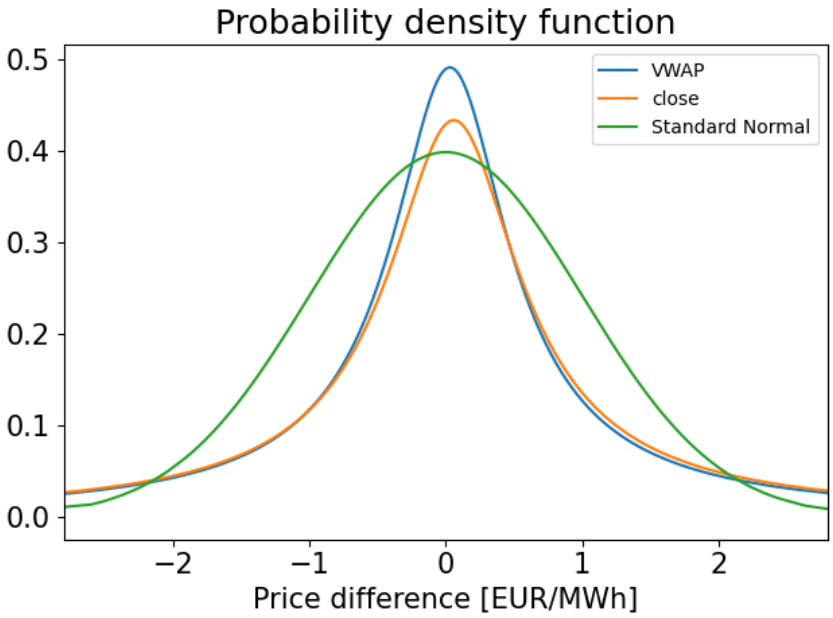


Figure 8: The fitted probability density functions of the VWAP and close prices, compared to a standard normal distribution.

If we look at the Mean Absolute Deviation (MAD) of the VWAP on a weekday basis, an interesting image appears. The MAD is defined in eq. (31).

$$\text{MAD} = \frac{1}{n}\sum_{i=1}^{n}|X_i - \mu|. \tag{31}$$

Where X_i are the samples in the dataset, n is the total number of samples in the dataset, and μ is the mean of the dataset. The MAD denotes the average absolute distance of the samples to the mean.

As we can see in fig. 9, the MAD is clearly the highest on Monday and Friday. The most probable explanation for this fact is that no trading occurs on weekends. During the weekend, information about the gas prices accumulate, and that translates to a correction on the market on Monday. Likewise, it might be true that traders tend to close positions before the weekend, causing fluctuations in the price on Friday.

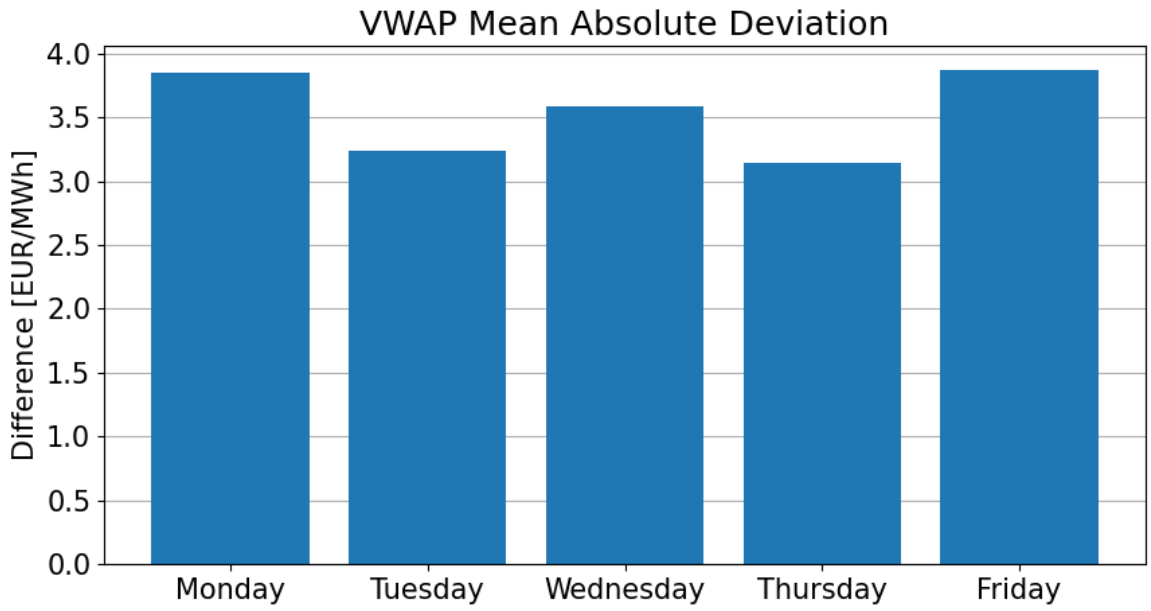


Figure 9: Mean Absolute Differences of the daily VWAP on a weekday basis.

3.5 Problem definition

Obviously, our problem definition is finding a model which makes the best gas price prediction given the dataset described in section 2. Part of the research is how to handle the characteristics of the dataset best. To quantify the performance of the model, we will be using the performance measures described in section 2.5. Defining the problem high-level, we are minimising the performance error between the forecast and the actual data:

$$\min_{M} \sum_{i=1,j<i}^{n} L(x_i - M(x_1, ..., x_j)), \tag{32}$$

where $\{x_1, ..., x_n\}$ denotes the dataset, the TTF month-ahead gas prices. M is the model employed to forecast the next price x_i using all (or part of) the previous data points: $\{x_1, ..., x_j\}, j < i$. $L(\cdot)$ is the performance measure which quantifies the prediction accuracy of the model M. We are comparing different variations on the model approach proposed by E et al. (2019). The reason to base our models on E et al. (2019) is that they achieve a good MAPE score. Besides this, E et al. (2019) use some advanced preprocessing and modelling techniques, which have favourable properties over other methods, something which will be discussed in section 3.6 and section 3.7.

A large part of our problem definition is how to address the differences in volatility which are present in the data set. As mentioned before, periods of high volatility occur, together with occasional outliers which up until that time, had never been seen before. These price dynamics might be inherent to the European market, which is why it is important to model them as accurately as possible.

3.6 Proposed data preprocessing techniques

3.6.1 Independent Component Analysis

Independent component analysis has been introduced by Herault and Jutten (1986), as a way to isolate N source signals from a collection of M sensor signals. Each sensor signal is an unknown linear combination of the N source signals. In the algorithm, the signals are defined as random vectors, allowing us to use probability and statistics to separate the sensor signals into independent components. The novel approach of this method is that no prior assumption about the distribution of the source signals is being made. After its inception, ICA proved to be very useful in many areas other than signal processing. It is very useful in the context of feature extraction for example, separating independent variables in a multivariate time series. Formally the signal mixing problem is defined as:

$$\mathbf{X} = \mathbf{AS}, \tag{33}$$

where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M]^\mathsf{T}$ is a matrix containing the M observed sensor signals, $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N]^\mathsf{T}$ is a matrix containing the N unknown independent source signals, and $\mathbf{A} \in \mathbb{R}^{M \times N}$ is a coefficient matrix describing the mixing of the source signals in the sensor signals. The goal of ICA is to find the matrix $\mathbf{S}$.

3.6.1.1 Non-Gaussianity and the Central Limit Theorem

ICA relies heavily on the Central Limit Theorem. The proof of the CLT is beyond the scope of this research. What is important to take away from this theorem is that any sum of non-Gaussian random variables is more Gaussian in distribution than the individual random variables themselves.

Definition 3.6.1 (Gaussian random variable)**.** *A Gaussian random variable, also called a Normal random variable, is a random variable with the Gaussian distribution as its probability density function:*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})} \tag{34}$$

Theorem 3.6.1 (Central Limit Theorem (CLT))**.** *Suppose $\{X_1, ..., X_n\}$ is a sequence of independent, identically distributed random variables with $\mathbb{E}[X_i] = \mu$ and $Var(X_i) = \sigma^2 < \infty$. Then, as n approaches infinity, the random variables $\sqrt{n}(\bar{X}_n - \mu)$ converge in distribution to a normal (Gaussian) variable $\mathcal{N}(0, \sigma^2)$:*

$$\sqrt{n}(\bar{X}_n - \mu) \to \mathcal{N}(0, \sigma^2). \tag{35}$$

As mentioned earlier, ICA does not make any assumptions about the prior distribution of the source signals. However, it does make some other assumptions. Before addressing these assumptions, we need to present a couple of definitions.

Definition 3.6.2 (Independent)**.** *Two random vectors $[\mathbf{x}_1, ..., \mathbf{x}_N]^T$ and $[\mathbf{y}_1, ..., \mathbf{y}_N]^T$ are independent if and only if their joint distribution is equal to the product of their marginal distributions:*

$$p(\mathbf{x}_1, ..., \mathbf{x}_N, \mathbf{y}_1, ..., \mathbf{y}_N) = p(\mathbf{x}_1, ..., \mathbf{x}_N)p(\mathbf{y}_1, ..., \mathbf{y}_N), \tag{36}$$

where p is some probability measure.

Definition 3.6.3 (White signal)**.** *A random vector is white if its components are uncorrelated and have variance 1.*

The two assumptions are presented below. Since assumption 3.6.1 can always be obtained by whitening and centring the data, we are not diving deeper into an explanation about it. For the proof that a whitened version of the data always exist we refer to Hyvärinen and Oja (2000). The sole purpose of centring is to simplify the FastICA algorithm. Centering and whitening are the minimal preprocessing procedures usually done, but depending on the situation, more preprocessing may be necessary.

Assumption 3.6.1. *The sensor signals are white and zero-mean.*

Assumption 3.6.2. *The source signals $\mathbf{s}_1, .., \mathbf{s}_N$ producing the mixed signals are independent and have a non-Gaussian distribution.*

Proof. For the sake of contradiction, assume $\mathbf{s}_1, .., \mathbf{s}_N$ have a Gaussian distribution, and assume the sensor signals $\mathbf{x}_1, ..., \mathbf{x}_M$ are Gaussian, uncorrelated and of unit variance. This is a reasonable assumption, since the input is assumed to be white. Because the Gaussian distribution is symmetric, the joint density of the sensor signals, $p(\mathbf{x}_1, ..., \mathbf{x}_1)$ will be symmetric as well. Thus, it would contain no information of the directions of the columns of the mixing matrix $\mathbf{A}$, which means $\mathbf{A}$ can not be estimated. □

To understand the intuition behind ICA consider a linear combination of the $\mathbf{x}_i$:

$$y = \mathbf{w}^\mathsf{T}\mathbf{x} = \sum_i w_i x_i, \tag{37}$$

where $\mathbf{w}$ is a weight vector to be determined. If

$$\mathbf{w} \in \text{colsp}(\mathbf{A}^{-1}), \tag{38}$$

then y would be equal to one of the independent components (source signals). Making a change of variables,

$$\mathbf{z} = \mathbf{A}^\mathsf{T}\mathbf{w}, \tag{39}$$

then

$$\begin{aligned} y &= \mathbf{w}^\mathsf{T}\mathbf{x} \\ &= \mathbf{w}^\mathsf{T}\mathbf{A}\mathbf{s} \\ &= \mathbf{z}^\mathsf{T}\mathbf{s}. \end{aligned} \tag{40}$$

y is a linear combination of the source signals. If we apply the central limit theorem, then y is more Gaussian than the individual source signals s_i of which y is made up of, since y is the sum of non-Gaussian distributions. Thus,

y will always be more Gaussian than the individual source signals, *except* if it is equal to one of the source signals, then it will be least Gaussian (since we assumed the source signals are non-Gaussian).

Picking $\mathbf{w}$ as the vector maximising the non-Gaussianity results in a $\mathbf{z}$ which has only one non-zero component. This means $\mathbf{w}^\mathsf{T}\mathbf{x} = \mathbf{z}^\mathsf{T}\mathbf{s}$ is one of the source signals. To find all of the independent components we need to find all of the local maxima. For an n-dimensional $\mathbf{w}$ this amounts to $2n$ local maxima, two for each independent component $-\mathbf{s}_i$ and $\mathbf{s}_i$ (the independent components can only be determined up to a multiplicative constant). To quantify the non-Gaussianity of the different signals, different ways exist.

A traditional way of measuring non-Gaussianity is by looking at the kurtosis of a distribution. Kurtosis is a measure of how often outliers occur, i.e. the 'fatness' of the tails of a distribution. However, this method is quite sensitive to outliers. That is why Hyvärinen and Oja (2000) propose a negentropy-based function to determine the "non-Gaussianity" of the signals. Negentropy is defined through the (differential) entropy H of a random vector $\mathbf{y}$.

Definition 3.6.4 (Differential entropy)**.** *Let* $\mathbf{y}$ *be a random vector with probability density function* f*, the differential entropy* H *is defined as:*

$$H(\mathbf{y}) = -\int f(\mathbf{y}) \log f(\mathbf{y}) d\mathbf{y}. \tag{41}$$

An important property of the Gaussian distribution is that among random variables with equal variance, the Gaussian distribution has the largest entropy. This means the Gaussian distribution is the most random of all random distributions. Thus, entropy can be used to measure Gaussianity, or rather "non-Gaussianity", since there is no distribution with higher entropy than the Gaussian distribution.

Definition 3.6.5 (Negentropy)**.** *Let* $\mathbf{y}_{gauss}$ *be a random Gaussian vector of the same covariance matrix as* $\mathbf{y}$*, then the negentropy* $J(\mathbf{y})$ *is defined as:*

$$J(\mathbf{y}) = H(\mathbf{y}_{gauss}) - H(\mathbf{y}), \tag{42}$$

Negentropy has the favourable properties that it is non-negative and that it is equal to zero if and only if $\mathbf{y}$ has a Gaussian distribution. Besides these properties it has very favourable statistical properties as well. A major disadvantage of negentropy is that it is computationally difficult to calculate. This is why Hyvärinen and Oja (2000) approximate the negentropy. The approximation is based on the maximum-entropy principle.

Principle 3.6.1 (Maximum-entropy principle)**.** *The probability distribution which best represents the current state of knowledge about a system is the one with largest entropy, in the context of precisely stated prior data.*

It states that the probability distribution with the largest entropy in terms of the observed data, is the best representation of the underlying probability distribution of the data. By applying this principle the following approximation for the negentropy is obtained:

$$J(\mathbf{y}) \approx \sum_{i=1}^{p} k_i [\mathbb{E}\{G_i(\mathbf{y})\} - \mathbb{E}\{G_i(v)\}]^2, \tag{43}$$

where k_i are some positive constants, $v \sim N(0,1)$, $\mathbf{y}$ is assumed to be of mean zero and unit variance, and G_i are non-quadratic functions. $J(\mathbf{y})$ is also called the contrast function. In E et al. (2019) they use two non-quadratic functions, causing the approximation to look as described in eq. (44).

$$J(\mathbf{y}) \approx k_1 [\mathbb{E}\{G_1(\mathbf{y})\}]^2 + k_2 [\mathbb{E}\{G_2(\mathbf{y})\} - \mathbb{E}\{G_2(v)\}]^2. \tag{44}$$

Hyvärinen and Oja (2000) found that some useful functions for G are:

$$\begin{aligned} G_1(u) &= \frac{1}{a_1} \log \cosh a_1 u \\ G_2(u) &= -\exp\left(\frac{-u^2}{2}\right) \end{aligned} \tag{45}$$

where $1 \leq a_1 \leq 2$ is a good parameter. E et al. (2019) use the following, commonly accepted, functions:

$$\begin{aligned} G_1(\mathbf{y}) &= \log \cosh(\mathbf{y}) \\ G_2(\mathbf{y}) &= -\exp(-\frac{\mathbf{y}^2}{2}). \end{aligned} \tag{46}$$

In case only one non-quadratic function is used, the expression for the negentropy is shown in eq. (47).

$$J(\mathbf{y}) \propto [\mathbb{E}\{G(\mathbf{y})\} - \mathbb{E}\{G(v)\}]^2. \tag{47}$$

3.6.1.2 FastICA

Now that we have established a way to measure non-Gaussianity we can introduce the optimisation algorithm, FastICA, to find the local maxima. We assume that the data is preprocessed by centering and whitening. To explain the FastICA algorithm, Hyvärinen and Oja (2000) first describe the process to find a single independent component (IC), after which they elaborate on the method to find multiple ICs. We shall follow the same order. To find a single IC, the FastICA uses a single computational 'unit', which can be seen as an artificial neuron. The unit has a weight vector $\mathbf{w}$ which can be updated by a learning rule (eq. (49)). The algorithm finds a direction, which is the $\mathbf{w}$ maximising the non-Gaussianity of $\mathbf{w}^\mathsf{T}\mathbf{x}$. The non-Gaussianity is measured by the approximation of negentropy $J(\mathbf{w}^\mathsf{T}\mathbf{x})$, defined in eq. (43) or eq. (44). To find the maximum non-Gaussianity of $\mathbf{w}^\mathsf{T}\mathbf{x}$, FastICA is based on a fixed-point iteration scheme, described in alg. 2.

Algorithm 2: Fixed point iteration scheme

Data: observed signals $\mathbf{x}$, initial weight vector $\mathbf{w}_0$

Result: weight vector $\mathbf{w}_{n+1}$ maximising non-Gaussianity of $\mathbf{w}^\mathsf{T}\mathbf{x}$

function `FixedPointIteration(`$\mathbf{x}, \mathbf{w}_0$`):`

$\mathbf{w}^+ \leftarrow \mathbf{w}_0$

$n \leftarrow 0$

while $\langle \mathbf{w}_{n+1}, \mathbf{w}_n \rangle \neq 1$ **do**

$$\mathbf{w}^+_{n+1} \leftarrow \mathbb{E}\{\mathbf{x} g(\mathbf{w}_n^\mathsf{T}\mathbf{x})\} - \mathbb{E}\{\mathbf{x} g'(\mathbf{w}_n^\mathsf{T}\mathbf{x})\}\mathbf{w}_n \tag{48}$$

$$\mathbf{w}_{n+1} = \frac{\mathbf{w}^+_{n+1}}{||\mathbf{w}^+_{n+1}||} \tag{49}$$

$$n \leftarrow n + 1$$

end

return $\mathbf{w}_{n+1}$

end function

Notice that g is the derivative of the non-quadratic function G used in eq. (47). We can set the convergence criterion equal to 1 since we assumed the data has been whitened. For the derivation of the method we refer to Hyvärinen and Oja (2000).

To calculate multiple ICs, the computational scheme described above has to be computed in parallel in several of the earlier described computational units, with weight vectors $\mathbf{w}_1, ..., \mathbf{w}_n$. To avoid the units from converging to the same independent components the intermediate results, the $\mathbf{w}_i^\mathsf{T}\mathbf{x}$'s, are decorrelated after every step. Three methods can be used to decorrelate the intermediate results (one of which is a modified Gram-Schmidt process). The simplest method is described in the following iterative algorithm:

Algorithm 3: FastICA to find n independent components

Data: observed signals $\mathbf{x}$, initial weight vectors $\mathbf{w}_0^0, ..., \mathbf{w}_0^n$

Result: matrix $\mathbf{W}$ maximising non-Gaussianity of $\mathbf{Wx}$

$\mathbf{W}_0 \leftarrow [\ \mathbf{w}_0^0 \quad ... \quad \mathbf{w}_0^n\ ]$

$$\mathbf{W}_0 = \frac{\mathbf{W}_0}{\sqrt{||\mathbf{W}_0\mathbf{W}_0^\mathsf{T}||}} \tag{50}$$

while $\mathbf{W}_j^\mathsf{T}\mathbf{W}_j \neq I$ **do**

  for $i = 0 : n$ **do**

    $\mathbf{W}_j[:, i] \leftarrow \textsc{FixedPointIteration}(\mathbf{x}, \mathbf{W}_{j-1}[:, i])$

  end

$$\mathbf{W}_{j+1} = \frac{3}{2}\mathbf{W}_j - \frac{1}{2}\mathbf{W}_j\mathbf{W}_j^\mathsf{T}\mathbf{W}_j \tag{51}$$

$$j \leftarrow j + 1$$

end

3.6.1.3 Convergence

Convergence of the FastICA algorithm has been exhaustively studied. Although publications of the convergence of a single computational cell dominate the literature, Oja and Zhijian Yuan (2006) investigate the convergence behaviour for the algorithm when multiple computational cells are working in parallel. With a generic cost function, they find a local quadratic convergence to the correct solution. However, when using the so-called kurtosis cost function, convergence can be locally increased up to cubic speeds. The kurtosis cost function is defined in Oja and Zhijian Yuan (2006) as:

$$J_G^{kurt} = \sum_{i=1}^{n} \mathbb{E}[(\mathbf{w}_i^\mathsf{T}\mathbf{z})^4]. \tag{52}$$

3.6.1.4 Advantages

There are several advantages of FastICA compared to traditional methods of calculating the ICA. We sum them up below:

1. FastICA has faster convergence than traditional methods: cubic in most cases, but at least quadratic, as opposed to linear in the traditional methods.
2. Most traditional methods are based on (stochastic) gradient descent, which can be difficult to tune. Since FastICA does not use a step size, no step size parameter has to be chosen, which makes it easy to use.
3. FastICA finds the independent components directly, without any estimates of their probability distributions a priori.
4. The algorithm can be tuned to have favourable properties such as robustness and minimum variance, by choosing the non-quadratic functions G optimally. The functions described in eq. (45) exhibit some optimal properties.
5. The independent components can be calculated one by one, so the process is explainable. Also, if there are only a certain number of ICs needed, FastICA can be chosen to stop after a certain number of components has been calculated.
6. FastICA exhibits most of the advantages of neural algorithms: it is parallel, distributed, computationally simple and requires little memory space.

3.6.2 (Ensemble) Empirical Mode Decomposition

Empirical mode decomposition has first been proposed by Huang et al. (1998), to decompose a time series signal into so-called intrinsic mode functions. The original signal can be obtained from this decomposition by summing up the intrinsic mode functions and adding back the residual left by the decomposition.

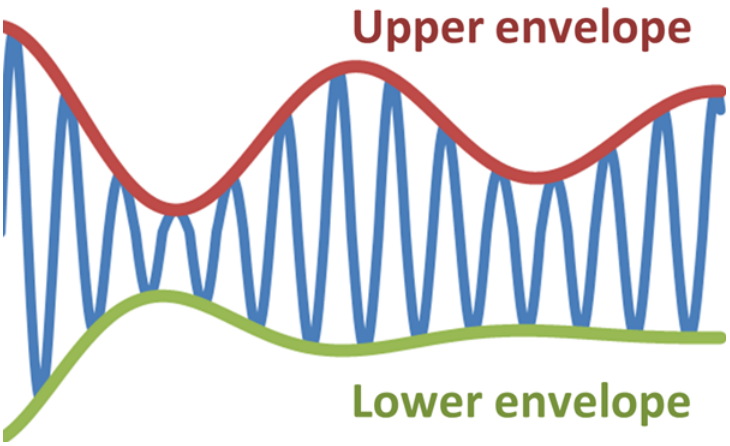
3.6.2.1 Intrinsic Mode Functions

A mode is a signal (i.e. time-series) whose number of local extrema and zero-crossings differ at most 1.

Definition 3.6.6 (Intrinsic Mode Function (IMF))**.** *The intrinsic mode function is an amplitude-modulated-frequency-modulated (AM-FM) signal, written as:*

$$u_k(t) = A_k(t)\cos(\phi_k(t)), \tag{53}$$

where $\phi_k(t)$ is called the phase, which is a non-decreasing function ($\phi_k'(t) \geq 0$). $A_k(t)$ is called the envelope, and it should be non-negative ($A_k(t) \geq 0$). The envelope of a signal is a smooth curve which outlines the extrema of the signal.



Source: Brews ohare - own work
Figure 10: Schematical representation of upper and lower envelopes of a time series

An important property of the IMF is that both the envelope $A_k(t)$ and the so-called instantaneous frequency, $\omega_k(t) := \phi_k'(t)$, vary much slower than the phase $\phi_k(t)$. A simple example of an IMF is a standard sine wave. EMD is similar to a Fast Fourier Transform (FFT) . A notable difference between EMD and FFT is that the decomposition produced by EMD remains in the time domain, whilst the FFT produces a decomposition in the frequency domain. Another difference is that the EMD is less restrictive on the signals that can be processed (EMD works well on non-linear and non-stationary time series as well). EMD is based on 3 assumptions:

1. The time series has at least two extrema - one maximum and one minimum.
2. The characteristic timescale is defined by the time lapse between the extrema.
3. If the data were totally devoid of extrema but contained only inflection points, then it can be differentiated once or more times to reveal the extrema.

The IMFs are determined by examining the characteristic timescales in the data.

Definition 3.6.7 (Characteristic timescale)**.** *Assume we are dealing with a dynamical system $\frac{d}{dt}X(t) = F(X)$, with an equilibrium $F(X_0) = 0$ and the system tends to restore the equilibrium exponentially:*

$$\lim_{t\to\infty} ||X(t) - X_0|| = \mathcal{O}(e^{-t/k}), \quad k > 0 \tag{54}$$

then the characteristic timescale of the system equals the supremum of k for which eq. (54) holds.

The characteristic timescale of a time series can be interpreted as the rate at which significant changes in the time series occur. There are multiple ways to calculate the characteristic timescale of a series, depending on the underlying process of the series. Huang et al. (1998) proposes to identify the different characteristic timescales by visual inspection of the series, which can be done in two ways. One is to look at the time lapse between the extrema, and the other is by looking at the time lapse between the successive zero-crossings. To systematically extract the characteristic timescales, a specific sifting process is proposed by Huang et al. (1998).

3.6.2.2 Calculating the envelopes

First, the extrema of the series are calculated, then the extrema are interpolated using their natural cubic spline line as the envelope of the series. Spline interpolation is a form of interpolation where the interpolating function (the spline) is piecewise defined by polynomials.

Definition 3.6.8 (Cubic spline)**.** *Given a set of samples* $C = [(x_0, y_0), ..., (x_n, y_n)]$ *the natural cubic spline between two of the samples is of the form* $S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$*, where* S_j *denotes the interpolated curve between* y_{j-1} *and* y_j*, with* $j = 1, ..., n$*. The spline satisfies the following equations:*

$$S_i(x_i) = y_i = S_{i-1}(x_i), \qquad i = 1, ..., n-1 \tag{55}$$
$$S_0(x_0) = y_0 \tag{56}$$
$$S_{n-1}(x_n) = y_n \tag{57}$$
$$S_i'(x_i) = S_{i-1}'(x_i), \qquad i = 1, ..., n-1 \tag{58}$$
$$S_i''(x_i) = S_{i-1}''(x_i), \qquad i = 1, ..., n-1 \tag{59}$$
$$S_0''(x_i) = S_{n-1}''(x_n) = 0, \qquad i = 1, ..., n-1 \tag{60}$$
$$\tag{61}$$

A number of algorithms exist to calculate the natural cubic spline of a set of data points. The cubic spline of the local minima forms the lower envelope, and the cubic spline of the local maxima forms the upper envelope.

3.6.2.3 Sifting

Next, the mean of the two envelopes is calculated, denoted as m_1. The difference between the data and m_1 is the first component of the decomposition, called h_1:

$$X(t) - m_1 = h_1. \tag{62}$$

In theory this procedure produces the first IMF in h_1, but in practice outliers usually cause the first round of sifting not to produce an IMF instantly. Repeated sifting on h_1 is needed until a true IMF is obtained. In the second sifting, h_1 becomes the data, and:

$$h_1 - m_{11} = h_{11}, \tag{63}$$

where m_{11} is the mean of the lower and upper envelopes of h_1. Repeating this procedure k times until a stopping criterion is satisfied we find that:

$$c_1 = h_{1k} \tag{64}$$

is the first IMF component of the data. The sifting process has two effects, it eliminates riding waves and it smooths out uneven (asymmetrical) amplitudes. Repeating the sifting process too often results in an IMF which carries no physical information, and which has constant amplitude. Therefore, Huang et al. (1998) propose a stopping criterion for the sifting process. They do this by limiting the size of the standard deviation, SD, computed from the two consecutive sifting results as:

$$\mathrm{SD} = \sum_{t=0}^{T} \left[\frac{|h_{1,(k-1)}(t) - h_{1,k}(t)|^2}{h_{1,(k-1)}^2(t)} \right]. \tag{65}$$

According to Huang et al. (1998) an appropriate value for SD as a stopping criterion is between 0.2 and 0.3, although they also mention that it is quite a rigorous limitation for the difference between siftings.

3.6.2.4 Reconstructing the original signal

The first IMF, c_1, contains the shortest period component of the series, with the highest frequency. After the first IMF c_1 is obtained, there is still longer period information left in the series. To obtain this information the sifting procedure is repeated on the residual r_1, which is defined as the difference between $X(t)$ and c_1:

$$X(t) - c_1 = r_1. \tag{66}$$

Repeating this procedure n times yields the n IMFs of the time series:

$$r_1 - c_2 = r_2, ..., r_{n-1} - c_n = r_n. \tag{67}$$

The procedure is written down in alg. 4 as well. Usually, the number of components is predefined by the user of the algorithm. However, Huang et al. (1998) also propose a stopping criterion to stop the generation of new components

whenever r_n or c_n becomes less than a predetermined value. A different stopping criterion they propose is to stop when the residue r_n is a monotonic function from which no IMF can be extracted. After the stopping criterion is satisfied, the original series $X(t)$ can be retrieved by summing the IMFs and the residual:

$$X(t) = \sum_{i=1}^{n} c_i + r_n \tag{68}$$

Algorithm 4: Empirical Mode Decomposition

Data: Observed signal X, n time-steps

Result: Matrix C containing K IMFs: $c_1, ..., c_K$

function `EMD(`X, K`):`

- $h_2 \leftarrow X$
- $SD \leftarrow 100$
- **for** $i = 1 : K$ **do**
  - **while** $SD < \varepsilon$ `/* according Huang et al. (1998)` ε `should be between 0.2 and 0.3 */` **do**
    - $r \leftarrow h_2$
    - $h_1 \leftarrow IMF(r)$
    - $h_2 \leftarrow IMF(h_1)$
    - SD $\leftarrow$ StandardDeviation(h_1, h_2)
  - **end**
  - $C[i] \leftarrow h_2$
  - $r \leftarrow IMF(h_2)$
- **end**
- **return** C

end function

function `IMF(`X`):`

- maxima, minima $\leftarrow$ Extrema(X)
- upperEnvelope, lowerEnvelope $\leftarrow$ CubicSpline(maxima, minima)
- $m_1 \leftarrow \frac{1}{2}$(upperEnvelope-lowerEnvelope)
- **return** $X - m_1$

end function

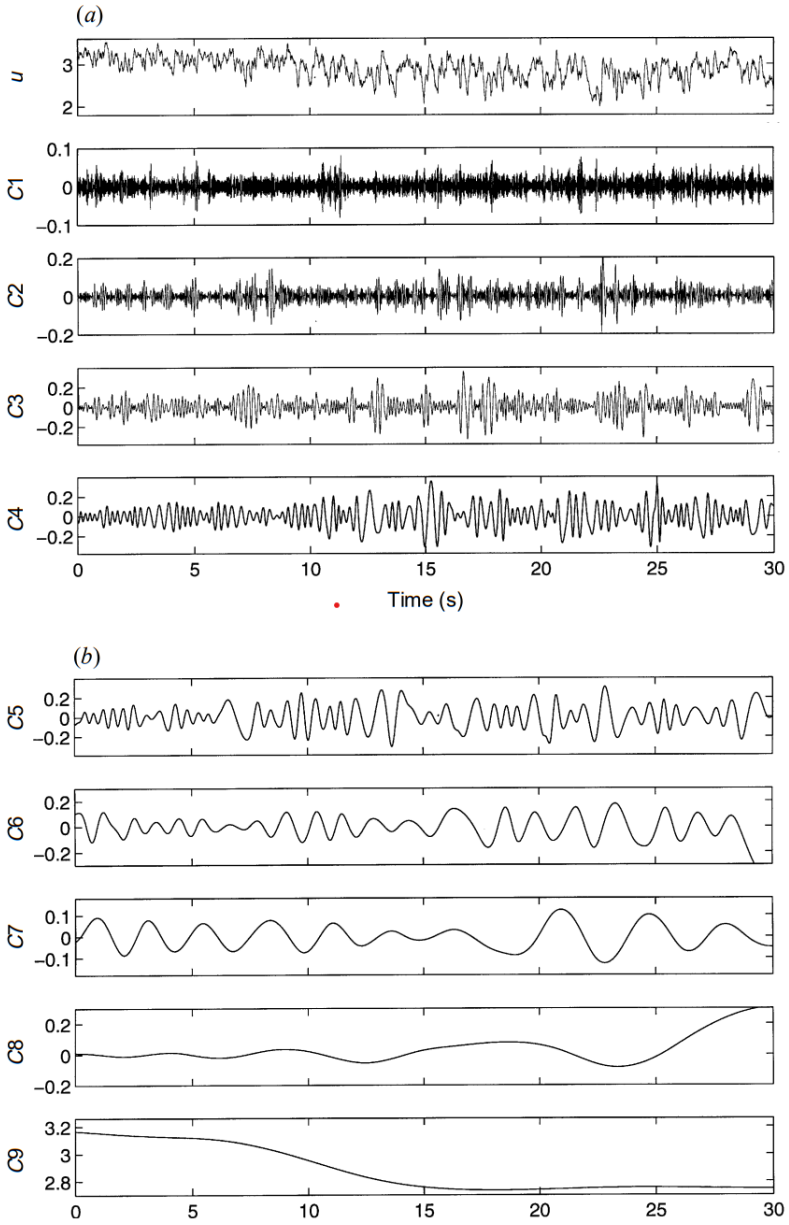
function `StandardDeviation(`h_1, h_2`):`

$$\text{SD} \leftarrow \sum_{t=0}^{T} \left[\frac{|h_{1,(k-1)}(t) - h_{1,k}(t)|^2}{h_{1,(k-1)}^2(t)} \right] \tag{69}$$

- **return** SD

end function

In alg. 4 the EMD procedure is shown. Extrema(X) denotes the function which extracts all extrama from the original signal X, and CubicSpline(X) does the same for the cubic spline. Detailed definitions of these functions are omitted for simplicity.



Source: Huang et al. (1998) *Figure 6*

Figure 11: Example EMD, taken from a series of wind speed data, with 9 components, (a): u denotes the original data, clearly C1 constitutes the shortest period component; (b): components 5-9, notice that component 9 is not an IMF, but rather the trend present in the original series

The example illustrated in fig. 11 shows the power of EMD. The original wind speed data is a series with a lot of extrema, but no zero-crossings. Since the process is non-stationary, it is hard to define a mean to take as a zero-reference to obtain zero-crossings. EMD eliminates this process by looking solely at the subsequent extrema via the envelopes. If we put the EMD technique in light of price forecasting, it might be interesting to look at the influence of exogenous variables on certain IMFs of the pricing series. Looking at fig. 11, pretending it is a pricing series, we might find that the periods where the amplitude is higher than usual in component C3 is correlated with higher volatility in a certain exogenous variable.

3.6.2.5 Ensemble EMD

EMD has a couple of notable shortcoming. One of the major drawbacks of the method is the frequent appearance of mode mixing, where a single IMF consists of signals with widely different scales, or a signal of similar scale being present in different IMF components. Mode mixing makes it harder to interpret the physical meaning of the IMFs. Huang et al. (1998) propose a solution to mode mixing by means of a so-called intermittence test. However, this method had its own problems. That is why in 2009, Wu and Huang (2009), propose the EEMD approach, to overcome the mode mixing difficulty, among some others, of the original method. The EEMD method sprung from

recent work on white noise, and its properties. EEMD relies heavily on the statistical properties of white noise.

The intuitive idea of EEMD is to add white noise to the original signal and perform EMD repeatedly on this new signal, each time using a newly generated white noise. The decompositions resulting from these signals with white noise are then combined into a single EMD result by taking the mean. The main assumption of the EEMD method is that white noise cancels out in a time-space ensemble mean. This means only the true signal can survive and persist in the final noise-added signal ensemble mean. Experiments in Wu and Huang (2009) prove that this enhanced version of EMD can indeed separate signals of different scales, without any occurrence of mode mixing. Besides this it has proven to utilise all statistical properties of white noise and visits all possible solutions in the finite neighbourhood of the true final answer.

3.6.3 Variational Mode Decomposition

Variational mode decomposition has been first described by Dragomiretskiy and Zosso (2014) as an alternative to EMD, which is sensitive to noise and sampling frequency. VMD is an entirely non-recursive method which extracts the IMFs concurrently. It looks for an ensemble (i.e. collection) of modes and their respective centre frequencies, such that the modes together form the original series. The result of VMD and EMD is the same, although the methods are very different. Before defining the optimisation problem around which VMD revolves, we first need to introduce the definition of the convolutional operator and the Dirac delta function.

Definition 3.6.9 (Convolutional operator)**.** *Suppose f and g are two integrable, real functions, then their convolution $(f * g)$ is defined as:*

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau) g(t - \tau) d\tau. \tag{70}$$

Definition 3.6.10 (Dirac delta function)**.** *The Dirac delta function is defined by its properties, and satisfies:*

$$\delta(t) = \begin{cases} 0, & x = 0 \\ +\infty, & x \neq 0, \end{cases} \tag{71}$$

and:

$$\int_{-\infty}^{\infty} \delta(t)\, dt = 1. \tag{72}$$

At the center of the VMD method lies the following optimisation problem:

$$\min_{\{u_k\},\{\omega_k\}} \left\{ \sum_k \left\| \partial_t \left[\left(\delta(t) + \tfrac{j}{\pi t}\right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \right\} \quad \text{s.t. } \textstyle\sum_k u_k = f, \tag{73}$$

where $\{u_k\}, \{\omega_k\}$ denote the set of modes and their centre frequencies respectively. $*$ denotes the convolutional operator, $\delta(t)$ denotes the Dirac delta function at time t, j is the imaginary unit ($j^2 = -1$) and $||\cdot||_2$ is the Euclidean distance. f denotes the original series, the constraint denotes the notion that the sum of the modes should equal the original series.

To solve eq. (73), Dragomiretskiy and Zosso (2014) propose to rewrite eq. (73) using a quadratic penalty term and Lagrangian multipliers, λ, in order to make the problem unconstrained. The use of a quadratic penalty term is a classic method to improve approximations of series where i.i.d. additive Gaussian noise is present. The proposed augmented Lagrangian $\mathcal{L}$ is presented in eq. (74).

$$\mathcal{L}\left(\{u_k\}, \{\omega_k\}, \lambda\right) := \alpha \sum_k \left\| \partial_t \left[\left(\delta(t) + \frac{j}{\pi t}\right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 + \left\| f(t) - \sum_k u_k(t) \right\|_2^2 + \left\langle \lambda(t), f(t) - \sum_k u_k(t) \right\rangle. \tag{74}$$

The solution to the original equation eq. (73) is a saddle point of the augmented Lagrangian $\mathcal{L}$. The saddle point is computed by performing a sequence of iterative sub-optimisations called the Alternate Direction Method of Multipliers (ADMM), which has a stopping criterion based on convergence. The ADMM framework is shown in alg. 5.

Algorithm 5: ADMM optimiser

Data: $\{u_k^1\}$, $\{\omega_k^1\}$, λ^1, $n \leftarrow 0$

Result: $\{u_k^n\}$, $\{\omega_k^n\}$

while $\sum_k \left\| u_k^{n+1} - u_k^n \right\|_2^2 / \left\| u_k^n \right\|_2^2 < \epsilon$ **do**

$n \leftarrow n + 1$

for $k = 1 : K$ **do**

$$u_k^{n+1} \leftarrow \arg\min_{u_k} \; \mathcal{L}(\{u_{i<k}^{n+1}\}, \{u_{i\geq k}^{n+1}\}, \{\omega_i^n\}, \lambda^n) \tag{75}$$

end

for $k = 1 : K$ **do**

$$\omega_k^{n+1} \leftarrow \arg\min_{\omega_k} \; \mathcal{L}(\{u_i^{n+1}\}, \{\omega_{i<k}^{n+1}\}, \{\omega_{i\geq k}^{n}\}, \lambda^n) \tag{76}$$

end

$$\lambda^{n+1} \leftarrow \lambda^n + \tau\left(f - \sum_k u_k^{n+1}\right) \tag{77}$$

end

3.6.3.1 Minimising the modes u_k^n

The minimisation with respect to u_k^n is done in a different way than with respect to ω_k^n. It makes sense to discuss the procedures separately. Starting with u_k^n (i.e. minimising eq. (75)), the problem is rewritten into the following equivalent expression:

$$u_k^{n+1} = \arg\min_{u_k \in X} \left\{ \alpha \left\| \partial_t \left[\left(\delta(t) + \frac{j}{\pi t} \right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 + \left\| f(t) - \sum_i u_i(t) + \frac{\lambda(t)}{2} \right\|_2^2 \right\}, \tag{78}$$

where α is a balancing parameter, and where we omit the indexing $\cdot^n$ and $\cdot^{n+1}$ for simplicity (each $u_{k\neq i}$ and ω_k is assumed to be the latest available one). Dragomiretskiy and Zosso (2014) go on to solve eq. (78) in the spectral domain by making use of the Parseval/Plancherel Fourier isometry under the L^2 norm:

$$\hat{u}_k^{n+1} = \arg\min_{\hat{u}_k, u_k \in X} \{ \alpha \left\| j\omega \left[(1 + \operatorname{sgn}(\omega + \omega_k)) \, \hat{u}_k(\omega + \omega_k) \right] \right\|_2^2 + \left\| \hat{f}(\omega) - \sum_i \hat{u}_i(\omega) + \frac{\hat{\lambda}(\omega)}{2} \right\|_2^2 \}, \tag{79}$$

they then apply a change of variables, $\omega \leftarrow \omega - \omega_k$, in the first term:

$$\hat{u}_k^{n+1} = \arg\min_{\hat{u}_k, u_k \in X} \{ \alpha \left\| j(\omega - \omega_k) \left[(1 + \operatorname{sgn}(\omega)) \, \hat{u}_k(\omega) \right] \right\|_2^2 + \left\| \hat{f}(\omega) - \sum_i \hat{u}_i(\omega) + \frac{\hat{\lambda}(\omega)}{2} \right\|_2^2 \}. \tag{80}$$

Exploiting the Hermitian symmetry of the real signals, they write both terms in eq. (80) as halfspace integrals over the non-negative frequencies:

$$\hat{u}_k^{n+1} = \arg\min_{\hat{u}_k, u_k \in X} \left\{ \int_0^\infty 4\alpha (\omega - \omega_k)^2 \left| \hat{u}_k(\omega) \right|^2 + 2 \left| \hat{f}(\omega) - \sum_i \hat{u}_i(\omega) + \frac{\hat{\lambda}(\omega)}{2} \right|^2 d\omega \right\} \tag{81}$$

The solution of eq. (81) is then found by letting the first variation vanish for the positive frequencies. The first variation of a function $J(y, h)$ is defined as:

$$\delta J(y, h) = \lim_{\varepsilon \to 0} \frac{J(y + \varepsilon h) - J(y)}{\varepsilon} = \frac{d}{d\varepsilon} J(y + \varepsilon h) \Big|_{\varepsilon = 0}. \tag{82}$$

This yields the solution to eq. (80):

$$\hat{u}_k^{n+1}(\omega) = \frac{\hat{f}(\omega) - \sum_{i \neq k} \hat{u}_i(\omega) + \frac{\hat{\lambda}(\omega)}{2}}{1 + 2\alpha (\omega - \omega_k)^2}. \tag{83}$$

The full spectrum of the real mode is then obtained by Hermitian symmetric completion. The mode in the time domain is obtained as the real part of the inverse Fourier transform of this filtered analytical series.

3.6.3.2 Minimising the frequencies ω_k^n

For the minimisation with respect to the frequencies ω_k^n, the problem is different from eq. (78), since the ω_k do not appear in the reconstruction fidelity terms $\|f(t) - \sum_k u_k(t)\|_2^2 + \langle \lambda(t), f(t) - \sum_k u_k(t) \rangle$, but only in the bandwidth prior $\alpha \sum_k \left\| \partial_t \left[\left(\delta(t) + \frac{j}{\pi t}\right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2$, as seen in eq. (74). Thus the relevant problem reads:

$$\omega_k^{n+1} = \arg\min_{\omega_k} \left\{ \left\| \partial_t \left[\left(\delta(t) + \frac{j}{\pi t}\right) * u_k(t) \right] e^{-j\omega_k t} \right\|_2^2 \right\}. \tag{84}$$

Again, they rewrite the problem to solve it in the Fourier domain:

$$\omega_k^{n+1} = \arg\min_{\omega_k} \left\{ \int_0^\infty (\omega - \omega_k)^2 \left|\hat{u}_k(\omega)\right|^2 d\omega \right\}. \tag{85}$$

The proposed solution to this quadratic problem is:

$$\omega_k^{n+1} = \frac{\int_0^\infty \omega |\hat{u}_k(\omega)|^2 d\omega}{\int_0^\infty |\hat{u}_k(\omega)|^2 d\omega} \tag{86}$$

Plugging both solutions (for the mode and for the frequencies) into alg. 5 we obtain the expression for the optimisation of the variational mode decomposition:

Algorithm 6: Complete ADMM optimiser for VMD

Data: $\{\hat{u}_k^1\}$, $\{\omega_k^1\}$, $\hat{\lambda}^1$, $n \leftarrow 0$

Result: $\{\hat{u}_k^n\}$, $\{\omega_k^n\}$

while $\sum_k \left\| \hat{u}_k^{n+1} - \hat{u}_k^n \right\|_2^2 / \left\| \hat{u}_k^n \right\|_2^2 < \epsilon$ **do**

$n \leftarrow n + 1$

for $k = 1 : K$ **do**

$$\hat{u}_k^{n+1}(\omega) \leftarrow \frac{\hat{f}(\omega) - \sum_{i<k} \hat{u}_i^{n+1}(\omega) - \sum_{i>k} \hat{u}_i^n(\omega) + \frac{\hat{\lambda}^n(\omega)}{2}}{1 + 2\alpha \left(\omega - \omega_k^n\right)^2}, \quad \text{for all } \omega \geq 0 \tag{87}$$

$$\omega_k^{n+1} \leftarrow \frac{\int_0^\infty \omega |\hat{u}_k^{n+1}(\omega)|^2 d\omega}{\int_0^\infty |\hat{u}_k^{n+1}(\omega)|^2 d\omega} \tag{88}$$

end

$$\hat{\lambda}^{n+1}(\omega) \leftarrow \hat{\lambda}^n(\omega) + \tau \left(\hat{f}(\omega) - \sum_k \hat{u}_k^{n+1}(\omega) \right), \quad \text{for all } \omega \geq 0 \tag{89}$$

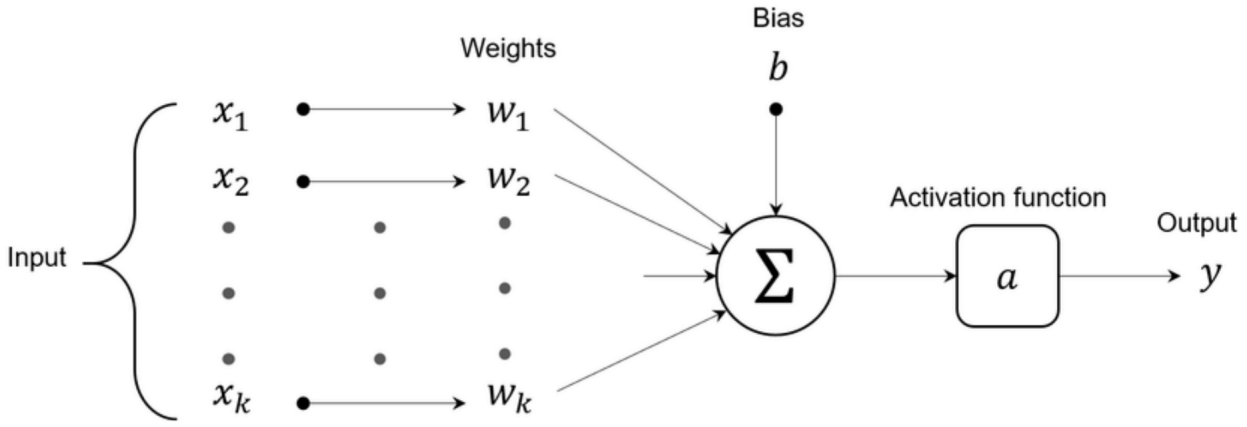
end

3.7 Proposed models

3.7.1 Artificial Neural Networks

A whole zoo of neural network architectures exists, of which fig. 14 gives a non-exhaustive summary. However, before diving into the types of neural networks used in this work, it is important to understand the general concept of a neural network. To explain the neural network in a general sense, we study the Deep Feed Forward ANN, of which the architecture is schematically shown in fig. 14. It consists of an input layer, two or more hidden layers, and an output layer. The layers are made up of computational cells, where special functions, the 'hidden' cells in fig. 14, handle the input and convert it to usable output.

All computational cells in the hidden layers are set up the same way. A schematic representation is given in fig. 12.



Source: Ataei et al. (2021)
Figure 12: Schematic representation of artificial neuron

Suppose the input is k dimensional: $\{x_1, ..., x_k\}$. Each cell takes in the outputs $\{y_1, ..., y_k\}$ produced by the previous k cells, or by the k input cells. It multiplies these inputs with weights, and sums it. A scalar bias is added, and the result is plugged into an activation function. Many different activation functions exist, but some of the most common are the Rectified Linear Unit (ReLU), sigmoid and hyperbolic tangent functions. The activation function controls the 'firing' of the neuron, just as in biological neurons, which either fire or don't.

3.7.1.1 Optimising performance

To optimise the performance of an ANN, three stages are usually defined: training, validation and testing. The dataset on which the ANN finds an optimal fit is divided into three sets: the training, validation and testing set. The performance of the model is measured using a loss function. Traditional loss functions such as the MSE and the RMSE are very common. The training set is used to determine which values should be assigned to the weights and biases of each cell. These values are assigned such that the loss function is minimised. The validation stage is commonly used to tune hyperparameters[5], or it is used to give a rough estimate of the performance on the test set. The purpose of the test set is to give an unbiased performance estimate of the model. It is very important not to use samples from the test set to build the model during training, since this would skew results.

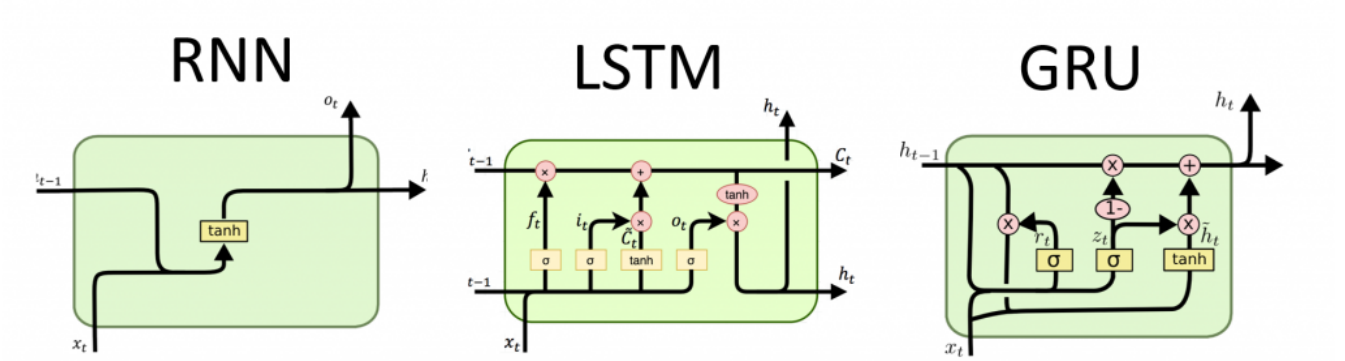
The ANN paradigm discussed above has been tried and tested over and over again. In the process, many spin-offs and adaptations have been published. The Gated Recurrent Unit Neural Network (GRUNN) used in this work is a special form of a Recurrent Neural Network (RNN). A key feature of the RNN is that it allows for cycles in the neural network architecture. This means that the output of some cells further on in the process can affect the inputs of cells earlier on in the process. This makes them suited for detecting temporal effects present in a dataset.

3.7.2 Gated Recurrent Unit Neural Network

Recurrent neural networks are widely used to make time series forecasts. However, some of the neural networks can be very hard to train, due to the occurrence of the so-called vanishing or exploding gradient problems. This is why Long Short Term Memory has been proposed by Hochreiter and Schmidhuber (1997), which overcomes these difficulties. To even further simplify the training, Cho et al. (2014) propose the GRUNN, which is based on LSTM, for analysing linguistic phrases.

Note that the RNN, LSTM and GRU actually have the architecture of a 'regular' feed-forward multi-layer perceptron, and that the difference lies in the hidden layers. In the hidden layers, the three models put to work different computational units, the 'computational cells'. These cells set the three methods apart from other neural networks. The structure of the cells employed in each of the three models are shown in more detail in fig. 13, by visualising their computational graphs.

[5]Hyperparameters are parameters set by the user, such as the number of layers or the activation function. The parameters of the model are the weights and biases inside the neurons, which are not determined by the user.



Source: DProgrammer - Daniel Lopez

Figure 13: The computation cells which set RNN, LSTM and GRUNN apart from the other neural network approaches.

For now we will focus on the GRU, for more information on LSTM we refer to Hochreiter and Schmidhuber (1997). The computational graphs for the gated memory cells are shown in fig. 13. The graph of the gated recurrent unit cell translates to a set of equations defining the inner workings of the cell. First, the so-called reset gate r_t is calculated:

$$r_t = \sigma\Big(W_r x_t + U_r h_{t-1}\Big), \tag{90}$$

where σ is the logistic sigmoid function, and x_t denotes the input vector (the gas price time series in our case). h_{t-1} is the previous hidden state, and W_r and U_r are weight matrices which will be 'learned' by training the neural network. The update gate z_t is defined exactly the same, except with different weight matrices:

$$z_t = \sigma\Big(W_z x_t + U_z h_{t-1}\Big). \tag{91}$$

The computation of h_t is defined as:

$$h_t = z_t \odot \tilde{h}_t + (1 - z_t) \odot h_{t-1}, \tag{92}$$

where:

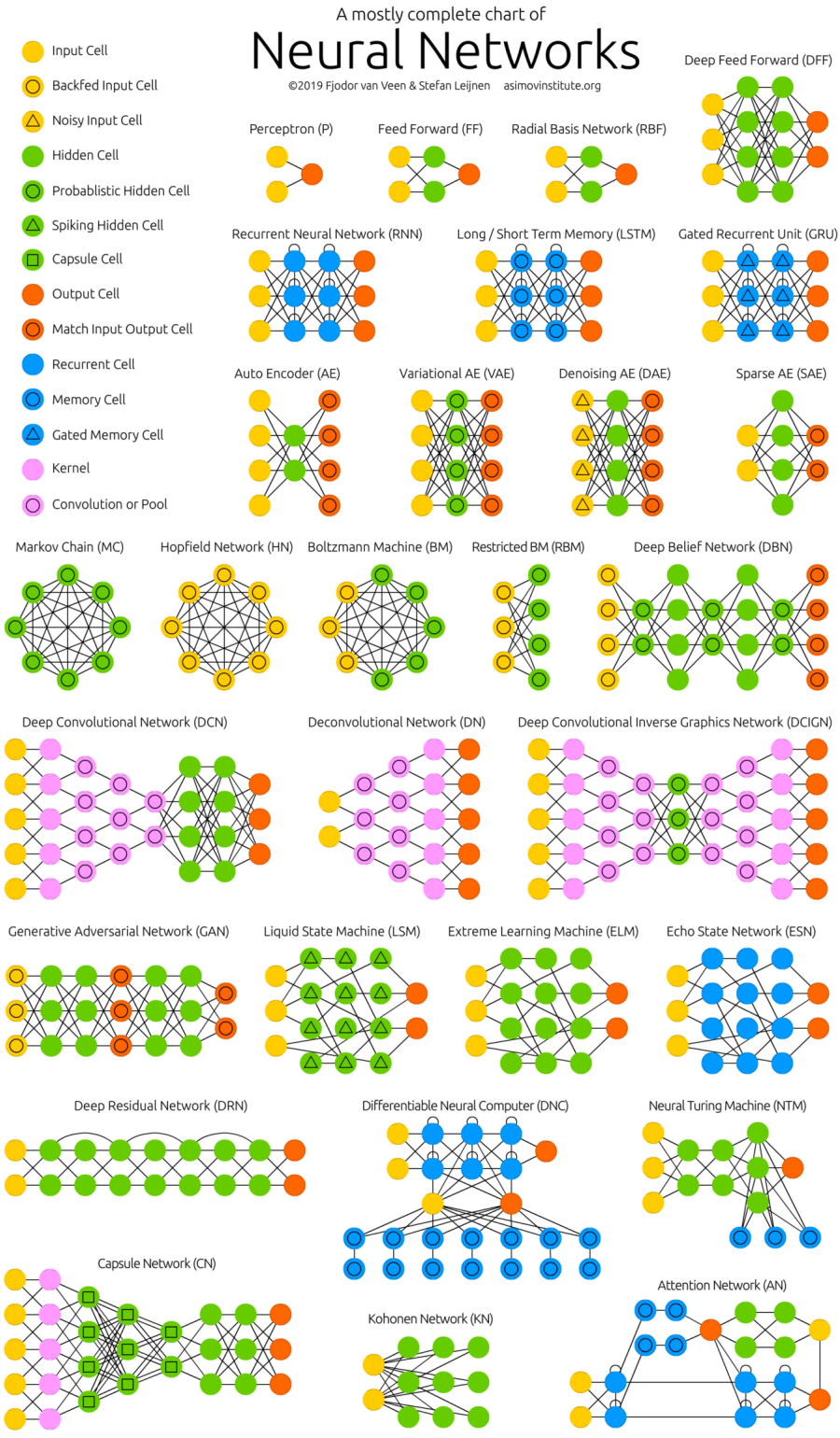
$$\tilde{h}_t = \phi\Big(W_h x_t + U_h(r_t \odot h_{t-1})\Big), \tag{93}$$

$\odot$ denotes element-wise multiplication between two vectors or matrices, it is also called the Hadamard product. ϕ is an activation function, and often the hyperbolic tangent is used, $\phi(x) = \tanh(x)$, as in fig. 13. In essence, we see that the final output state h_t defined in eq. (92) is just a linear combination of the past state and the current state, weighted by z_t. In this setup for the gated memory cells, if the reset gate r_t is close to 0, the hidden state ignores the previous hidden state and reset with the current input. This is a more compact representation than the memory cells in the LSTM framework, since the hidden state may drop any past information. The update gate z_t controls how much information from past states will be carried on to the next state, it helps the GRUNN 'remember' long-term trends, just as in an LSTM-network. Since all of the cells have their separate update and reset gates, each cell will model trends on different timescales. The cells modelling the short-term trends will forget past values quicker, whereas the cells focusing on the long term trends will have less active reset gates.

3.7.3 Support Vector Regression

Support vector regression is largely based on support vector machines. SVMs maximise linear separability of two groups of values by using the kernel trick. The kernel trick is the process of mapping values onto a higher-order space. The trick is that even if the values are not separable in their native space, they might be in a higher dimension space after the mapping. When performing SVR, the kernel trick translates to non-linearly transforming the input space into a feature space. In SVR, multivariate linear regression is performed on the transformed data. SVR was first proposed by Cortes and Vapnik (1995), and further described by Drucker et al. (1996).

Before describing the minimisation problem associated with SVR, we first need to address a notable difference between the regression in SVR and traditional (multivariate) regression. The difference is the loss function used to assess the fit of the regression. Instead of an ordinary least squares loss function for example, Cortes and Vapnik (1995), propose the so-called ε-insensitive loss function.



Source: The Asimov Institute - The Neural Network Zoo

Figure 14: A non-exhaustive overview of some of the most common neural network architectures.

Definition 3.7.1 (ε-insensitive loss function)**.** *The ε-insensitive loss function is defined as:*

$$L(y, f(x, \alpha)) = L(|y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})|_\varepsilon), \tag{94}$$

where:

$$|y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})|_\varepsilon = \begin{cases} 0, & \text{if } |y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})| \leq \varepsilon \\ |y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})| - \varepsilon, & \text{otherwise,} \end{cases} \tag{95}$$

with y_i a sample of the data set, $\mathbf{x}_i$ the predictor variable and $f(\mathbf{x}_i, \hat{\mathbf{w}})$ the regression function, with weights $\hat{\mathbf{w}}$.

Cortes and Vapnik (1995) define three types of ε-sensitive loss functions:

1. The linear ε-insensitive loss function

$$L(y_i, f(\mathbf{x}_i, \hat{\mathbf{w}})) = |y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})|_\varepsilon \tag{96}$$

2. The quadratic ε-loss function

$$L(y_i, f(\mathbf{x}_i, \hat{\mathbf{w}})) = |y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})|_\varepsilon^2 \tag{97}$$

3. The Huber loss function

$$L(y_i, f(\mathbf{x}_i, \hat{\mathbf{w}})) = \begin{cases} c|y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})| - \frac{c^2}{2} & \text{for } |y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})| > c \\ \frac{1}{2}|y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})|^2 & \text{for } |y_i - f(\mathbf{x}_i, \hat{\mathbf{w}})| \leq c. \end{cases} \tag{98}$$

Any convex loss function may be used, but Cortes and Vapnik (1995) choose these loss functions specifically because they lead to a simple optimisation task used in pattern recognition as well. The intuition of the linear ε-sensitive loss function is that the loss equals zero if the predicted value is inside a tube with radius ε. If the prediction is outside of the tube, the loss equals the magnitude of the difference between the predicted value and the radius ε of the tube. The tube with radius ε has the decision surface as centre. In general, this leads to the following unconstrained quadratic optimisation problem:

$$\min_{\mathbf{w}} \quad U\sum_{j=1}^{N} L(y_j - f(\mathbf{v}_j, \mathbf{w})) + ||\mathbf{w}||^2, \tag{99}$$

which is dependent on the samples y_j, the corresponding feature-vector $\mathbf{v}_j$, and the weight vector $\mathbf{w}$, which will be approximated by $\hat{\mathbf{w}}$. U is called the regularisation constant. U determines whether emphasis is laid on the error, if it is large. If U is small, less emphasis is laid on the error, which leads to better generalisation.

In detail, the ε-insensitive loss function leads to the following quadratic optimisation problem:

$$\begin{aligned} \min \quad & U(\sum_{i=1}^{N}\xi_i^* + \sum_{i=1}^{N}\xi_i) + \frac{1}{2}||\mathbf{w}||^2 \\ \text{s.t.} \quad & y_i - (\mathbf{w}^\mathsf{T}\mathbf{v}_i) - b \leq \varepsilon + \xi_i \\ & (\mathbf{w}^\mathsf{T}\mathbf{v}_i) + b - y_i \leq \varepsilon + \xi_i^* \\ & \xi_i^*, \xi_i \geq 0, \end{aligned} \tag{100}$$

where b represents the bias. ξ_i and ξ_i^* represent the values of the ε-insensitive loss function. They are zero if the sample point is inside the tube. If the observed sample is 'above' the tube, ξ_i is the positive difference between the observed value and ε. Conversely, if the observed value is 'below' the tube, ξ_i^* is the non-negative difference between the observed value and ε. Either ξ_i or ξ_i^* will be zero, since a sample can not be above and below the tube simultaneously. If the sample is inside the tube, both ξ_i and ξ_i^* will be zero.

To solve eq. (100), Drucker et al. (1996) propose to find a saddle point by differentiating with respect to $\mathbf{w}, b$ and ξ_i. This leads to the following dual quadratic optimisation problem:

$$\begin{aligned} \max_{\alpha_i, \alpha_i^*} \quad & -\varepsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) + \sum_{i=1}^{N} y_i(\alpha_i^* + \alpha_i) - \frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)(\mathbf{v}_i^\mathsf{T}\mathbf{v}_j + 1)^p \\ \text{s.t.} \quad & \sum_{i=1}^{N}\alpha_i^* = \sum_{i=1}^{N}\alpha_i \\ & 0 \leq \alpha_i \leq U, \quad i = 1, ..., N \\ & 0 \leq \alpha_i^* \leq U, \quad i = 1, ..., N. \end{aligned} \tag{101}$$

Writing eq. (101) in standard form and solving it using an active set method, Drucker et al. (1996) describe the following expression for the support vector regression representation, developed by Cortes and Vapnik (1995):

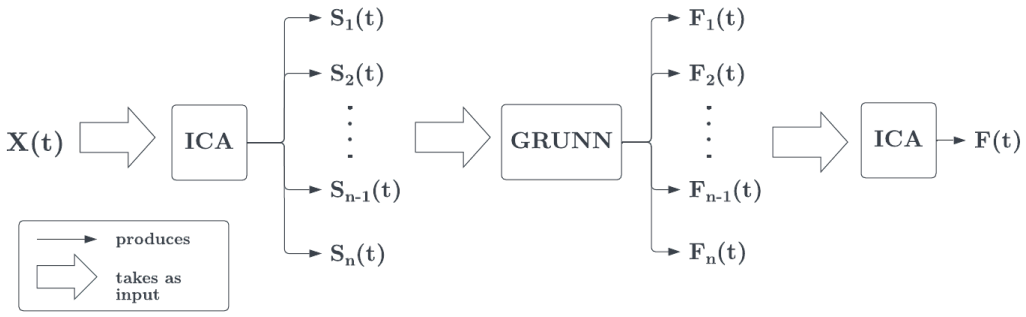
$$f(\mathbf{x}, \hat{\mathbf{w}}) = \sum_{i=1}^{N} (\alpha_i^* - \alpha_i)(\mathbf{v}_i^\mathsf{T}\mathbf{x} + 1)^p + b. \tag{102}$$

In general, for a p-th order polynomial as described in eq. (102), and a d-dimensional input space, the feature dimensionality df of $\hat{\mathbf{w}}$ equals:

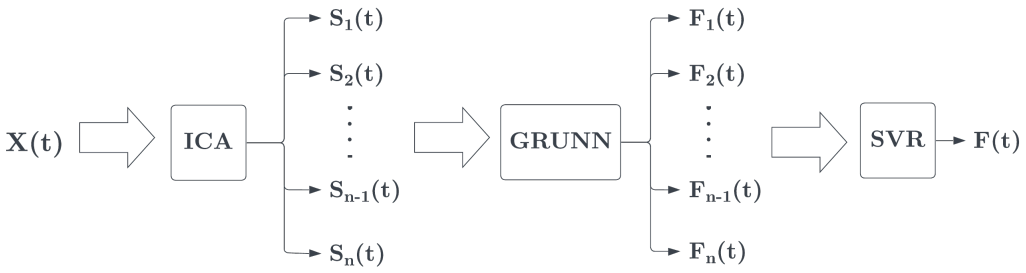
$$df = \sum_{i=d-1}^{p+d-1} C_{d-1}^i, \quad \text{where} \quad C_k^n = \frac{n!}{k!(n-k)!}. \tag{103}$$

3.7.4 Overview of methods

A high-level overview of the forecasting process combining all of the previously mentioned methods is shown in fig. 15b. Besides this approach proposed by E et al. (2019), we look at a variation of this approach where we do not use SVR for integration of the component forecasts made by GRUNN, but rather mix the GRUNN forecasts directly to arrive at a single price forecast. This approach is shown in fig. 15a.



(a) Data flow when SVR is not used



(b) Data flow when using SVR

Figure 15: Difference between SVR and non-SVR approach.

$X(t)$ denotes the input price series. With the use of VMD and ICA it will be split into multiple independent components, $S_i(t)$, which will be easier to forecast because of the reduced complexity per component. The forecasts $F_i(t)$ for the individual components are produced by the GRUNN, which will be discussed in section 3.7. Finally, these are combined by using one of two methods: either by directly mixing the forecasts using the original mixing matrix produced by ICA, or by using SVR to arrive at a single price forecast $F(t)$.

3.8 Research questions

The questions associated with the process of producing a forecasting model for the dataset are presented below. They are ordered according to the order in which they will be researched and answered during the research.

1. How do the EU TTF gas prices from 2020-2023 compare to the data used in other works?

2. How does the ICA-VMD-GRUNN-SVR model perform on daily month-ahead European TTF prices?
   (a) What is the best way to address the different volatility regimes which seem to be present in the dataset?
      i. What are the characteristics of the volatility regimes we are trying to define?
      ii. Is it possible to recognise these volatility regimes using preprocessing techniques?
   (b) Which parameters are optimal for the performance?
   (c) How does the model perform against a selection of benchmark models?
   (d) What performance measures are most appropriate?
      i. Which and how many performance measures should we use for the final model?
      ii. How does the preprocessing of the data skew the performance results?
   (e) How are we going to address the non-deterministic nature of the FastICA method when training the GRUNN model?
3. Does the inclusion of exogenous variables improve the forecast of the ICA-VMD-GRUNN-SVR approach?
   (a) Which exogenous variables (if any) improve forecasting the most, and why?
   (b) For which regimes does the inclusion of exogenous variables improve the forecasting the most, high or low volatility?

Scope of future research could be to build a hybrid-quantile forecasting model using the previously described ICA-VMD-GRUNN-SVR framework of E et al. (2019). Quantile SVR and quantile LSTMs are already used, and described in Xie and Wen (2019) and Yu et al. (2018). Combining these would be a novel approach to gas price prediction, all the while providing a more informative prediction tool for the traders at Northpool. Extending the model to forecast intraday (i.e. hourly) prices, different products, or multiple days ahead are different subjects for further research.

4 Results

Before diving into the forecasting results, we first present the results of the preprocessing methods, after which we discuss some parameters influencing the performance of the model. We discuss the performance measures used for training in section 4.3.2.

4.1 Comparison of datasets

As mentioned earlier, the dataset used in this work is very different from the datasets used in other works. In table 6 some key metrics from our dataset are compared to metrics from datasets presented in other works, and a clear difference emerges.

	current dataset	Berrisch and Ziel (2022)	Salehnia et al. (2013)	Su et al. (2019b)
gas hub	EU TTF	EU TTF	US Henry Hub	US Henry Hub
sample frequency	daily	daily	daily	monthly
type of contract	month-ahead	month-ahead	day-ahead	day-ahead
period	2020-2023	2011-2020	1997-2012	2001-2008
mean	149.08	20.25	4.854	4.706
std	62.12	4.92	2.426	2.215
min	39	8.53	1.050	1.73
max	382.27	29.35	18.48	13.42
1st Q	101.42	16.25	-	-
3rd Q	178.62	24.25	-	-

Table 6: Comparison of datasets used in some papers.

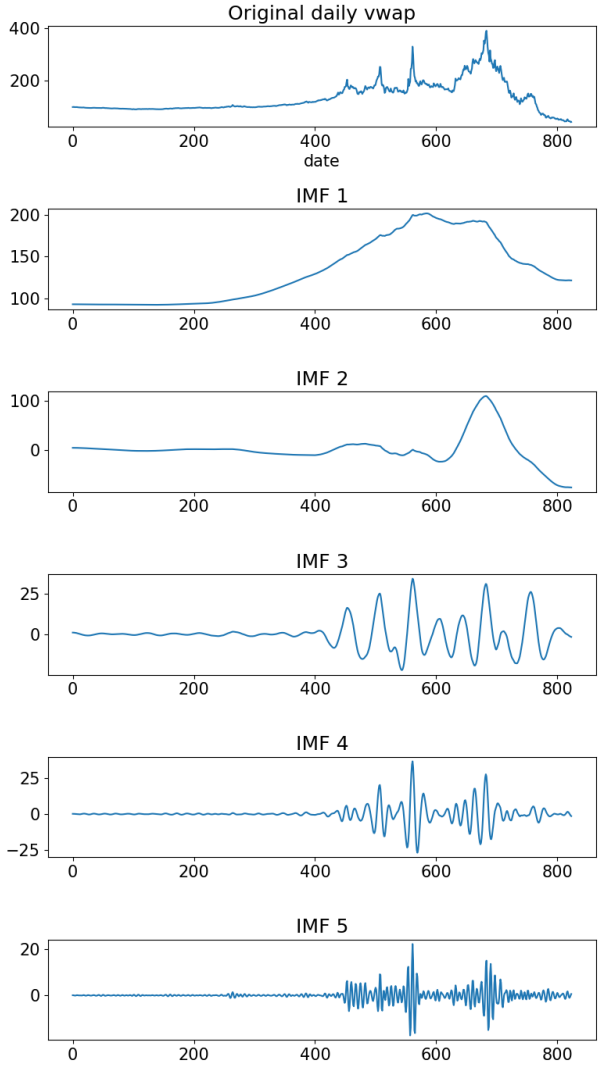
The main difference in table 6 is the extraordinary high standard deviation our dataset exhibits. This is visible when looking at the price levels in fig. 6. Besides the high standard deviation, what stands out is that US gas prices are generally lower than European gas prices. This might have to do something with the fact that the EU has to import its gas from elsewhere, whereas the US can produce low cost (shale) gas itself.[6]

[6] RBN Energy: The Space Between - European And U.S. Gas Markets Look To Each Other For Price Direction

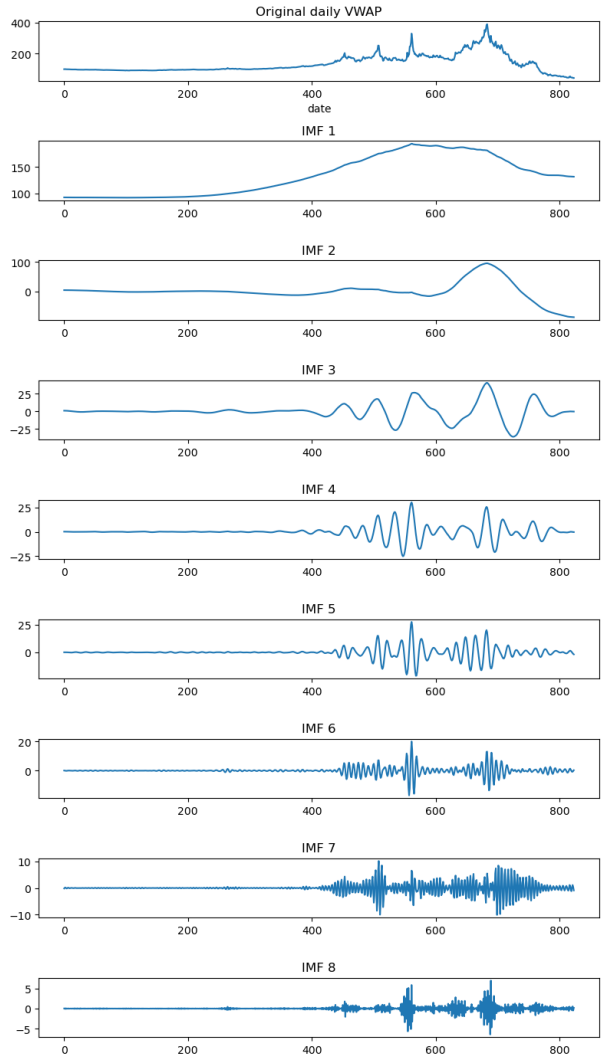
In this case however, it probably has more to do with the different time periods the data has been recorded. One of the consequences of the difference in price between the EU and US is that performance measures such as the MSE, RMSE and MAE are practically incomparable across publications. Even compared to the dataset used in Berrisch and Ziel (2022), which also looks at EU TTF gas prices, big differences are present in terms of almost all metrics. Due to structural changes in the market price dynamics during 2020-2023, our dataset has a mean which is circa seven times higher than the dataset in Berrisch and Ziel (2022), with a standard deviation which is approximately 12 times as high. This makes the period 2020-2023 very interesting for analysis.

4.2 Preprocessing: VMD and ICA

Let's start by looking at the dataset as a whole, as depicted in fig. 6, and calculate the VMD for this time series.



(a) A variational mode decomposition with 5 components.



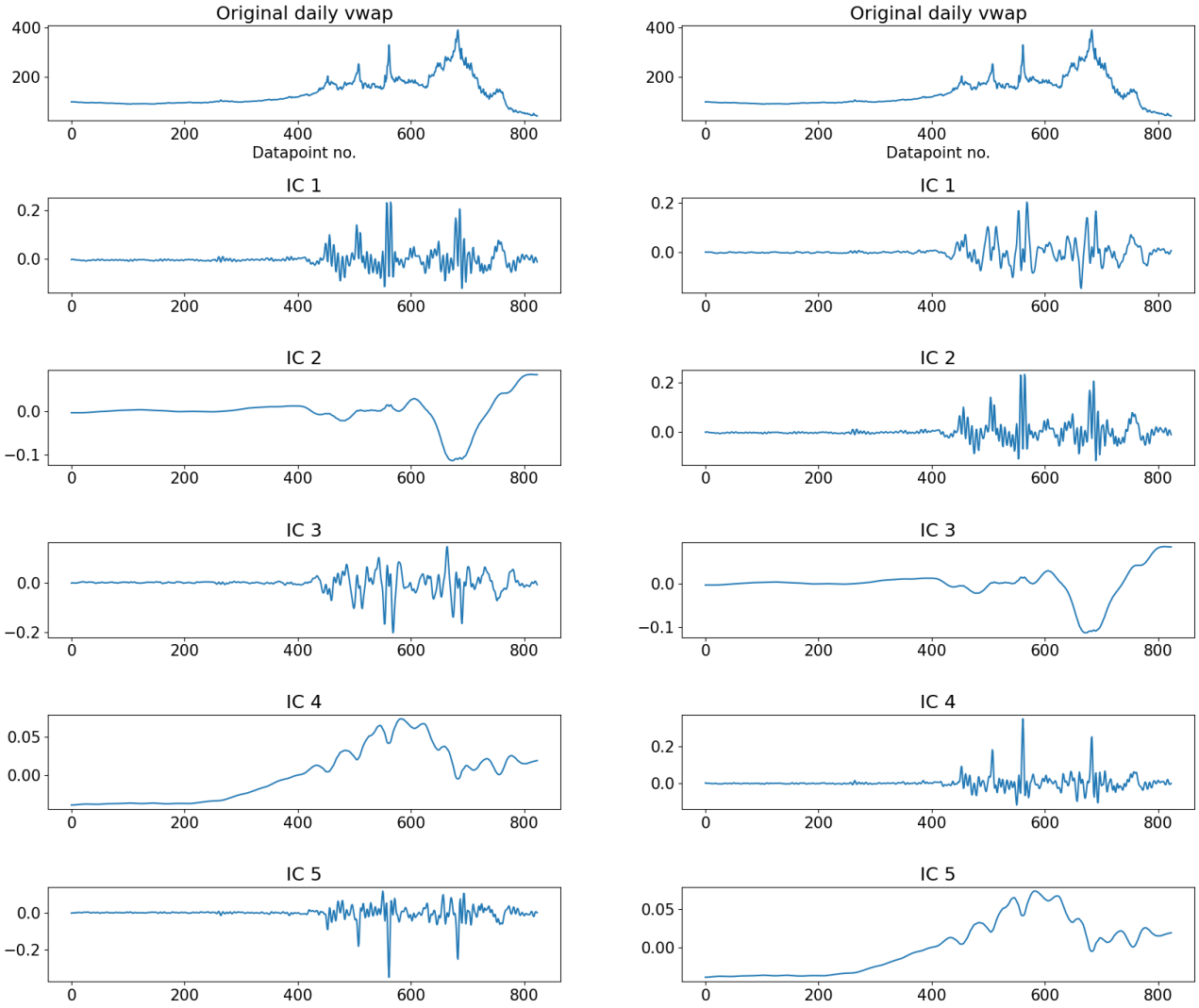
(b) A variational mode decomposition with 8 components.

Figure 16: Variational Mode Decomposition with 5 and with 8 components.

The presumed volatility regimes become more pronounced when looking at the decompositions shown in fig. 16. In the high frequency components 3, 4 and 5 shown in fig. 16a we see an increase in amplitude starting somewhere around sample number 400. This probably indicates elevated levels of volatility in the original TTF price. Note that for convenience and ease of comparison we will only depict the sample numbers when dealing with decompositions, instead of the corresponding dates.

Notice that the components are neatly ordered based on their frequency. At the top we see the low frequency, slow components, and at the bottom the faster and more high frequency components. This changes when we apply ICA to the VMD, since ICA does not produce an ordered component output. As we can see from fig. 17, the high frequency and low frequency components are randomly ordered. However, some of the components are very similar (such as IC 1 in fig. 17a and IC 2 in fig. 17b; or IC 4 in fig. 17a and IC 5 in fig. 17b).

When comparing two ICA decompositions stemming from the same dataset we see some components that are very similar, and some that are each others negative counterpart. This difference in decompositions is going to be a problem when we want to train models on a certain dataset, and reuse those trained models on a dataset with similar characteristics but a (slightly) different decomposition. Since the model has been trained on one ICA decomposition, it will be tailored to that specific order of independent components. When reapplying ICA on a different dataset, or even the same dataset, the resulting decomposition might be substantially different, rendering the trained model useless.



(a) An ICA decomposition for our dataset.

(b) Another ICA decomposition for our dataset.

Figure 17: Two ICA decompositions derived from exactly the same dataset.

4.2.1 Ordering and synchronisation

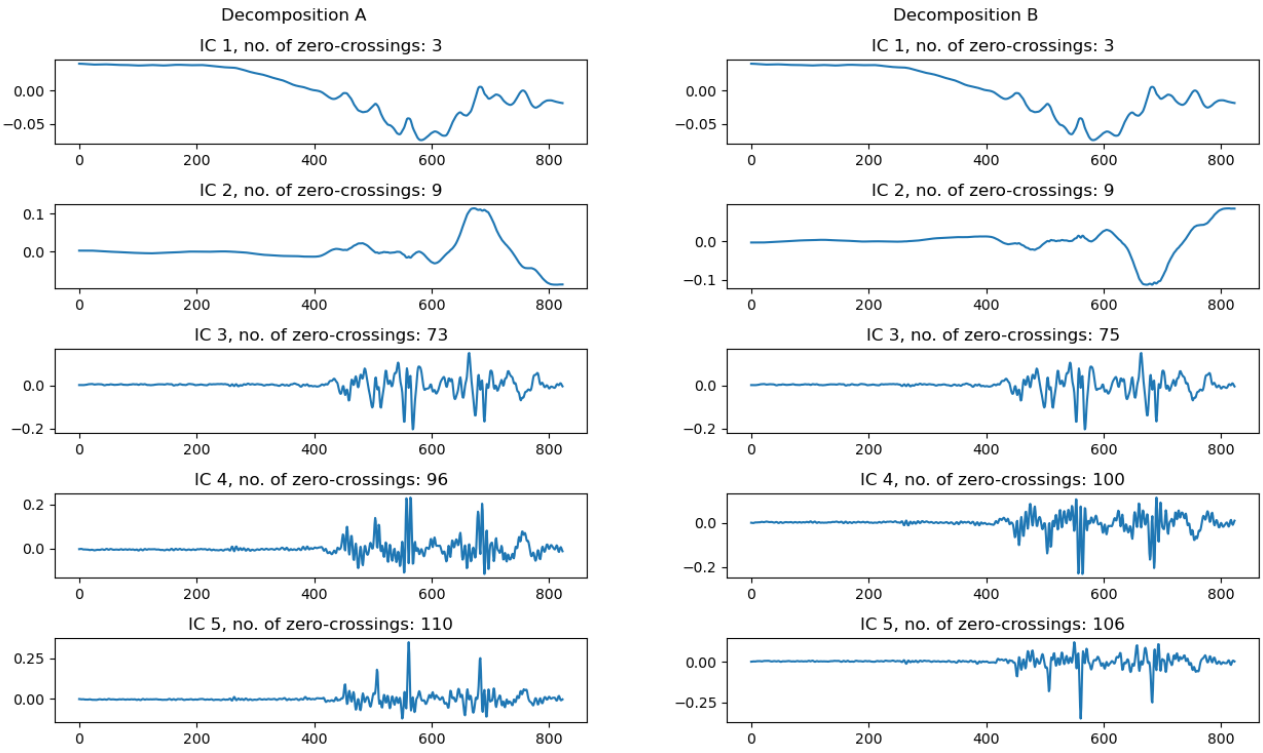
To be able to reuse a previously trained model on a new dataset we need to do two things to the ICA decomposition: first we need to order the independent components based on frequency instead of randomly generating them. Second, we need to save the exact decomposition on which the model has been trained. This way we can synchronise the new decomposition such that it fits the decomposition on which the model has been trained best.

Let's discuss the ordering of the components based on frequency. Keep in mind that all components are zero-mean because of the assumption that the data has been centred before we apply ICA (in fact, the components produced by VMD are already zero-mean). This means we can approximate the frequency of the components by counting the number of zero-crossings.

Definition 4.2.1 (Zero-crossing)**.** *A zero-crossing is a point where the sign of a function changes (e.g. from positive to negative). Graphically, it is the point where the function crosses the x-axis.*

The higher the number of zero-crossings, the higher the frequency of the component likely will be. Two decompositions (originating from the same dataset) which have been ordered based on their zero-crossings are shown in

fig. 18.



(a) A sorted ICA decomposition for our dataset..

(b) ..a second sorted ICA decomposition for our dataset.

Figure 18: Two sorted ICA decompositions already look more similar.

Although they look quite similar, it still seems that some of the components are each others negative counterpart (e.g. IC 2 and IC 4). To make the decompositions match even better we 'synchronise' the two decompositions. By synchronising we mean multiplying one component by -1 to make the two components as similar as possible.

The so-called 'true' decomposition is the decomposition on which the model has been trained, while the new decomposition will be synchronised to match the true decomposition as best as possible. In practice, this new decomposition stems from the new dataset on which the trained model will be used to make a forecast.

Synchronisation takes place after the ordering of the components. The synchronisation algorithm is described in alg. 7. Assume that S_A, S_B are the matrices containing the source signals for decompositions A and B. A_A and A_B are the corresponding mixing matrices for decompositions A and B. When synchronising the new decomposition to match the true decomposition, we should also synchronise the mixing matrix at the same time, otherwise the original signal can not be reconstructed correctly. Furthermore, assume that S_A and S_B have dimension $n \times m$, where m is the number of components in the decomposition, and n the number of time steps in the signal.

Algorithm 7: Synchronisation algorithm

Data: S_A, A_A and S_B, A_B, the mixing and source matrices for decompositions A and B.

Result: S_B, A_B the synchronised mixing and source matrices for decomposition B.

function `Synchronisation`(S_A, A_A, S_B, A_B)**:**
  for $i = 1 : m$ **do**
    if RMSE$(S_A[:, i], -S_B[:, i])$ < RMSE$(S_A[:, i], S_B[:, i])$ **then**
      $S_B[:, i] \leftarrow -S_B[:, i]$
      $A_B[:, i] \leftarrow -A_B[:, i]$
    end
  end
  return S_B, A_B
end function

The main idea of the algorithm is that it only synchronises the new component on a certain condition: if the RMSE between the new and the true component is lower when the new component has been synchronised. This way components which are already 'correct' are not changed. A visual interpretation of the procedure is shown in fig. 19.

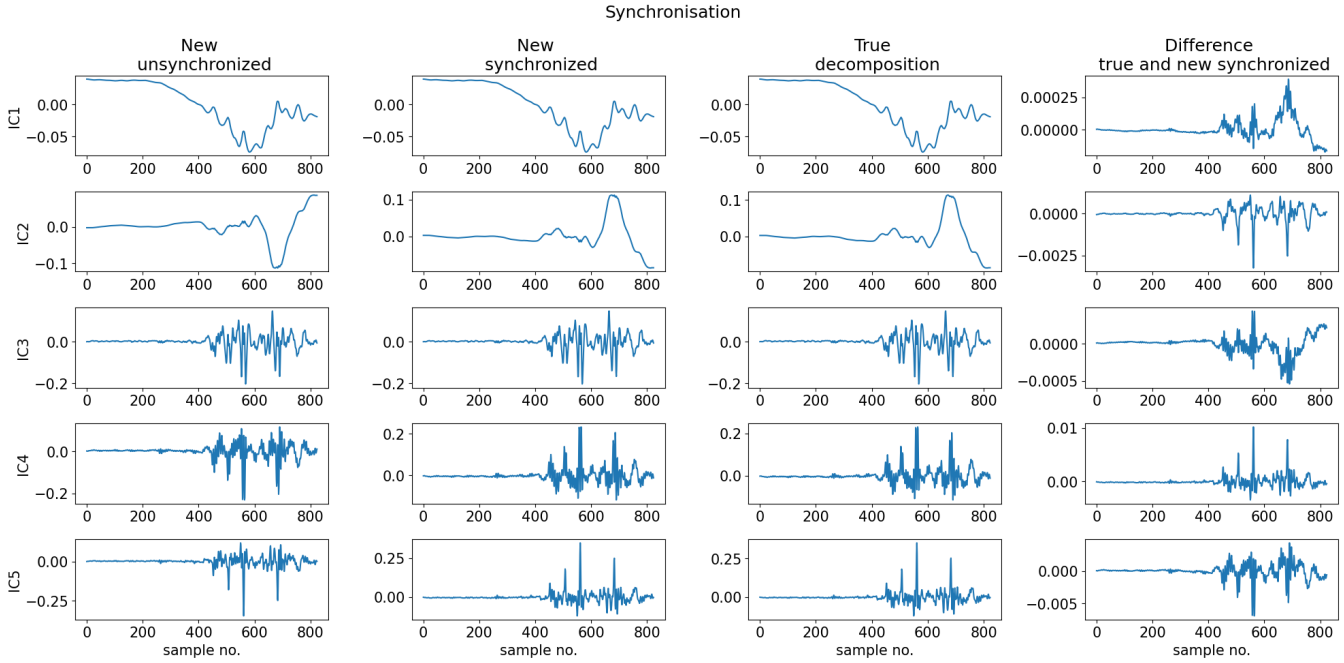


Figure 19: The synchronisation procedure visualised.

As we can see, IC 2 and IC 4 have been multiplied by -1 since they clearly fit the true decomposition better this way. Even after synchronisation however, small differences between the new decomposition and the true decomposition persist. Interestingly, some of these differences look very similar to some of the independent components (such as the differences of IC 5, which look like IC 4). This might indicate that the independent separation of the components is not entirely perfect. More research is needed to fully grasp this effect. Nevertheless, the synchronisation is a major step towards comparing model performances on different datasets.

4.2.2 Regime splits

Looking back at fig. 6, if a model is trained on the entire dataset, then it is trained on low volatility data and tested on high volatility data. This procedure will not produce a fair image of the model performance, since the training and testing set are incomparable. That is why we split the dataset into 5 regimes, shown in fig. 20.

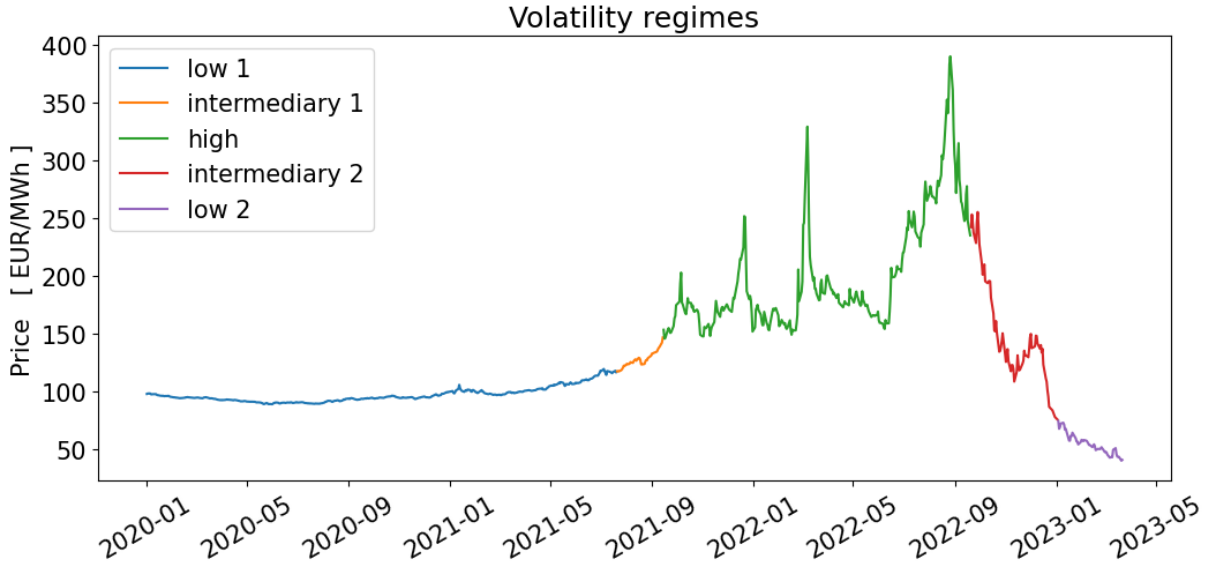


Figure 20: Proposed regime split to optimise forecasting performance.

We arrived at this regime split by inspecting the ICA decompositions for different regime intervals. We make a decision based on the comparison of the quality of the different decompositions. All intervals and their decompositions are

included in the appendix (figures 44-49). For simplicity and ease of comparison, sample numbers are given instead of dates. We start by defining the low volatility regime, which we observe is present by visual inspection of fig. 6. The question is what the length of this first low volatility regime should be. Upon first glance, the end of the low volatility regime seems to be somewhere around September 2021, which translates to roughly datapoint number 400. To find which interval is the best, we plot the ICA decompositions for different lengths of the low volatility regimes.

As we can see in fig. 21, the regimes with length 408 and 410 produce fairly different decompositions. This is especially visible in components 2 and 3. The reason decompositions D and E are not preferable is because IC 2 and 3 contain secondary trends. The complete decompositions are shown in fig. 44.

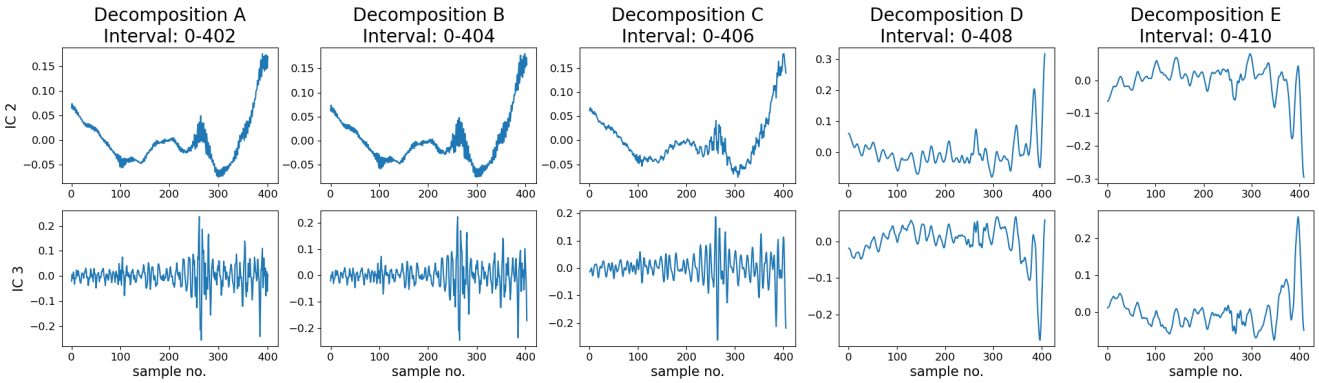
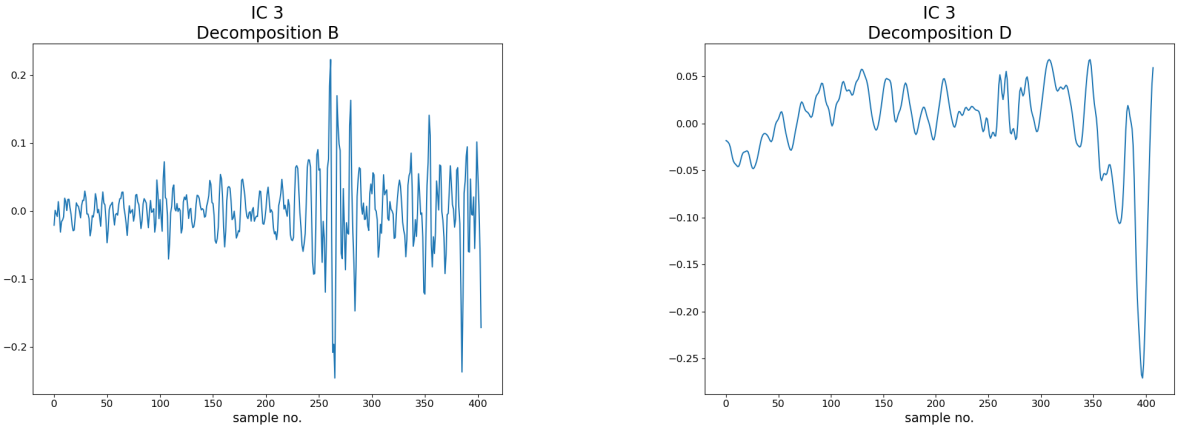


Figure 21: Decompositions D and E are different from the rest.

To illustrate the presence of a secondary trend, we compare the third component of decompositions B and D in fig. 22. As shown in fig. 22b, a slight upward trend is present until sample number 150, and a downward trend from sample number 350 onwards. Since we want to maximally separate effects, we do not want any kind of secondary effects in the components, as shown in fig. 22a.

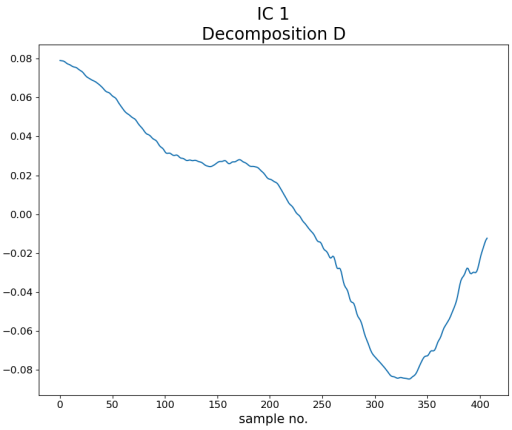


(a) Component without secondary trend.

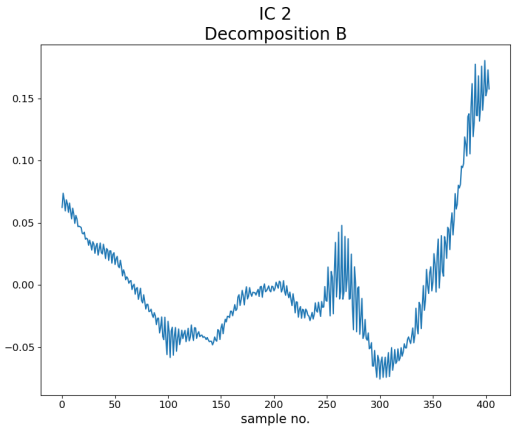
(b) Component with a secondary trend.

Figure 22: Two components illustrating the presence of a secondary trend.

A different secondary phenomena can be observed in IC 2 of decomposition B, shown in fig. 23b. Here we see higher frequency effects in what should have been a second lower frequency component. IC 1 of decomposition D has almost no secondary frequencies, and is much cleaner. Both components are taken directly from fig. 44.



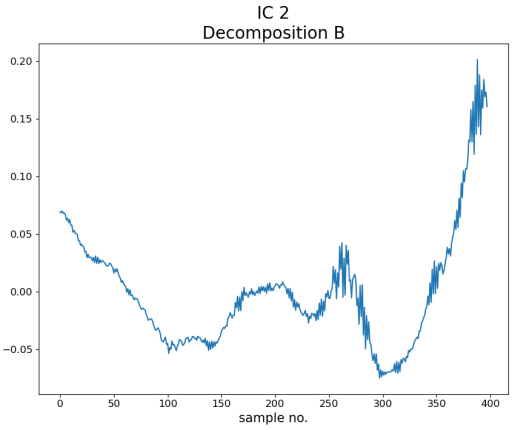
(a) Component without secondary frequency.



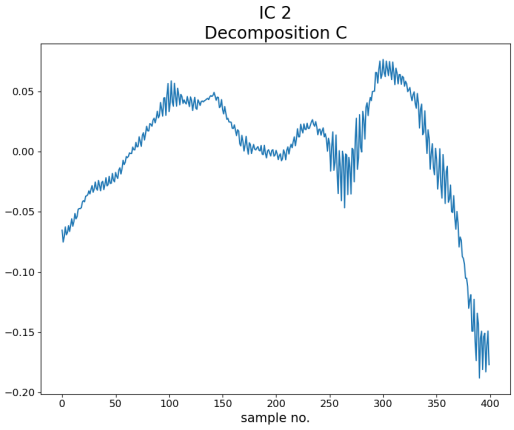
(b) Another component with a secondary frequency.

Figure 23: Two components illustrating the presence of a secondary trend.

To minimise secondary effects in the components we shorten the lower volatility regime with a couple of datapoints, and check different regime lengths again. The complete result of that is shown in fig. 45. We shorten the low volatility regime instead of elongating it, because by elongating we lose a slow component in decompositions D and E, as shown in fig. 21. Zooming in on IC 2 of decompositions B and C yields fig. 24. We conclude decomposition B has the least secondary frequencies or trends present. Since the other components are very similar, we set the length of our volatility regime to 398 samples. In terms of dates, this translates to all transactions from the 2nd of January 2020 until the 20th of July 2021.



(a) Component 2 for a decomposition on samples 0-397 has less secondary frequencies.



(b) The secondary components for longer intervals have more secondary frequencies.

Figure 24: Component 2 of decomposition B has less secondary frequencies.

Repeating this procedure of comparing decompositions to find new regimes leads to the regime split shown in fig. 20. The decomposition comparison for the other regimes is taken up into the appendix. Note that for the high volatility regime we had to use more components to see a clearer difference on where to split the regime. What number of components to use for the high and low volatility regimes will be discussed in the next section. Also, the regime intervals are double checked by evaluating the forecasting performance, to make a final decision. The regimes with their sample intervals and corresponding dates are summarised in table 7.

Regime	Sample interval	Regime length	Corresponding dates
Low 1	0-397	398	2020/01/02 - 2021/07/20
Intermediary 1	398-437	40	2021/07/21 - 2021/09/14
High	438-699	262	2021/09/15 - 2022/09/19
Intermediary 2	700-771	72	2022/09/20 - 2023/01/02
Low 2	772-823	52	2023/01/03 - 2023/03/21

Table 7: Summary of regimes shown in fig. 20.

We are mainly interested in forecasting prices for the low and high volatility regimes. The intermediate regimes are too short to completely train, validate, and test a model, and are thus of less interest to the research. For now, we will train, validate and test the models on regimes Low 1 and High. If time allows, we will look at combining the Low 1 and Low 2 regimes to research whether it improves our forecast. Statistics for selected regimes are summarised in table 8.

	Regimes	
	Low 1	High
mean	97.13	199.25
std	6.68	49.69
min	88.85	145.66
max	119.39	390.22
1st Q	92.60	166.09
3rd Q	99.62	220.46

Table 8: Summary statistics for low and high volatility regimes.

Similar to the statistics shown in table 6, the standard deviation between the two regimes is the biggest difference. The standard deviation of the high volatility regime is roughly seven times higher than the standard deviation of the low volatility regime, and the mean is roughly two times higher. Notice that the standard deviation of the low volatility regime is roughly in the same ballpark as in the studies described in table 6. On the contrary, the high volatility regime has a very high standard deviation compared to the other studies. These differences also follow from the minima and maxima of the regimes and the quartiles.

4.2.3 Number of components

Now that we have established the regime intervals, we need to determine the optimal number of components to use for the ICA decomposition of each regime. Since the high volatility regime has more complex dynamics, we expect more components are needed to correctly describe it. In fig. 25 we compare the first three components of three different decompositions (with 5, 6 and 7 components) made on the low volatility regime.

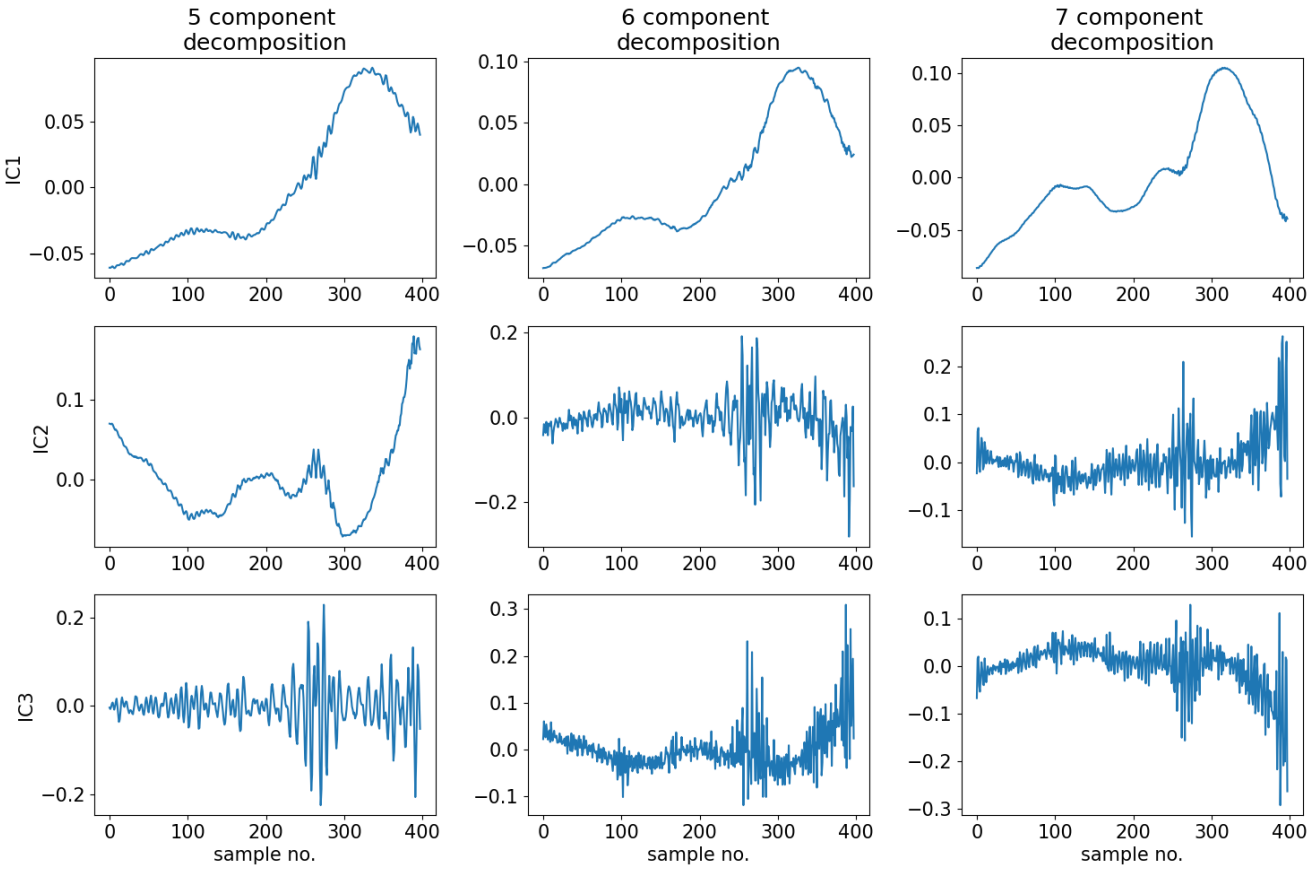


Figure 25: The first three components of three different low volatility decompositions.

From fig. 25 we can deduce that every decomposition with more than 5 components is less preferable for the low volatility regime. When looking at the decompositions with more than 5 components we see that some of the components contain secondary effects. Besides this, when increasing the number of components from 5 to 6 or more, we lose a low frequency component. Based on the full comparison shown in fig. 50, it even seems like 3 components would be enough to describe the signal, although we will lose some information, as will be discussed in section 4.2.4.

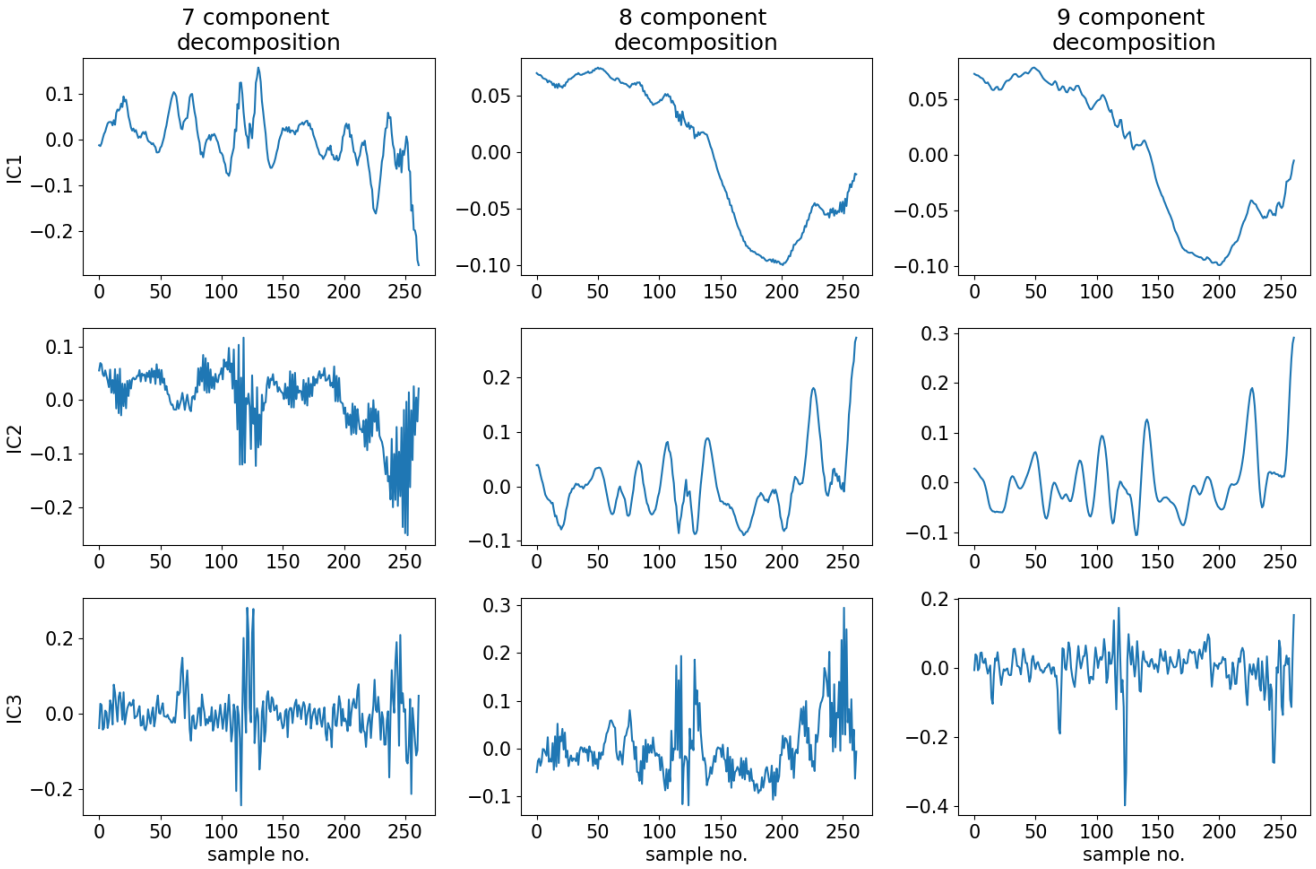


Figure 26: The first three components of three different high volatility decompositions.

From fig. 26 we can conclude that 7 components is too little to describe the high volatility regime, since no clear slow component is present. There is not much difference between the low frequency components of decompositions with 8 components or more for the high volatility regime. For the sake of computational speed, we choose 8 components for the high volatility regime. All decompositions with different numbers of components are shown in fig. 50 for the low volatility regime, and in fig. 51 for the high volatility regime.

4.2.4 Accuracy

When discussing the VMD-ICA decompositions, an important side note needs to be made. The reconstruction of the original series will not be perfect for the number of components we are using. Usually, the higher the number of components in the decomposition, the lower the error in the reconstruction is. Table 9 summarises reconstruction errors for the high and low volatility regimes.

Low volatility		High volatility	
Number of components	MAPE	Number of components	MAPE
3	0.24%	3	2.38%
5	0.19%	5	1.71%
8	0.14%	8	1.15%
10	0.13%	10	0.69%

Table 9: MAPE errors for VMD-ICA decompositions with different numbers of components.

We see that the MAPE error decreases as the number of components increases. This makes sense, since more information can be captured with more components. Also, the MAPE error for the high volatility regime is higher than for the low volatility regime, which justifies our choice of picking more components for the high volatility regime. Figure 27 shows two reconstructions and the corresponding original price series for the first 100 samples of the high volatility regime.

We can see that the decomposition with 3 components loosely follows the movement of the price, without following the outliers very closely. The decomposition with 10 components follows the pricing series much closer, although it still does not capture the outliers perfectly. 10 components is not necessarily better than 3 components. Sometimes it is preferable to have a model which is less subject to outliers. Besides this, a model based on 3 components will be faster to train than a model based on 10 components.

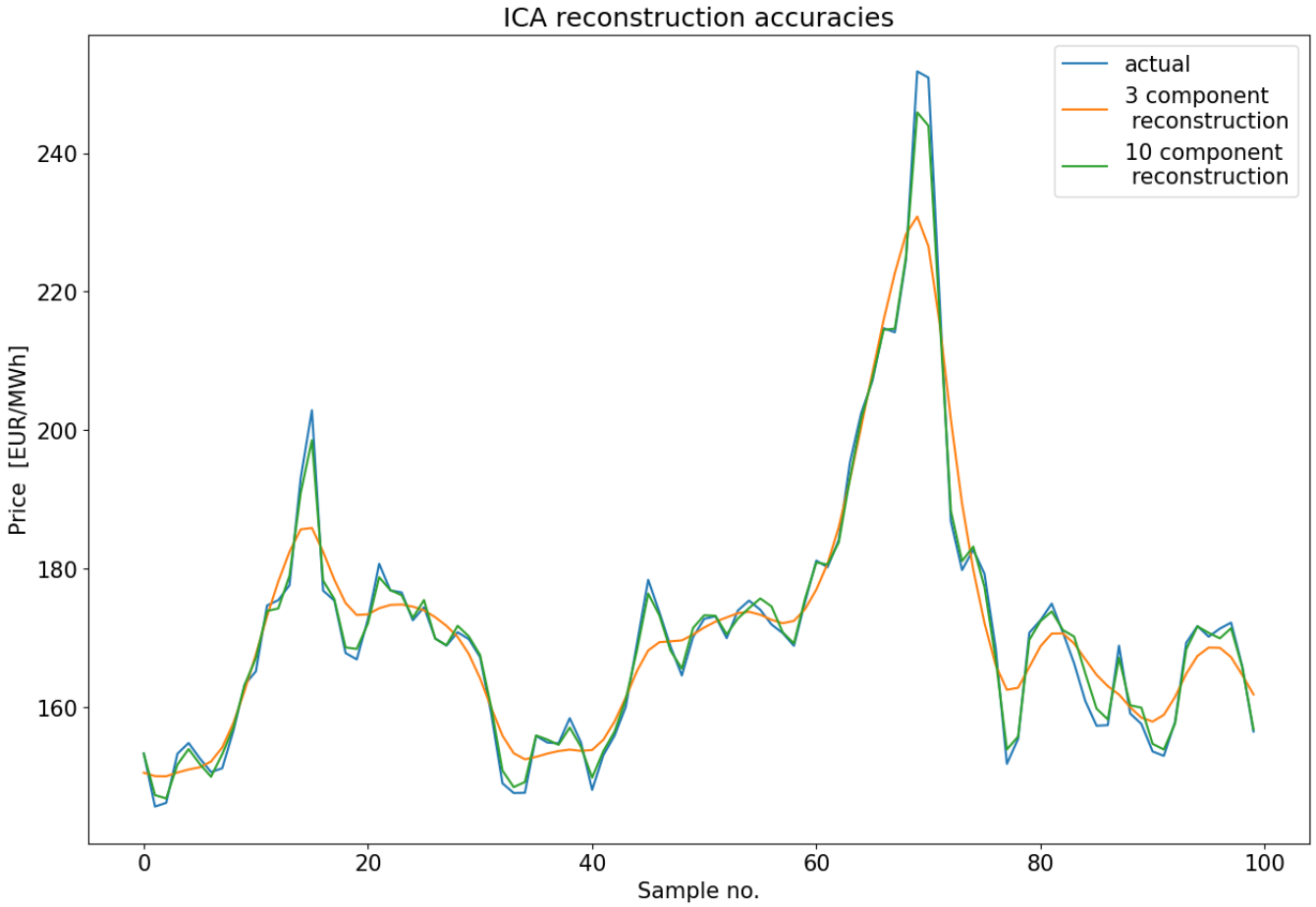


Figure 27: ICA reconstructed price series comparison.

4.3 Forecasting: GRUNN

Now that we have an initial indication of the regime splits and the number of components to describe the pricing series, it is time to make the first forecasts. The first forecast is made by the GRU-neural network. For this first analysis, the GRUNN makes a forecast for each component individually, after which the signal is reconstructed using the mixing matrix and mean produced by the ICA, as depicted in fig. 15a, without incorporating SVR yet.

4.3.1 GRUNN parameter settings

Before discussing results of the GRUNN forecast, there are a number of hyperparameters which are important to define and explain. Hyperparameters are parameters set by the user of the model, and can have a large impact on the performance or training time of the model. The hyperparameter settings for the models trained on the low and high volatility regimes are shown in table 10.

	High volatility regime	Low volatility regime
Rolling window size	3	15
Layers	1	1
Cells per layer	32	32
Batch size	4	4
Train share	70%	70%
Validation share	5%	5%
Test share	25%	25%
Maximum number of epochs	50	50

Table 10: Most important hyperparameters of the GRUNN model.

The rolling window size denotes the number of samples the GRUNN model takes as input before making its first forecast. For training, the forecasts are then compared to the corresponding actual prices attained. A schematic overview of the rolling window and label split is given in fig. 28 for a rolling window size of 6 and 1 label value. The label value represents the price which the GRUNN model should forecast. In the case of a multi-day forecast, where we forecast multiple days ahead at once, the label width would not be 1, but higher, depending on the number of days we want to forecast at once.

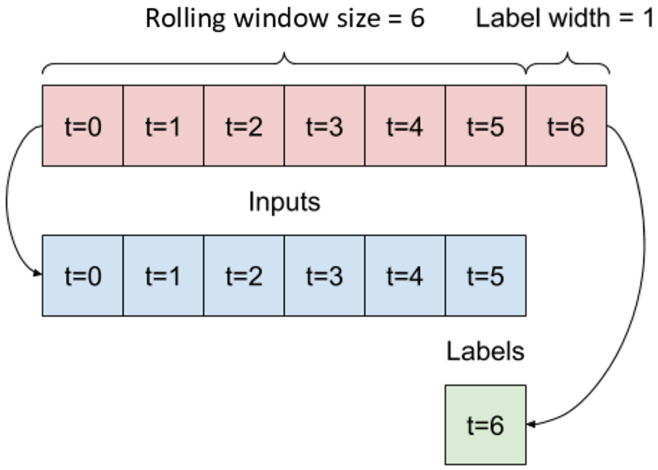


Figure 28: Visual representation of rolling window size and label width.

We have chosen a relatively simple architecture for the neural network, since our dataset is not large enough to train multiple layers of GRU cells. Our GRUNN architecture consists of three layers: a convolutional input layer, a GRU layer and a simple dense output layer. The dense output layer has the same number of output nodes as the number of ICA components we want to forecast. The purpose of the convolutional input layer is that it scales the inputs to the correct dimension for the GRU layer. It is common practice to choose the number of cells per layer as a power of 2. We found that we had the best results when choosing 32 GRU cells for the GRU layer.

The batch size determines how many training samples a single batch contains. During a training iteration (also called epoch) the GRUNN is being trained on a single batch. Increasing the batch size means the GRUNN will be trained quicker, since there are less epochs to complete. However, this means its performance may be worse since the model has less iterations to fit to the data optimally. We found that decreasing the batch size improved model performance. This may have to do with the fact that we have a relatively small dataset (compared to usual applications of neural networks). The train, validation and test shares all have been tailored to our dataset, and represent the share of samples in each set relative to the total dataset. To make sure the training does not go on infinitely, we set a maximum number of training epochs. Increasing this number leads to a longer training time, not necessarily to a better fit. Full hyperparameter tuning results are presented in the appendix, in tables 20-23.

4.3.2 Performance measures for training

One model parameter not mentioned in table 10 is the type of loss function to use during training. The ICA decompositions, which are zero-mean by definition, form the training set for the GRUNN. This has impact on the value of the loss function which we use for training. For example, the MAPE of a forecast performed on a zero-mean label series will likely be inflated, since the denominator in the calculation will be close to zero. In our case, we

observed training-MAPEs of 200% or more, while our final fit (after reconstructing the original price series) would be much more accurate in terms of MAPE. This means the MAPE is not really suited for the training of the GRUNN models, something which is clear from the results in table 11 as well.

Some common loss functions used for training neural networks are the MSE and RMSE. Considering the input of the GRUNN model, we deem the RMSE to be more suitable. Since most components fluctuate between 1 and -1, calculating MSE results in relatively small loss values (we observed orders of magnitude in the range of 10^{-5}). When training the GRUNN the process is moving towards a minimal loss value. In an ideal scenario this path to the minimum is direct. The smaller the differences in loss values, the more sensitive this process will be to noise, and the less direct this path to optimality will be. The benefit of the RMSE versus the MSE is that it inflates the differences in loss values (since we take the root of a value between 0 and 1 the loss value becomes larger), this makes the direction to the optimum in the parameter space more clear during training.

The difference in test performance between the RMSE and MSE trained models is larger for the high volatility regime. This could be because price levels are (much) higher in the high volatility regime compared to the low volatility regime, which means the loss values are even smaller (since the error is divided by a larger denominator). Inflating them could thus improve the search for optimality even more. Comparisons of the RMSE and MSE trained GRUNN models in term of the MAPE on the test set are shown in table 11.

	Regime	
	Low volatility test set MAPE	High volatility test set MAPE
MSE trained	1.05%	5.06%
RMSE trained	0.71%	4.18%
MAPE trained	7.72%	22.9%

Table 11: Test set MAPE performance comparison between a GRUNN model trained using RMSE and MSE.

This effect has to be researched more thoroughly, which is outside of the scope of this research. For now though, this result is enough for us to make the decision to train our models using RMSE. The comparison in table 11 has been made using the hyperparameter settings shown in table 10.

4.3.3 Regime calibration based on forecast

Besides using the ICA decompositions to define the regimes, as in section 4.2.2, we can use the final forecasts as well, as a way to double check the regime definitions. Defining the regimes based on the forecasting performance makes sense, since the forecasting performance is eventually the objective to optimise. Performing multiple train-validate-test sequences over a selection of regime intervals gives us an overview of model performance on different regime intervals. The best performing models with their corresponding regimes are shown in figures 29 and 30. A complete overview of all runs is shown in figures 52 and 53. The interval for the low volatility regime which produces the best average MAPE over three runs is 0-399, with an average MAPE of 0.49%. For the high volatility regime the interval producing the best average MAPE over three runs is 438-699, with an average MAPE of 2.02%

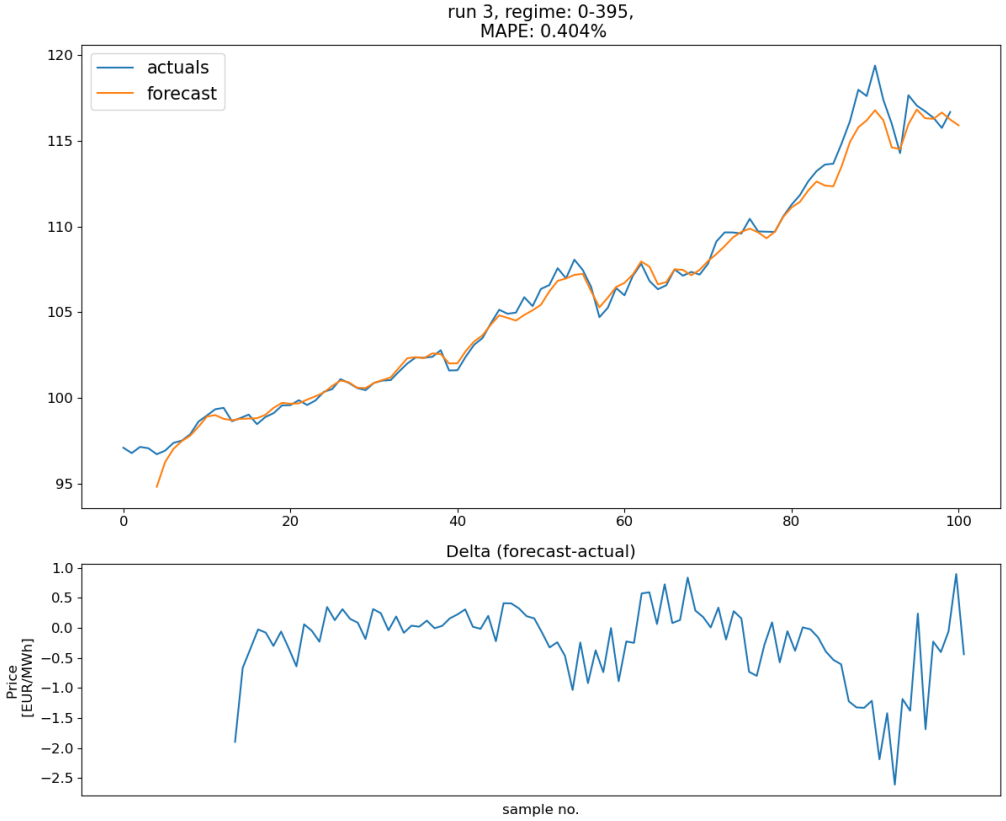


Figure 29: Best performing model over multiple runs for different low volatility regime intervals. Forecasting starts at sample no. 3, due to input settings mentioned in table 10.

As we can observe in both figures fig. 29 and fig. 30, no trends are present in the delta's. This indicates that the GRUNN model probably did not miss any significant effects present when modelling the price series.

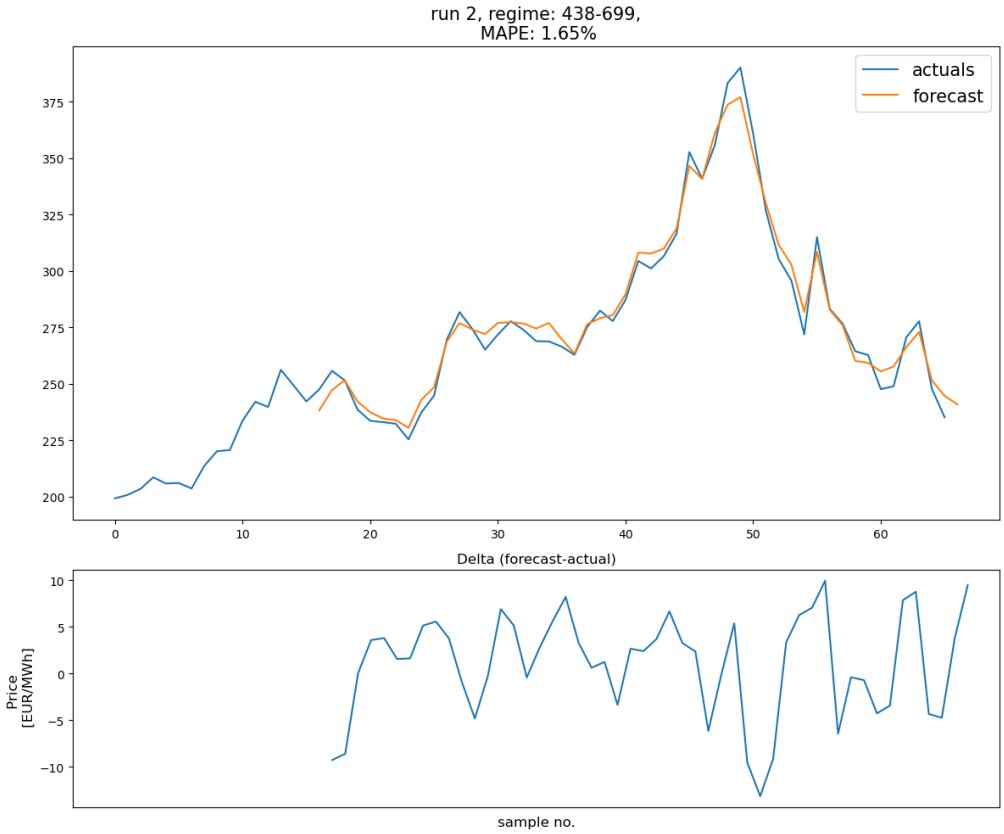


Figure 30: Best performing model over multiple runs for different high volatility regime intervals. Forecasting starts at sample no. 15, due to input settings mentioned in table 10.

4.3.4 Number of components based on forecast

When assessing the optimal number of components based on the forecast, for the low volatility regime we will be looking at decompositions with 3, 4 or 5 components, and for the high volatility regimes at decompositions with 8, 9 or 10 components. Again, as in the previous section, for each situation the train-validate-test cycle will be run three times to get a better indication of the overall performance of the setup. The regime choices in the plots below are 0-397 for the low regime and 438-699. The best performing models out of the analysis are shown in fig. 31 and fig. 32. The complete overview of all runs of the models is shown in the appendix, fig. 54 and fig. 55.

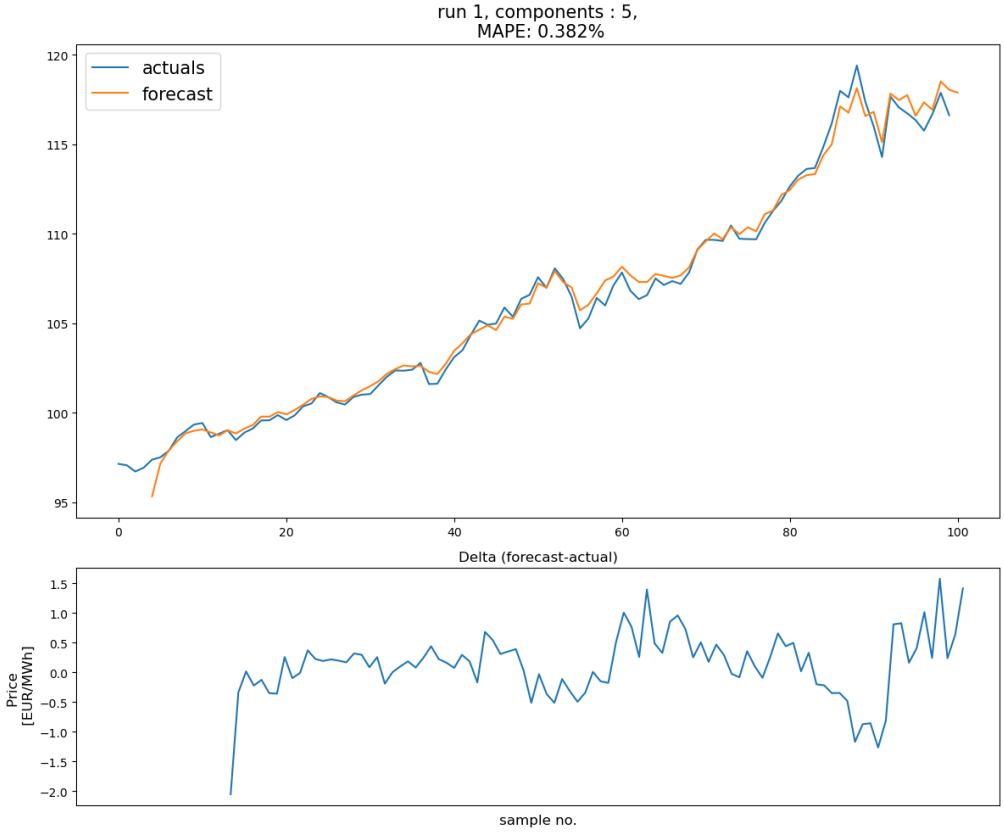


Figure 31: Best performing model over multiple runs for different number of components for the low volatility regime.

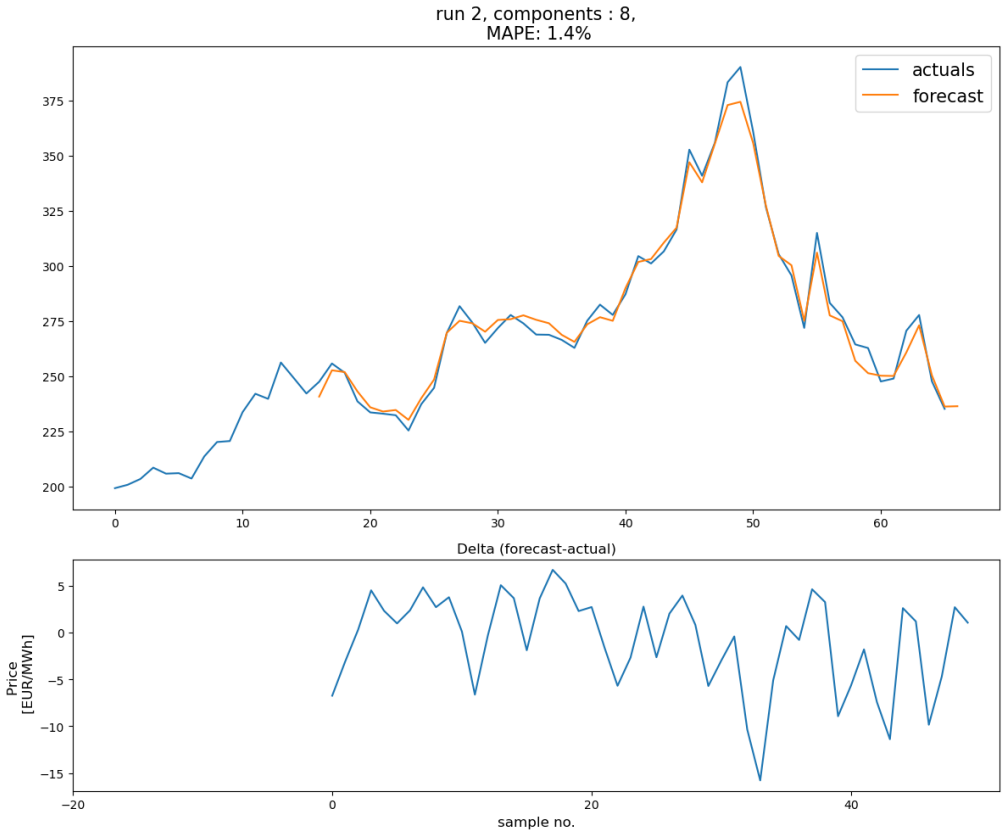


Figure 32: Best performing model over multiple runs for different number of components for the high volatility regime.

When combining the results from both sections section 4.3.3 and section 4.3.4, we see that the best MAPE scores we achieve are 1.4% for the high volatility regime and 0.38% for the low volatility regime. Full comparisons of regime definitions and number of components based on decomposition inspection, and final forecasting performance is shown in table 12.

Decision based on	Volatility regime	Best regime interval	Best MAPE	Best number of components	Best MAPE
Forecast	Low	0-395	0.404%	5	0.382%
Decomposition	Low	0-397	-	not more than 5	-
Forecast	High	438-699	1.65%	8	1.4%
Decomposition	High	438-699	-	not less than 8	-

Table 12: Comparison of results of two approaches, one based on decomposition inspection and one based on forecasting performance.

Based on table 12, we chose the final settings of the regimes and number of components. For the low volatility regime we choose the interval 0-395 and 5 components. For the high volatility regime we choose the interval 438-699, and 8 components. In section 4.5 these MAPE scores are compared to some benchmark models.

4.4 Forecasting: SVR

All the component forecasts up until this point have been integrated into a single price forecast simply by reversing the ICA decomposition. However, recall that SVR can also be used to integrate the component forecasts into a single price forecast, as proposed in E et al. (2019). Unfortunately, E et al. (2019) describe their methods and results only high-level. This is appropriate for a journal publication, however for reproducing results this is less convenient. The lack of detail in E et al. (2019) becomes an issue when implementing SVR. In their paper, E et al. (2019), state the following about the implementation of SVR in their forecasting approach:

> "Due to the independence among the ICs, the tendency of each IC can be predicted accurately, we adopt the GRUNN model which contributes to capture the short-term and long-term dependencies in temporal sequence to predict each IC and we can obtain the forecasting results $F_1(t), ..., F_K(t)$ of each IC. The final prediction result of the energy price $F(t)$ is the integration of $F_1(t), ..., F_K(t)$ via SVR."

After reaching out to the authors for some extra clarification[7], we decided to compare the approach of E et al. (2019) to some variations on it.

E et al. (2019) propose to take the row sum of the mixing matrix A, resulting in a set of K weights, corresponding to the K components produced by the ICA decomposition. One of the underlying reasons for doing this could be to further separate the ICs, since SVR thrives on datasets which can be (linearly) separated. The original source ICs are obviously not very well separated since they are all zero-mean.

In this line of reasoning, we could further separate the ICs by adding the corresponding mean to each component. This way, we separate the components even more, hopefully benefiting the performance of the SVR. The results of all approaches are shown in table 13. The results shown in table 13 have been produced by choosing $C = 2000$, and $\varepsilon = 0.01$ as hyperparameters for the SVR model. As we recall: ε defines the radius of the loss free tube around the hyperplane. C regulates the flexibility of the separating hyperplane: the higher the value, the more flexible the hyperplane.

SVR input		regime	
weighted	mean added	low vol	high vol
no	no	6.71%	23.41%
yes	no	3.19%	24.47%
yes	yes	1.46%	1.38%

Table 13: MAPE scores of different SVR implementations as integration method.

[7] Paraphrasing the reply of dr.ir. Jianwei E: "After predicting, each of the $F_i(t)$, $i = 1, ..., K$ need to multiply by coefficient, respectively, that is a_i (acquired by the code 'sum(A)' in MATLAB, here 'A' is the mixing matrix) in Eq. (24) [of the original paper]. Secondly, we consider SVR as a integration method, wherein the traning [sic] input of the SVR model is the forecasting results (such as the prediction results of 5 ICs with corresponding coefficients for the Natural gas price) with the size of 80 percent and the training ouput [sic] is the original price series with the size of 80 percent."

We observe that, indeed, further separating the components by adding the ICA mean to each component enhances the SVR performance. Note that SVR does not improve the forecasting performance with the current settings. Continuing with the weighted components, and the mean added, we optimise the hyperparameter settings for SVR. The results are shown in table 14.

C \ ε	1e-5	1e-3	0.1	1	2	5
	low volatility regime					
100	1.17%	1.17%	1.22%	1.48%	2.49%	5.8%
500	1.25%	1.25%	1.23%	1.13%	1.81%	3.0%
1000	1.21%	1.21%	1.42%	1.08%	1.61%	3.0%
2000	1.47%	1.48%	1.46%	1.1%	1.37%	3.0%
5000	1.42%	1.42%	1.42%	1.5%	0.75%	3.0%
10000	1.44%	1.43%	1.43%	1.49%	0.74%	3.0%
	high volatility regime					
100	16.19%	16.19%	16.33%	16.42%	17.38%	18.31%
500	5.52%	5.52%	5.41%	4.66%	4.3%	5.81%
1000	2.6%	2.6%	2.44%	2.28%	2.56%	3.31%
2000	1.45%	1.42%	1.42%	1.6%	1.83%	2.39%
5000	1.65%	1.65%	1.74%	1.5%	1.8%	2.32%
10000	1.5%	1.51%	1.78%	1.42%	1.68%	2.17%

Table 14: Hyperparameter tuning results of SVR.

Notice the MAPE being constant for different values of C at the low volatility regime, when $\varepsilon = 5$. This has to do with the loss-free tube defined around the hyperplane. If it is set to 5 euros, most of the samples will fall within the loss-free tube. Increasing the flexibility of the hyperplane will not alter the loss of the fit (and thus the forecast made by SVR), since all samples lie within the loss-free tube. This causes the MAPE to be constant.

Overall, we do not observe an improvement in forecast when applying SVR to the forecasts made by GRUNN, compared to reconstructing the price series directly using the ICA mixing matrix A and mean. The best forecast we produce by using SVR as an integration method produces a MAPE of 0.74% for the low volatility regime, and 1.42% for the high volatility. The MAPE errors of the GRUNN forecasts which form the input for SVR are 0.38% and 1.4% respectively, when reconstructed using ICA. Thus, SVR did not improve the forecast compared to remixing the component forecasts made by GRUNN.

4.5 Benchmark performance

A benchmark is a common, established forecasting model to which other, new models are compared to measure their relative performance. Multiple common benchmarks are used across the literature. A notoriously hard benchmark to beat is the naive forecast discussed in section 4.5.1. The models proposed in Čeperić et al. (2017), Wang et al. (2020) and Herrera et al. (2019) are benchmarked by the naive forecast. Other benchmarks include an ARIMA model (as in Siddiqui (2019) or Čeperić et al. (2017)), a random walk process (Nguyen and Nabney (2010)) or other models proposed in different papers. For example, Berrisch and Ziel (2022) claim to beat the forecast made by Herrera et al. (2019) in their paper. When choosing benchmarks the main question to keep in mind is what we would like to show by comparing our models to them.

We have chosen 3 benchmark models: a simple naive forecast, a GRUNN model trained directly on the price and a GRUNN model trained on the VMD-ICA decomposition of the entire dataset. Comparing our models to the naive forecast will show if our efforts have been fruitful at all. By comparing our models to a benchmark model trained directly on the price we want to show that the preprocessing of the data adds value to the forecast. Finally, comparing our models to a GRUNN model trained on the VMD-ICA decomposition of the entire pricing series will reveal the impact of the regime definition on the forecasting performance.

4.5.1 Naive forecast

The naive forecast takes the last known price as the forecast for the next period. Mathematically we can state:

$$F_{t+1}(X_t) = X_t. \tag{104}$$

Where $F_{t+1}(X_t)$ is the forecast for the next period, and X_t is the actual data known at time t. As an example, the naive forecast is graphed for the first 25 samples of the high volatility regime in fig. 33.

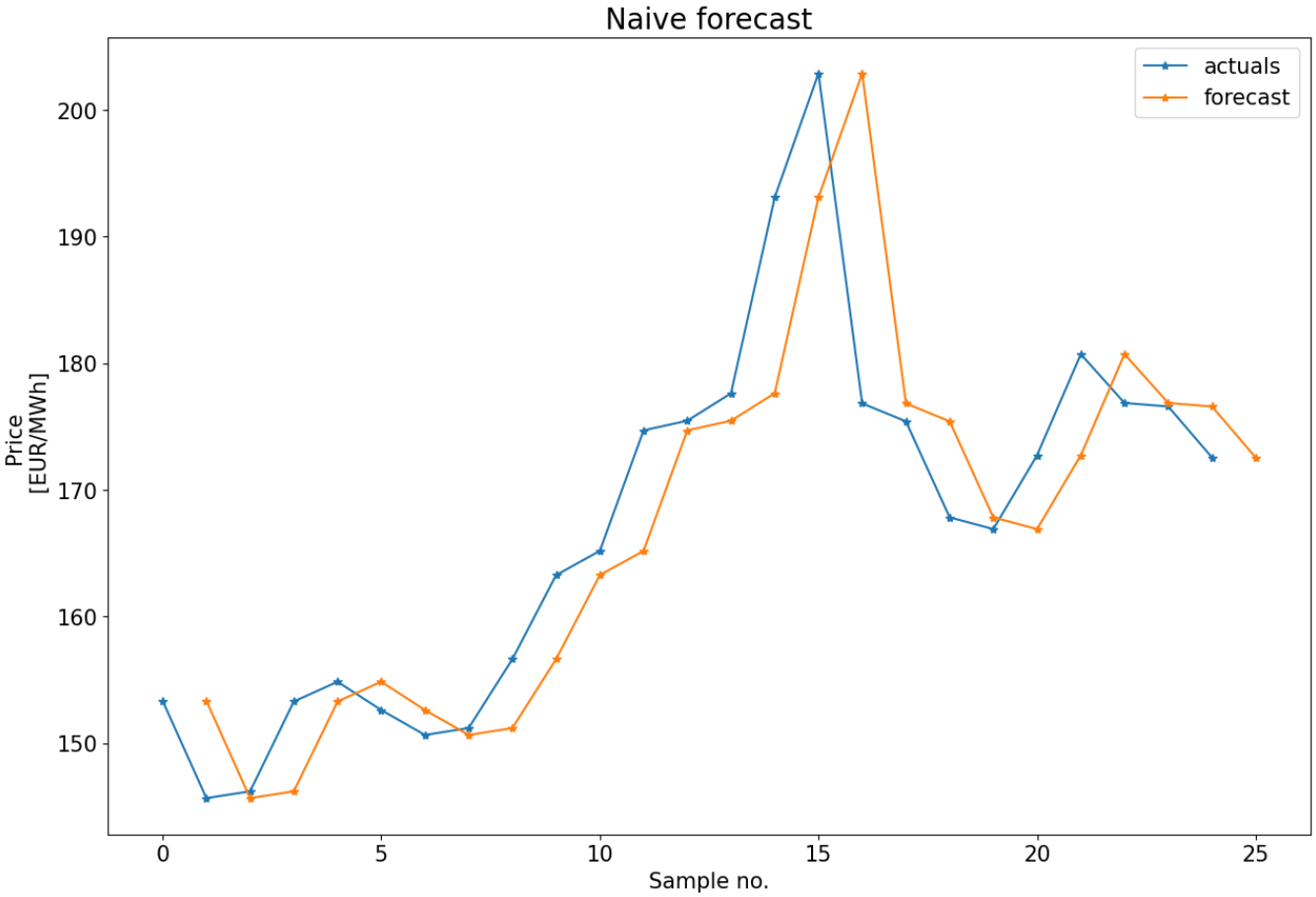


Figure 33: Naive forecast compared to high volatility regime actuals.

Notice that a visual illusion may occur when looking at fig. 33, where the observer assesses the quality of the forecast by comparing forecast and actuals horizontally. This is wrong, since we should only compare samples which are at the same sample number. This means they must be vertically aligned, instead of horizontally. Smaller errors can be observed in the forecast of sample number 12, and larger errors at sample numbers 14 and 15 for example. Now, if we apply the naive forecast to the entire dataset, the MAPE error is equal to 1.99%. This is worse than our best forecasting performance on the high and low volatility regime, which equals 1.4% and 0.38%.

If we calculate the performance for the naive model on the same test sets used for the assessment of the GRUNN in fig. 52 and fig. 53, the MAPE equals 0.56% for the low volatility regime, and 3.9% for the high volatility regime. This means we outperform the naive forecast as well when we are comparing performance on the original test sets. Notice that the naive forecast performs better on the low volatility regime than on the high volatility regime. Obviously, this has to do with the fact that the price changes much more in the high volatility regime, which causes the naive forecast to be further off more often.

4.5.2 GRUNN trained directly on price

To test whether the preprocessing of the data improves the forecast of the model, we compare our models to a GRUNN model trained directly on the price level, without preprocessing it first. Keeping the same hyperparameter settings as in table 10, the train, validation and test set are shown in fig. 34.

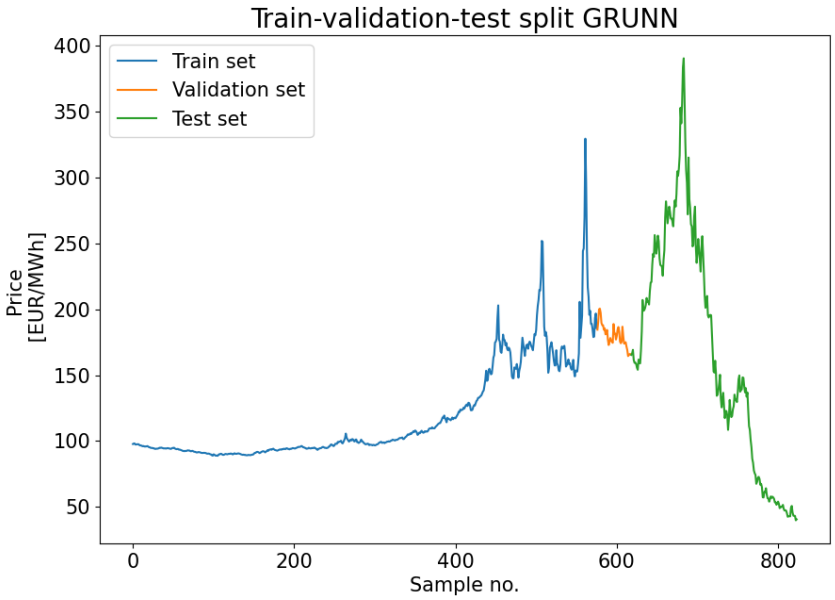
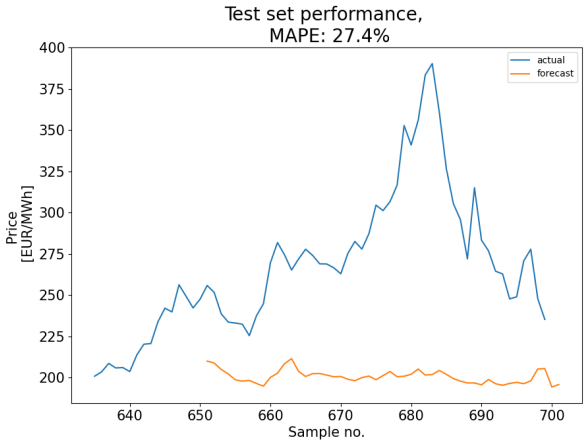


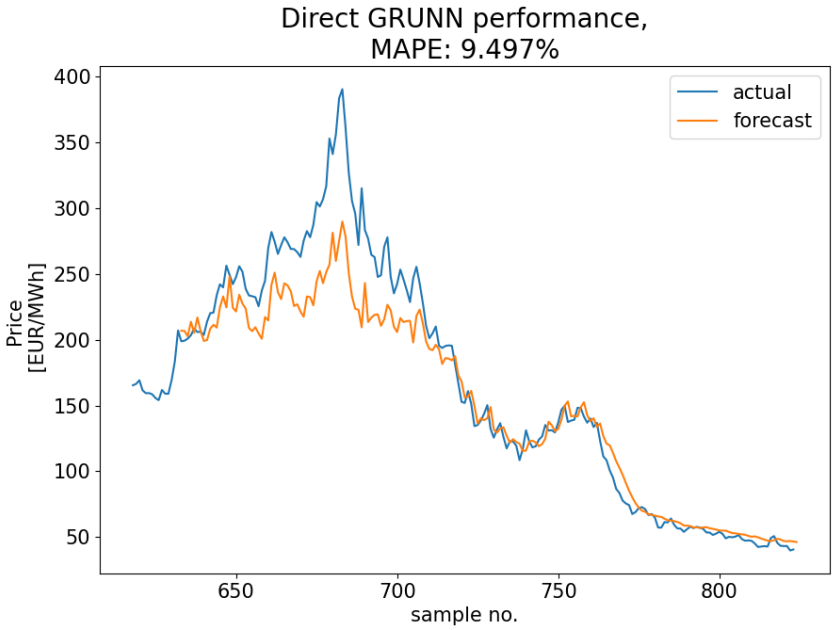
Figure 34: Train, validation and test set for the direct GRUNN benchmark models.

As mentioned before, the model is for a large part trained on relatively low volatility data, but tested on a dataset with high volatility data. The performance of the model directly trained on the pricing data is shown in fig. 35. To be able to make the comparison with the model performances shown in fig. 31 and 32, we also evaluated this benchmark model on the same regimes as in fig. 31 and 32. All benchmark performances together with the performance of the proposed model are shown in table 15.

(a) High volatility regime forecasting performance.



(b) Low volatility regime forecasting performance.



(c) Total dataset forecasting performance.

Figure 35: Three approaches to a direct forecast on price.

4.5.3 GRUNN trained on decomposition of entire dataset

To test whether the regime approach improves the price forecast, we will train and evaluate a GRUNN model on the ICA decomposition of the entire dataset. The train, validation and test sets are shown in fig. 36. Instead of a one-dimensional feature space as in fig. 34, we now have a 5 dimensional feature space. Each independent component forms an extra (close to orthogonal) dimension.

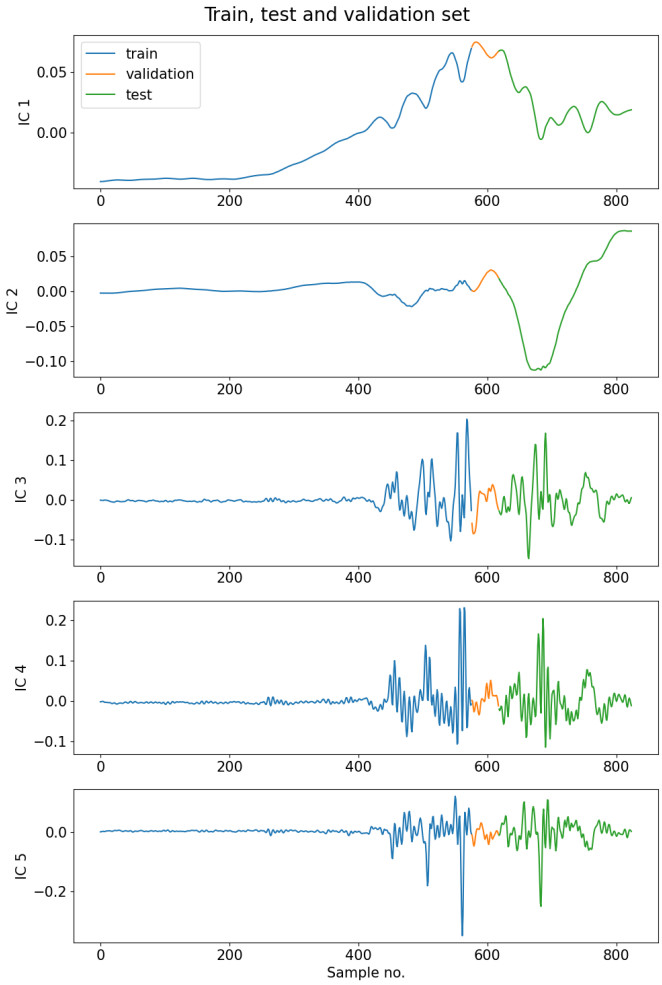


Figure 36: Train, validation and test sets for the GRUNN model trained on an ICA decomposition of the entire dataset.

Comparing fig. 37 to fig. 35c, then we observe that performance has increased from 9.50% MAPE to 6.37% MAPE. The forecast improved the most at the beginning of the test set, where the the Russian-Ukrainian conflict occurred.

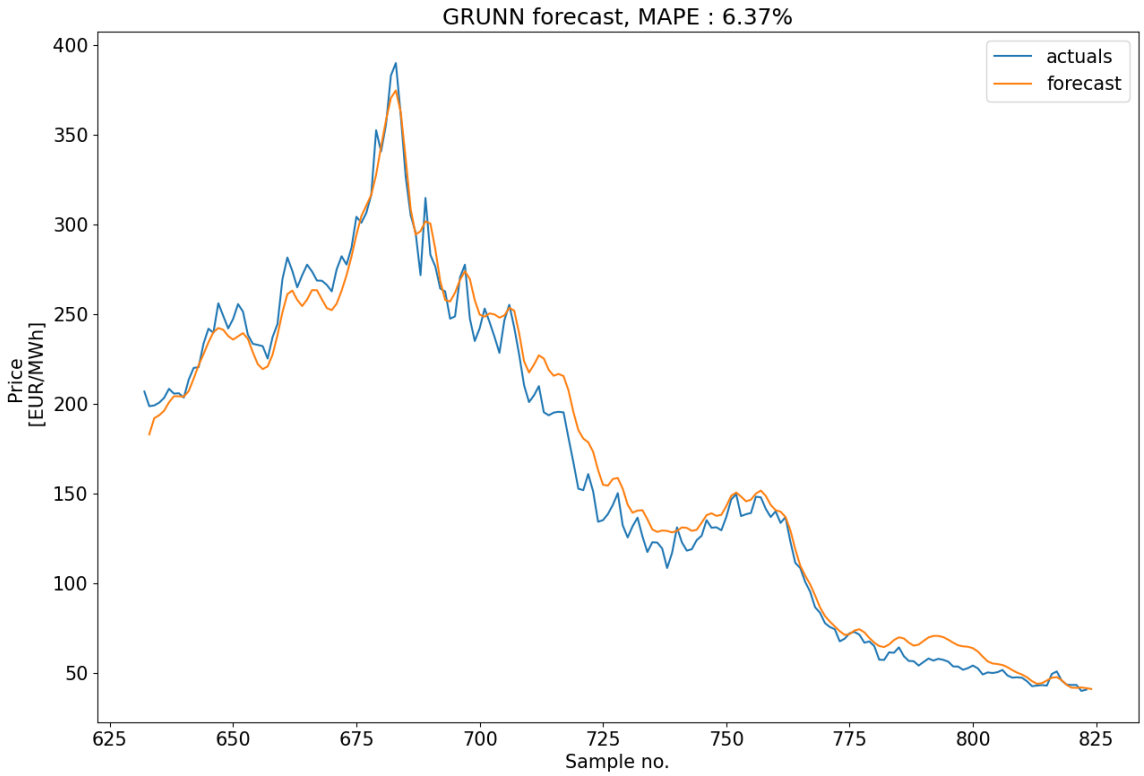


Figure 37: Forecasting performance of GRUNN trained on ICA decomposition of entire dataset.

The results our models produced versus the benchmarks we propose are summarised in table 15.

Benchmark	Regime	MAPE	Regime	MAPE	Regime	MAPE
Naive	Low	0.56%	High	3.90%	Entire	1.99%
Direct	Low	3.58%	High	27.4%	Entire	9.50%
Decomposition	Low	-	High	-	Entire	6.37%
Proposed model	Low	**0.38%**	High	**1.4%**	Entire	-

Table 15: Comparison of best models proposed in this work compared to proposed benchmark models.

The best performing models using the proposed approach of preprocessing the data and training two separate models on different volatility regimes outperform all of the benchmark models. This means a couple of things:

1. Preprocessing the price series using VMD and ICA helps the performance.
2. Splitting the price series into 5 regimes, and focusing on the high and low volatility regimes increases forecasting performance.
3. Only splitting the price series into regimes and not performing preprocessing does not improve the forecasting performance compared to the proposed model.

4.6 Exogenous variables

Exogenous variables are variables which are included in a model which do not influence the objective value directly, but indirectly. In our case, these are variables which influence the gas price indirectly. We choose to carefully select a number of exogenous variables which make sense to add. This means they either improve the forecast for the low frequency components (or low volatility regime), or for the high frequency components (or high volatility regime) in the ICA decomposition, or for both.

To improve the forecast of the low frequency component, we decide to add components which are low frequency by nature as well. An example of a component with low frequency is the gas storage level in Europe, which shows seasonal oscillations. Gas storage facilities are traditionally filled in summer, when gas prices are lower. Then, in winter, when the weather is colder than expected or for other reasons, gas storages are emptied again. This causes the gas storage levels in Europe to exhibit a very strong seasonality with a period of roughly 6 months. It makes for a good exogenous variable to improve the slow component forecast of the price ICA decomposition.

For the higher frequency components we chose a different exogenous variable. To improve the high frequency forecast, we choose the British NBP gas price. The UK is more reliant on LNG and only has two connections to mainland Europe, one through the Interconnector pipeline, which is connected to Zeebrugge, Belgium, and one through the Bacton Balgzand Line (BBL) to The Netherlands. This makes that the NBP gas price usually is more sensitive to price shocks, news or weather events for example. These effects influence the high frequency components, we try to capture these effects by including the NBP price.

For the exogenous variables we use the same approach as for the TTF gas price earlier. First, we perform a short data exploration analysis, to see what the general characteristics and dynamics in the dataset are. Then, we perform the preprocessing, assess the number of components, and which ones to use to improve the forecast. Finally, we add the new components of the exogenous variables to the original decomposition of the TTF gas price, make a forecast and compare performances.

To clarify the new forecasting flow, with the use of exogenous variables, we slightly adapt fig. 15a to represent the new situation. It is shown in fig. 38.

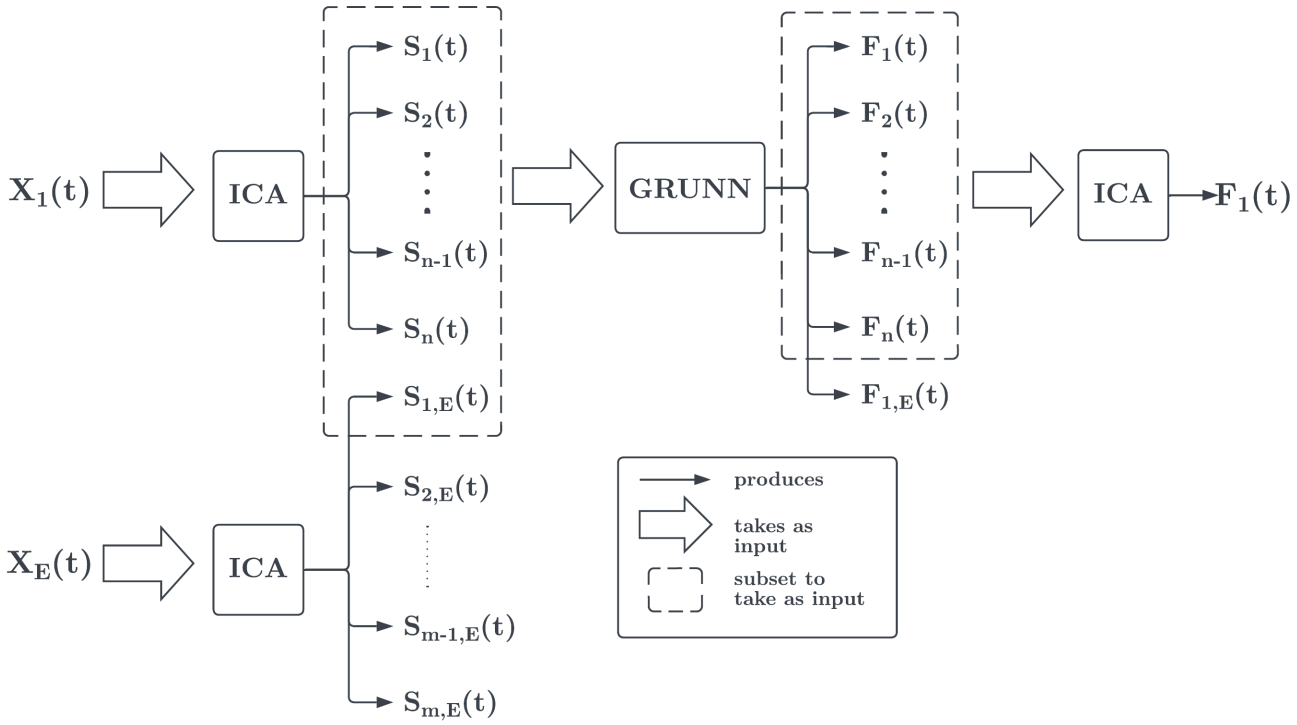


Figure 38: Data flow through the model including exogenous variables.

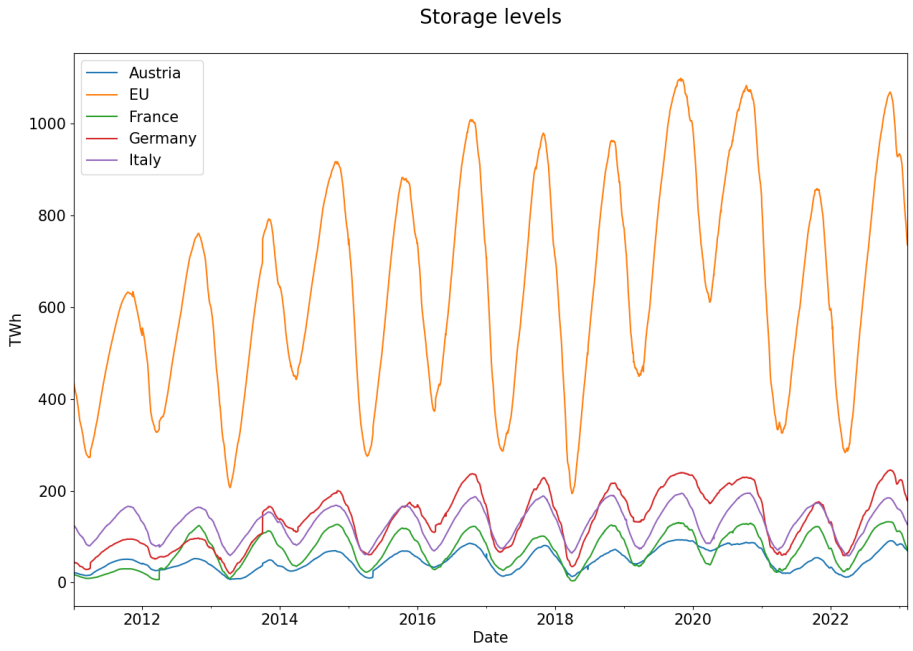
As we observe from fig. 38, we preprocess two data series. $X_1(t)$ is the original month-ahead TTF gas price, which is our ultimate goal to forecast. Besides this target variable, we also preprocess $X_E(t)$, which denotes the time series of any exogenous variable. This way, we get two sets of components $\{S_1(t), ..., S_n(t)\}$ and $\{S_{1,E}(t), ..., S_{m,E}(t)\}$, of which the components are independent within their decomposition. Out of the components derived from the exogenous variable $\{S_{1,E}(t), ..., S_{m,E}(t)\}$, we pick a subset of components, $\{S_{1,E}(t)\}$ in the case of fig. 38, to add to the original TTF price decomposition $\{S_1(t), ..., S_n(t)\}$. This new set of components forms the input for the GRUNN model. The GRUNN model then makes a forecast for each of the components $\{F_1(t), ..., F_n(t), F_{1,E}(t)\}$ of which we only use the components associated with the month-ahead TTF price to produce the final price forecast. To produce the final price forecast, we reverse the ICA by mixing and adding the mean.

4.6.1 Gas storage: exploration

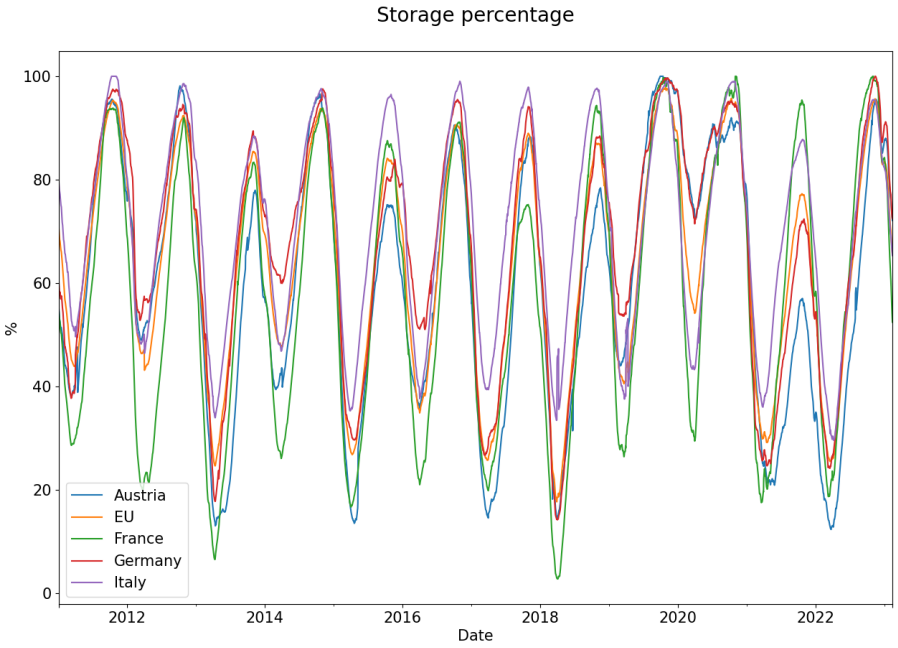
The storage level dataset we use contains 22,135 entries, it contains daily storage levels of 5 different regions over the period of 2011-01-01 until 2023-02-13, provided by the ENTSOG. The 5 regions are: Austria, France, Germany, Italy and the EU. Note that the EU storage levels are not equal to the sum of the 4 other regions. The storage levels are recorded in two different formats: the storagelevel in terawatt-hour (TWh), and the percentage of storage which is filled. Both the level and percentages for the entire time span of the dataset are plotted in fig. 39.

We observe strong periodicity in the storages for all regions. As mentioned before, storages are filled in summer, and emptied during winter. The periodicity of the storage data is further endorsed by the autocorrelation plots shown in fig. 40. Comparing the ACF plots of the different countries to each other, we observe that the storage series of Italy is very close to a perfect cosine. This indicates there is a very strong seasonality effect present in the Italian storage facilities. On the other hand, the ACF plot of Germany shows a different image. The autocorrelation in the first peak (circa sample no. 350) is higher than in the second peak (circa sample no. 750). Besides this slight effect in declining autocorrelation we also observe a general downward trend in the ACF plot of Germany. This might indicate that the seasonality in Germany has some decline to it. This decline might have to do something with the Russo-Ukrainian conflict, the increasing supply of renewable energy, or the increased import of LNG, to name a couple important influences on the European gas market.

Some basic statistics about the gas storage data are summarised in table 16. From table 16 we deduce that all storage series are stationary, since the ADF test produces results far below the common threshold of 0.05. This aligns with the image in fig. 39a and fig. 39b, where we do not observe obvious changes in underlying processes or regimes.



(a) Storage levels for selected regions.



(b) Storages percentages for selected regions.

Figure 39: Level and percentages of storage facilities for selected regions.

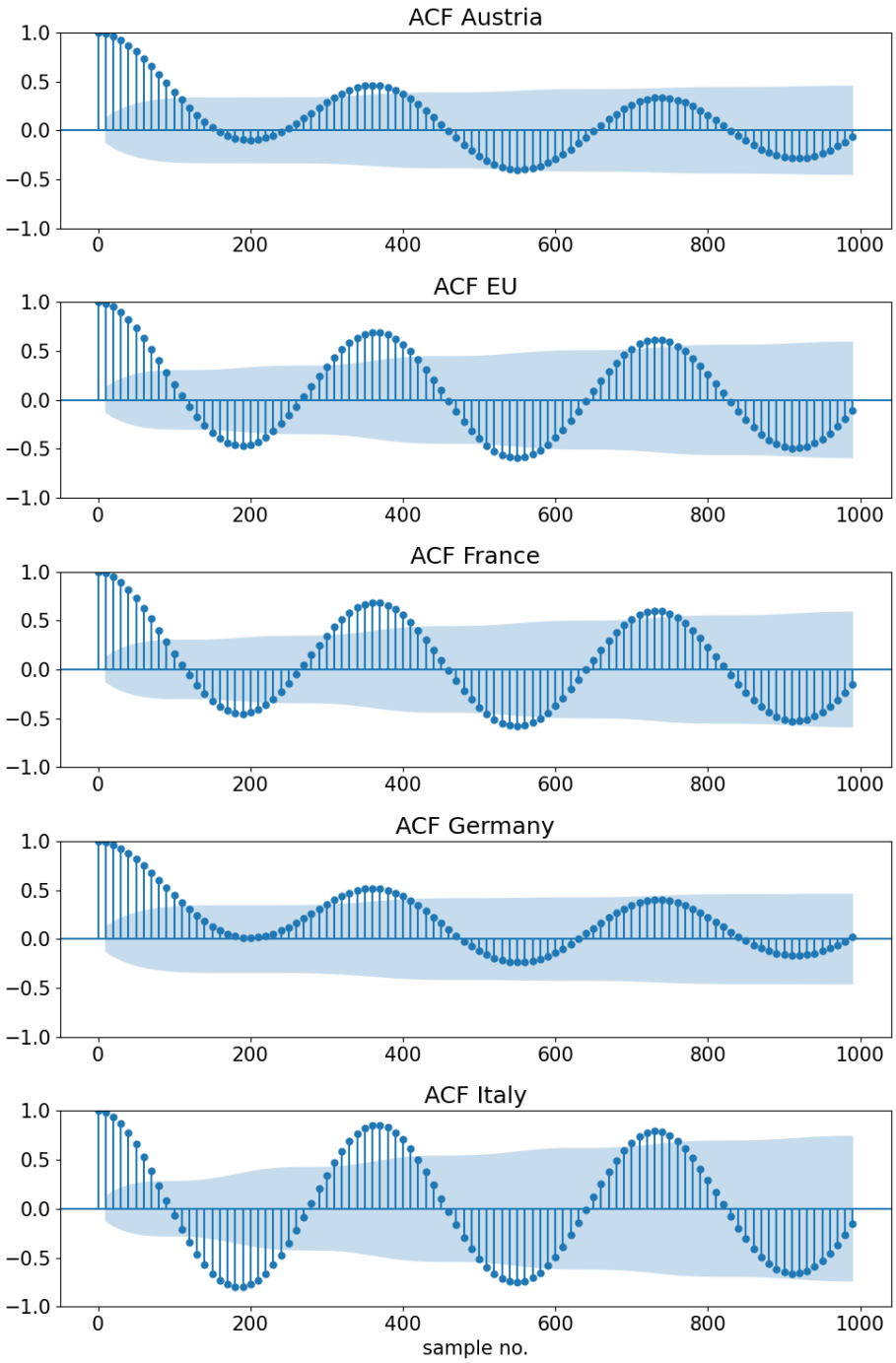


Figure 40: ACF plots for all storage regions recorded.

	mean	median	std	min	max	ADF-test score
Austria	48.42	46.03	23.01	7.01	93.32	2.28e-07
EU	643.48	631.22	232.67	193.88	1098.85	1.55e-14
France	70.95	71.41	37.40	3.79	132.66	1.91e-13
Germany	135.87	136.07	60.47	19.81	245.44	2.79e-05
Italy	129.54	132.66	36.77	58.64	195.01	2.13e-18

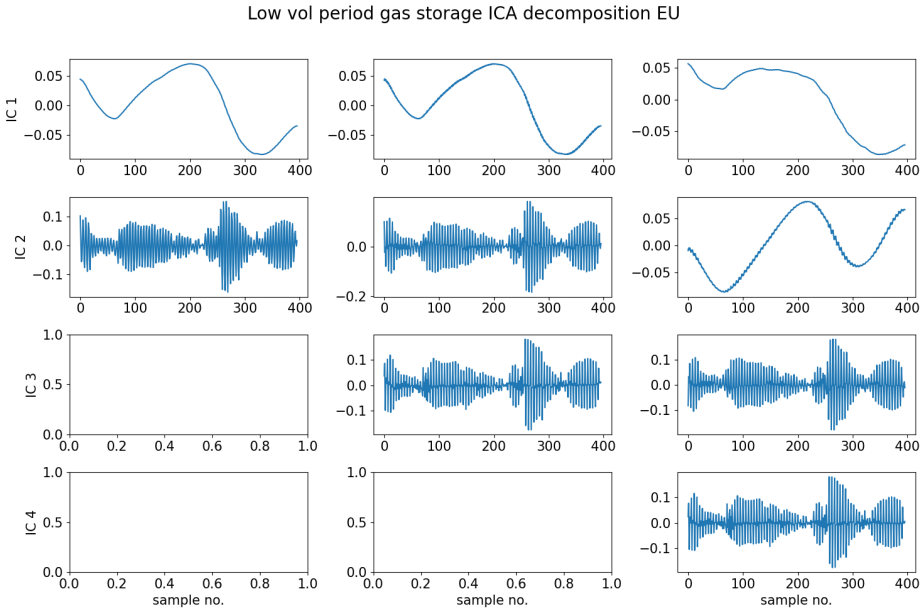
Table 16: Summary statistics for storage data.

To use the storage variables for improving the forecast of the TTF price, we need to take a subset of the data, which matches the regimes we defined in section 4.3.4. This means we only use the data set with storage information for

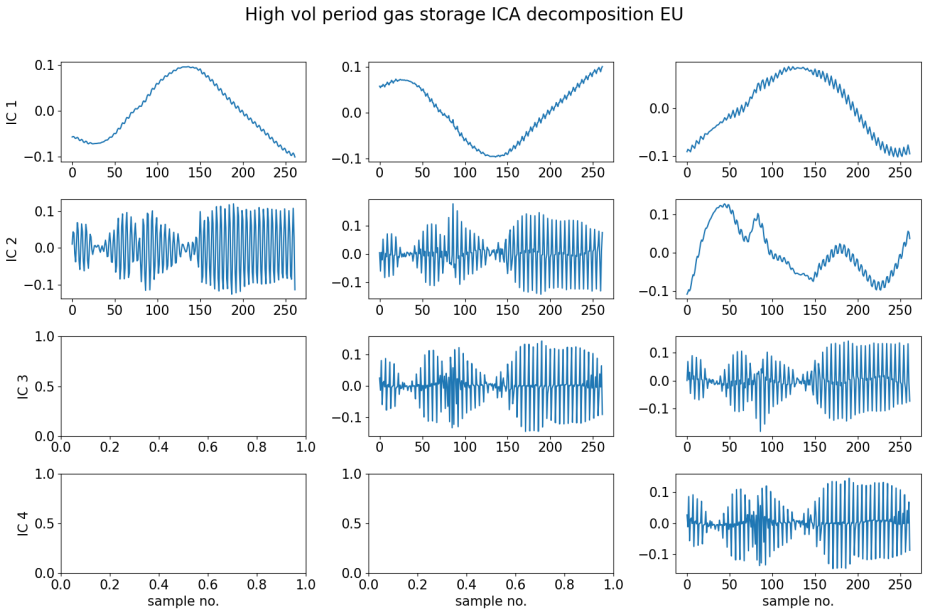
the period of 2020/01/02 - 2021/07/16 to improve the forecast of the low volatility regime, and of 2021/09/15 - 2022/09/19 for the high volatility regime.

4.6.2 Gas storage: preprocessing

We observe that the signal is close to a pure low frequency component, together with a negligible noise term. For now, we focus on EU storage levels since this data spans the widest region, and thus impacts the TTF price the most. The ICA decompositions for the high and low volatility regimes are plotted in fig. 41.



(a) ICA decomposition for EU storage levels during the low volatility period.



(b) ICA decomposition for EU storage levels during the high volatility period.

Figure 41: EU storage level ICA decompositions for the low and high volatility regime periods.

From fig. 41 we observe that as little as 2 components are enough to describe the EU storage level. By adding an additional component to the decomposition, we only add an extra high frequency noise term. When adding the storage components to improve the forecast, we will only add the low frequency component, since the high frequency component does not bear any relevant information. Besides this, our goal with adding the storage components is to improve the forecast of the slow component, not the fast components.

4.6.3 Gas storage: forecasting

We incorporate the components of the storage which we need for forecasting by concatenating them to the source signal matrix of the TTF price, as schematically shown in fig. 38. This way we add another dimension to the feature space, which is not necessarily independent from the others. It does provide an extra dimension which is not derived from the same dataset though.

	Regime	
Model	Low	High
no exogenous variables	0.52%	2.11%
storage 2 components	0.71%	3.90%
storage 1 slow component	0.36%*	2.96%

*best over three runs.

Table 17: MAPE performance measures of including gas storage data into model.

Note that for the no exogenous variables benchmark we retrained a VMD-ICA-GRUNN model on the TTF dataset. We see a slight improvement in the low volatility regime forecast when adding a single low frequency component of the storage decomposition to it. For the high volatility regime, adding the storage data did not improve the forecasting performance compared to the best GRUNN mode presented in previous sections. One explanation of these results could be that in the low volatility regime, the low frequency component is more dominant, since there is less volatility. This could explain why only the low frequency regime forecast benefited from adding the storage component, since it aimed at improving the low frequency component forecast.

4.6.4 NBP month-ahead price: exploration

As mentioned earlier, we use the NBP month-ahead price to improve the forecast on the high frequency components in the ICA decomposition. The NBP price is more volatile than the mainland European price. The NBP price dataset also contains transactions, 1,831,114 in total. The transactions contain the same data fields for each trade as in the original TTF dataset. We applied the same data parsing, filtering and rolling methods as we applied to the TTF price dataset, to arrive at a daily VWAP for the UK NBP price. Figure 42 shows the NBP VWAP in comparison to the TTF VWAP.

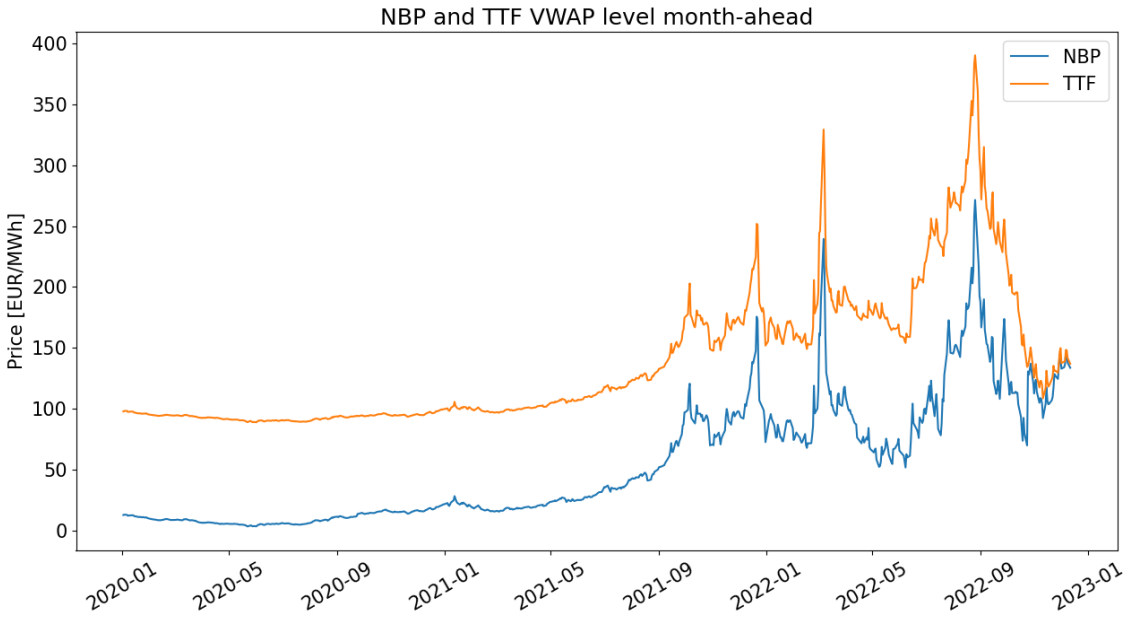


Figure 42: NBP and TTF month-ahead price levels.

From the plot we observe that price levels for NBP have the highest peaks on roughly the same moments as the TTF price. However, the price level in general seems to be lower. The more volatile behaviour of the NBP price is substantiated by the summary statistics in table 18.

	NBP pence/therm	NBP EUR/MWh	TTF EUR/MWh
mean	138.03	55.13	149.38
median	75.99	30.61	141.90
std	127.17	51.88	62.43
min	8.75	3.33	39.95
max	672.90	271.43	390.22

Table 18: Summary statistics on the month-ahead NBP and TTF VWAP levels.

The TTF statistics in table 18 have been taken directly from table 5. The NBP contracts are traded in a different unit than the TTF contracts. The UK NBP trading unit is pence per therm. A therm is a unit of energy and equals 29.3071 kWh. To arrive at the EUR/MWh price of NBP, we divide by 2.93071 and multiply with the daily EUR/GBP exchange rate. A couple of things are worth mentioning in table 18. First, the mean of NBP data is higher than the median, which indicates the dataset is skewed to the right. This is an indication of fat tails on the right end the distribution, something which we observed from fig. 42 as well, with its high peaks. Second, the standard deviation of the pence/term NBP dataset is approximately two times higher than the standard deviation of the TTF dataset. This effect still persists after conversion, only less obvious. We still observe that the standard deviation of the NBP price in EUR/MWh is relatively two times as high as for the TTF price, compared to the mean. NBP price standard deviation is almost equal to the mean, whereas the standard deviation of the TTF price is close to half of the mean. Lastly, the difference in maximum and minimum price between the TTF and NBP price is notable. The maximum price the NBP price attains is higher than the maximum TTF price. All together this table is a first indication that the NBP dataset might introduce some new characteristics to the input set of the forecasting model.

4.6.5 NBP month-ahead price: preprocessing & forecasting

Comparing the first two components of a 5-component ICA decomposition of both the TTF price and the NBP price is a further indication that NBP might bring some new dynamics to the input space. As we notice in fig. 43, the trend component of the NBP decomposition (IC1) has a more pronounced periodicity in it than the trend component of the TTF decomposition. IC2 of the TTF and NBP price are of a totally different nature, where the TTF IC2 is still slow in nature, whilst the NBP IC2 is already higher frequency.

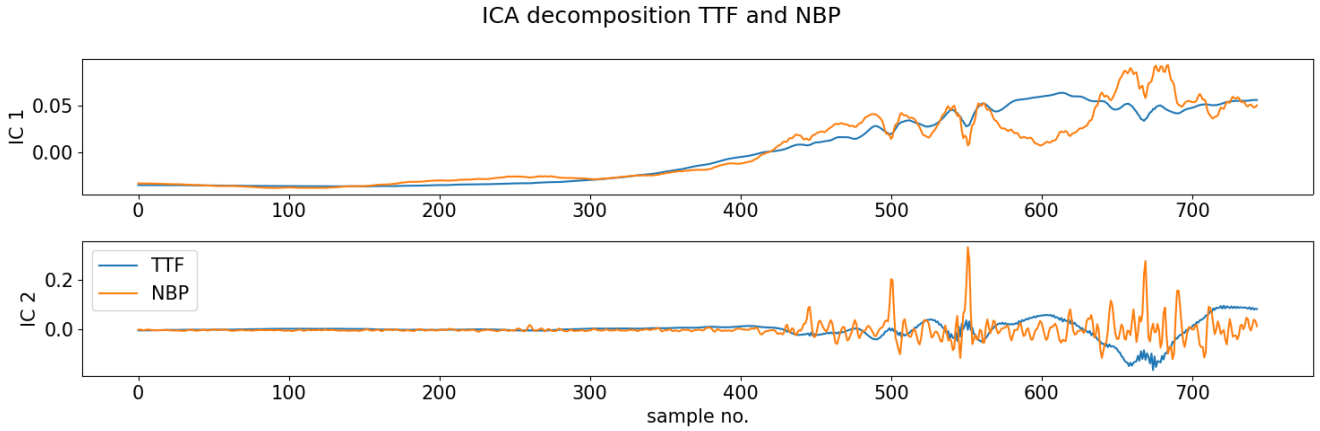


Figure 43: First two low frequency components of TTF and NBP ICA decompositions.

We incorporate the ICA decomposition of the NBP price into the input space for the GRUNN models on the same regime definition used before, to be able to compare results. Meaning, 2020/01/02-2021/07/16 for the low volatility regime and 2021/09/15-2022/09/19 for the high volatility regime. For these regimes, different numbers of components are tested for the NBP decomposition, but not for the TTF decomposition, as to be able to compare results to the original forecasts made. We synchronise the TTF dataset to the NBP dataset to account for e.g. different bank holidays.

To be able to compare results, we retrain two GRUNN models on the new TTF data set. These are the benchmarks for the high and low volatility regimes. To determine which components are best to add, we try several different configurations. For the low volatility regime we try to add a subset of components from a decomposition with 3, 4 or 5 components. For the high volatility we do the same for decompositions of 6, 7 or 8 components. The subsets of

components we add to the input space is the same for the low and high volatility regimes. We either add the lowest frequency component, or the lowest 2; the highest frequency component, or the highest 2; or we add all components to the input space. The forecasting performances for the TTF gas price are shown in table 19, for different model inputs using the NBP gas price.

NBP components added	total NBP components			total NBP components		
	3	4	5	7	8	9
lowest	0.48%	2.36%	1.56%	14.40%	7.30%	4.92%
lowest 2	3.62%	6.05%	2.00%	6.25%	6.15%	6.40%
highest	2.65%	0.71%	0.87%	6.51%	4.74%	4.02%
highest 2	3.38%	0.48%	0.58%	7.24%	7.51%	5.17%
all	2.42%	6.42%	6.31%	4.03%	3.29%	7.18%
	low volatility regime			high volatility regime		
benchmark	0.66%			4.75%		

Table 19: TTF price forecasting performance comparison for different number of NBP components added to the GRUNN input space.

As we observe from table 19, adding NBP data to both the TTF models for the high and low volatility regimes improves the forecast compared to the benchmark. There are multiple configurations which perform better than the benchmark. For the low volatility regime, adding the slowest component of a 3 component NBP decomposition and adding the fastest 2 components of a 4 or 5 component NBP decomposition improve the benchmark performance. For the high volatility regime adding all components of a 7 or 8 component NBP decomposition, and adding the fastest component of an 8 or 9 component decomposition improve the benchmark performance. Note that these results have been produced by using the same decompositions of the NBP price, only different subsets of those decompositions. Interestingly, it does not always hold that if the forecast is improved by adding a single component, this means that adding all components improves the forecast as well. For example, adding only the high frequency component of a 9 component NBP decomposition improves the forecast compared to the benchmark, while adding all components does not. We might expect this, since the component which improves the forecast is also present when we add all of the components. However, adding all components provides for a more complex input space, possibly corrupting the fitting process.

Note that an exact comparison between the best models mentioned earlier, with MAPEs of 0.38% and 1.4% for the low and high volatility regimes, is not possible due to the different data sets, even though the datasets only differ less than 10 data points. Results do indeed show what we expected, which is an improvement in forecasting performance by adding selected components of the NBP decomposition.

What is interesting to note though, is that the benchmark MAPEs and the best TTF trained MAPEs do differ quite a bit. The benchmark models trained on the new dataset (without UK bank holidays) produce substantially worse MAPEs of 0.66% and 4.75% compared to 0.38% and 1.4% on the original regimes. The exact reason behind this is hard to guess, but it might have something to do with the decompositions being quite different, because of the missing samples. As we have seen earlier, datasets with different intervals (albeit only a couple samples difference) can produce vastly different decompositions, which might contribute to the difference in performance.

5 Conclusion

In this thesis, we examined the problem of forecasting the price of European month-ahead TTF gas contracts. Up until now, only a small number of papers have been written about forecasting European gas prices. We only found a limited number, two of which are: Berrisch and Ziel (2022), who made a traditional time series forecast for the TTF month-ahead and day-ahead price and Ram et al. (2019), who implemented a plain ANN to forecast the gasprice at multiple EU hubs, including the TTF price. The fact that so little papers have been published likely has to do with the fact that the European gas market only recently reached a more developed state, due to the opening of multiple gas hubs and the decoupling of the market from long-term, pipeline supplied, gas contracts. Fortunately, many papers have been published on the American gas market. Borrowing from that experience we began our work on forecasting the EU gas price. During the literature review, it became clear that preprocessing the data and using a hybrid combination of multiple models could benefit the forecasting performance.

To this end, we chose to use a combination of Variational Mode Decomposition and Independent Component Analysis to preprocess the data. By inspecting the decompositions resulting from the preprocessing, we decided to

split the data into two volatility regimes, a high volatility regime, and a low volatility regime. The data in itself provided an interesting topic for study, since it exhibits some very interesting, new dynamics, not yet described in earlier papers. Under the influence of a recovering economy from the COVID-19 pandemic and the Russo-Ukrainian conflict, prices surged to levels never seen before.

To make a forecast using the preprocessed data, we used a combination of a Gated Recurrent Unit Neural Network and Support Vector Regression, as proposed by E et al. (2019). However, the forecast produced by the SVR integration did not outperform the forecast where we integrated the GRUNN forecasts directly to form a single price. That is why for the most part of the research we integrated the component forecasts directly to a single price forecast by mixing them again, using the mixing matrix produced by ICA instead of the SVR approach.

Next, we researched the influence of our decisions to split the dataset into volatility regimes, and to preprocess the data before making any forecasts by defining three benchmark models. Both approaches improved the performance, and also outperformed the naive forecast, where today's price is tomorrow's forecast.

For the final part of the thesis we researched the influence of adding the decompositions of certain exogenous variables to the feature space of the GRUNN. The exogenous variables are specifically selected to improve on either the low frequency components of the model, or the high frequency components. To improve model performance on the low frequency components, gas storage information was added, which has a naturally low frequency effect in it. We decided to add NBP gas pricing data to improve the forecast of the high frequency components. The NBP gas price is more sensitive to shocks in the market and weather events, making it generally more volatile than the European TTF price.

By adding the gas storage data, we observed an improvement in performance for the low volatility regime, when adding only the lowest frequency component of the storage decomposition to the input space of the GRUNN model. For the high volatility regime, we did not observe any improvement in performance by adding gas storage data. We can interpret this result by noting that the low frequency component is more dominant in the low volatility regime, since the regime is less volatile. This could be one of the reasons that forecasting performance improved only for the low volatility regime. Preprocessing NBP month-ahead prices and including them as an exogenous variable improved the performance of the forecasting model for both the low volatility and high volatility regimes.

Interesting topics for further research could be the inspection of the ICA decompositions in light of the regime definition. Some clear differences in decompositions are observable when shifting the regime interval by a couple of sample points. Using this information, we might be able to detect regime changes in the future, and act accordingly. Also, stability in convergence of the ICA algorithm are tied to this topic. Some regime lengths produced more stable decompositions than others, sometimes varying only a couple of sample points.

Extensions of this thesis could be researching the influence of other exogenous variables than gas storage or NBP month-ahead prices. Weather analyses or electricity prices for example might provide other good exogenous variables for improving the high frequency component forecasts. Inflation or economic cycles might improve the low frequency component forecasts even further.

Finally, adapting the GRUNN models to output a probabilistic forecast would be very interesting, since probabilistic forecasts provide more information on the distribution of the forecasts. This would allow for rigorous risk analyses, and different performance measures to research the forecasts across the entire probabilistic spectrum.

A Appendices

A.1 Decomposition comparison regimes

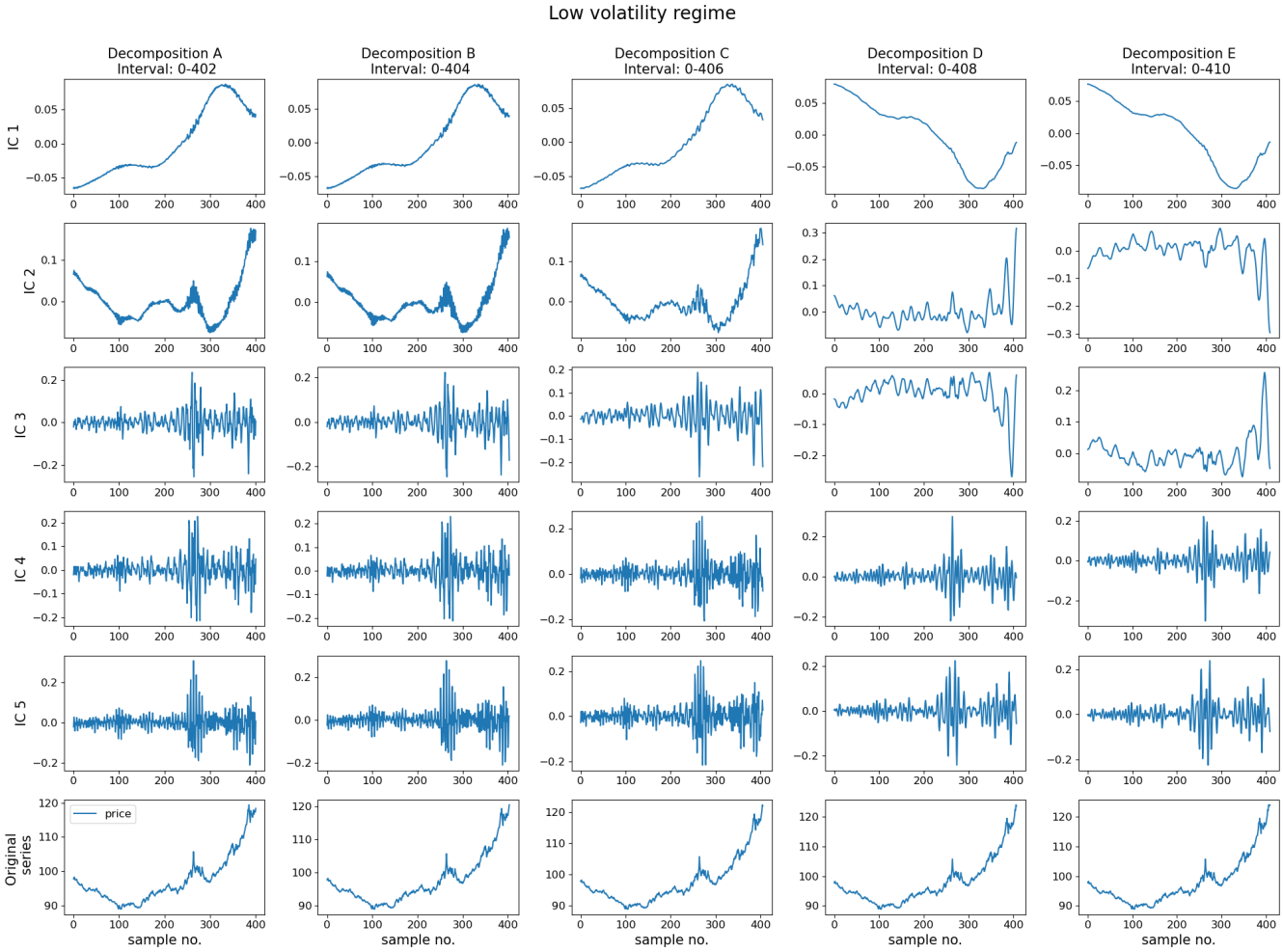


Figure 44: 5 independent component analyses, for 5 different low volatility regimes.

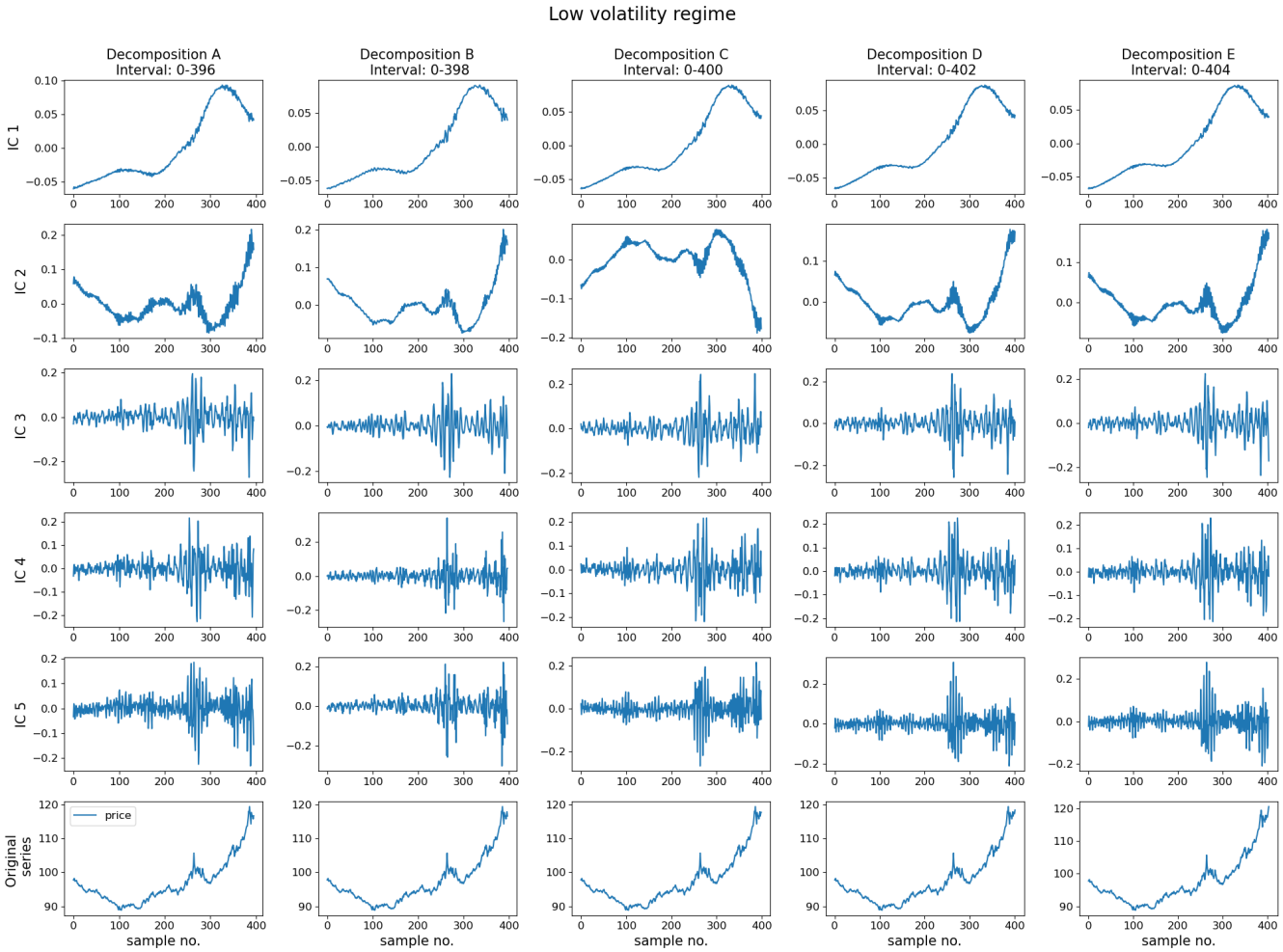


Figure 45: Shifted regime ends to show the best decompositions for regimes with 5 independent components.

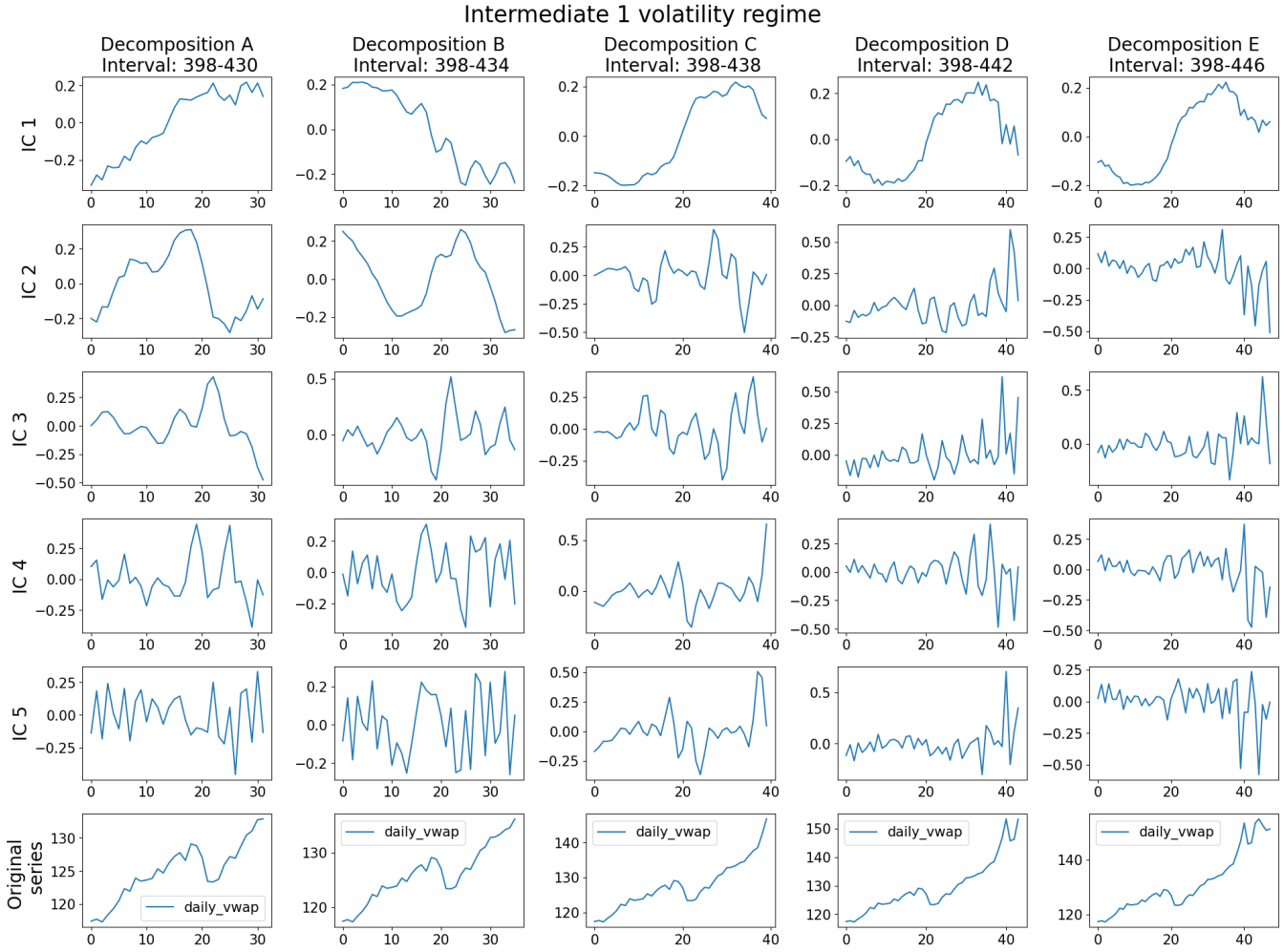


Figure 46: First intermediate regime decomposition comparison

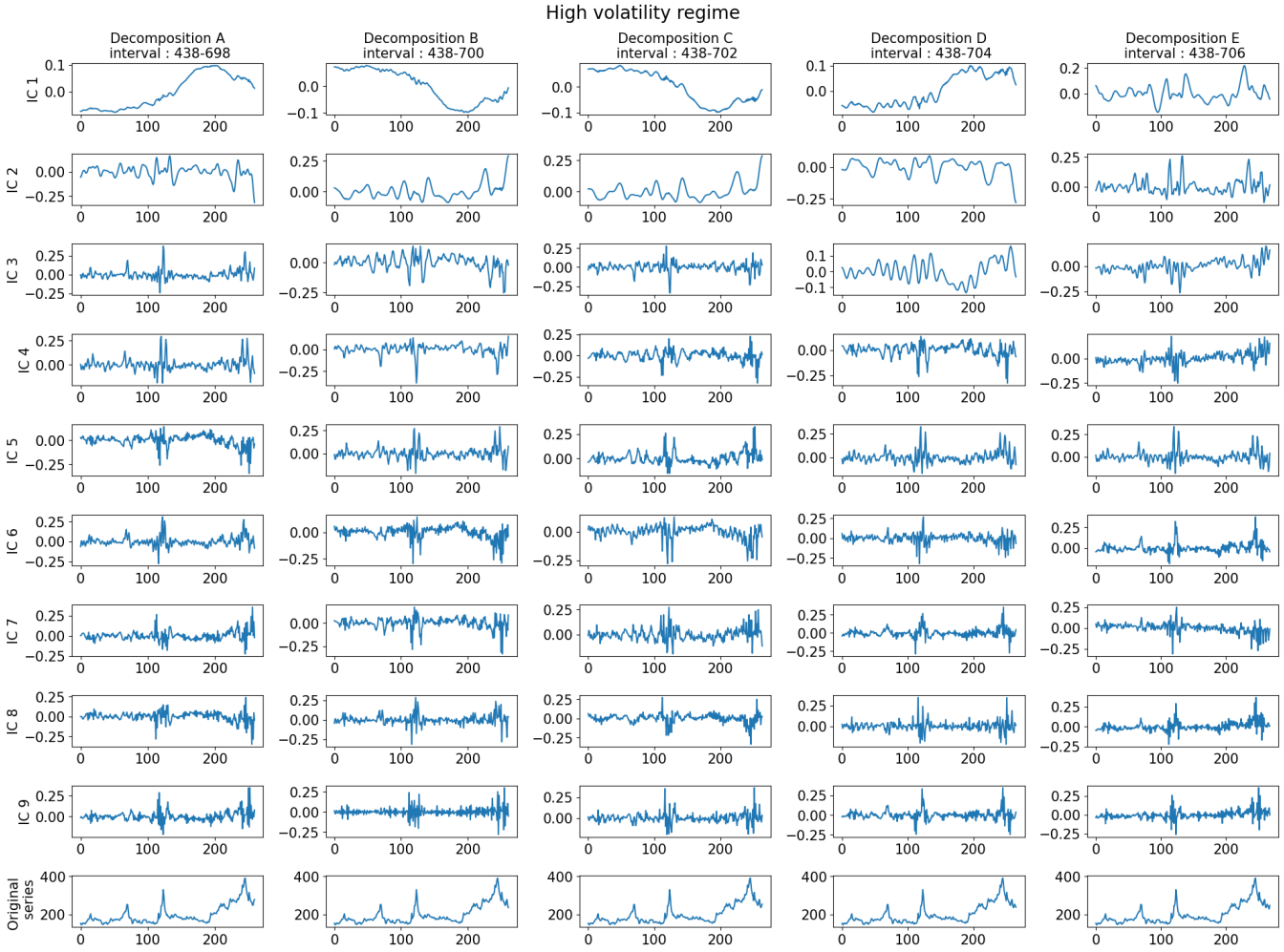


Figure 47: High regime decomposition comparison

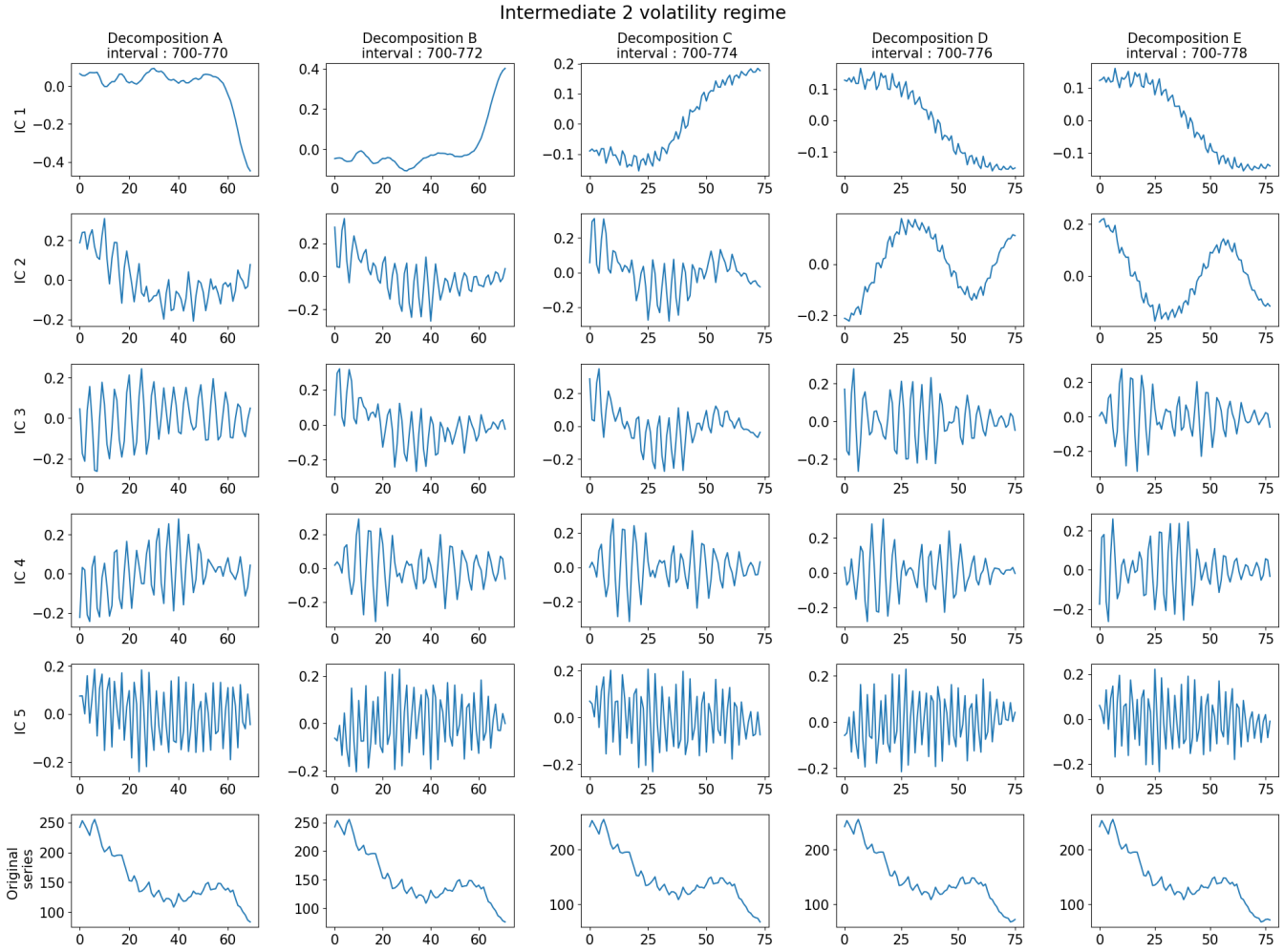


Figure 48: Intermediate 2 regime decomposition comparison

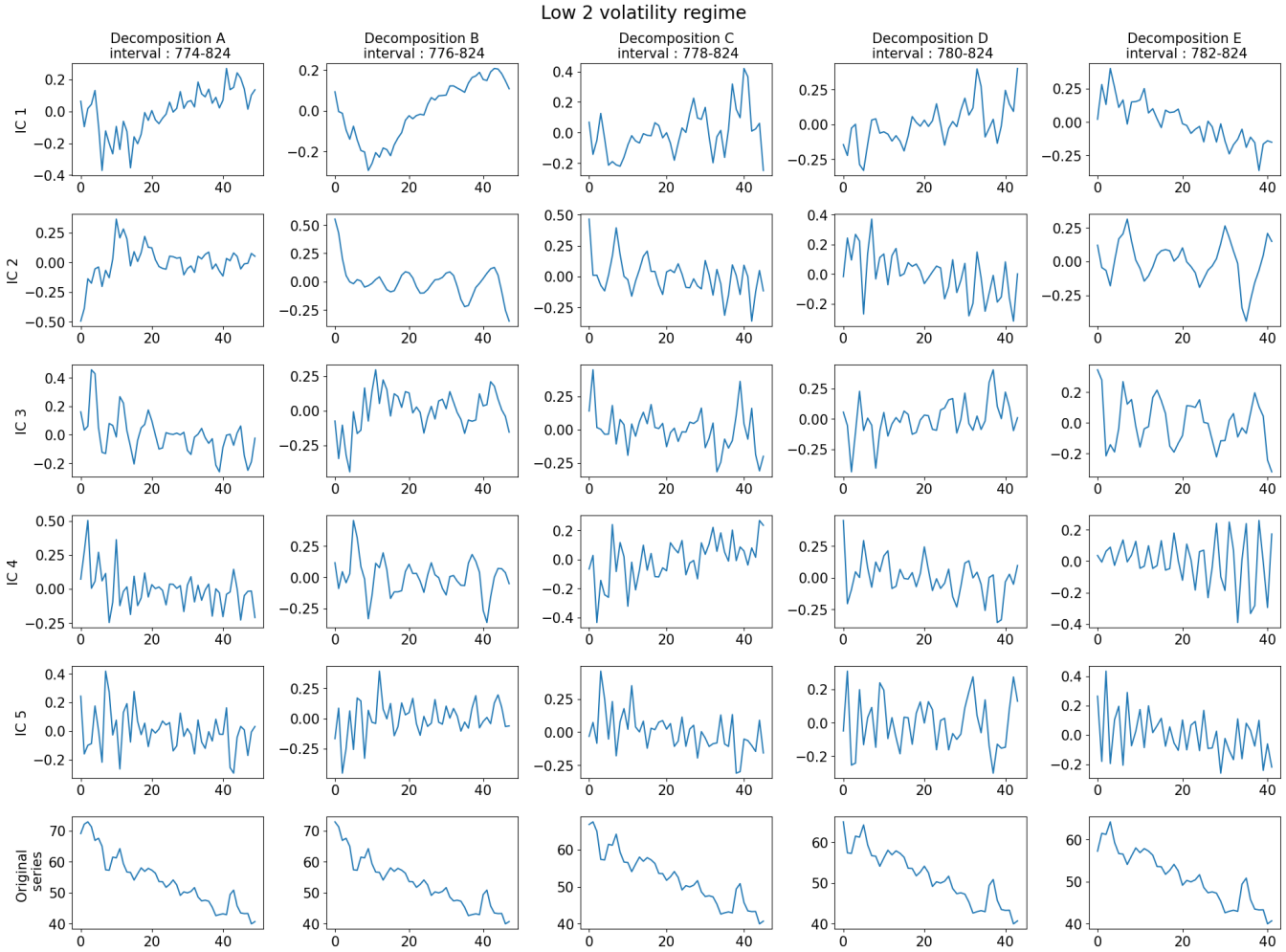


Figure 49: Low 2 regime decomposition comparison

A.2 Decomposition comparison different numbers of components

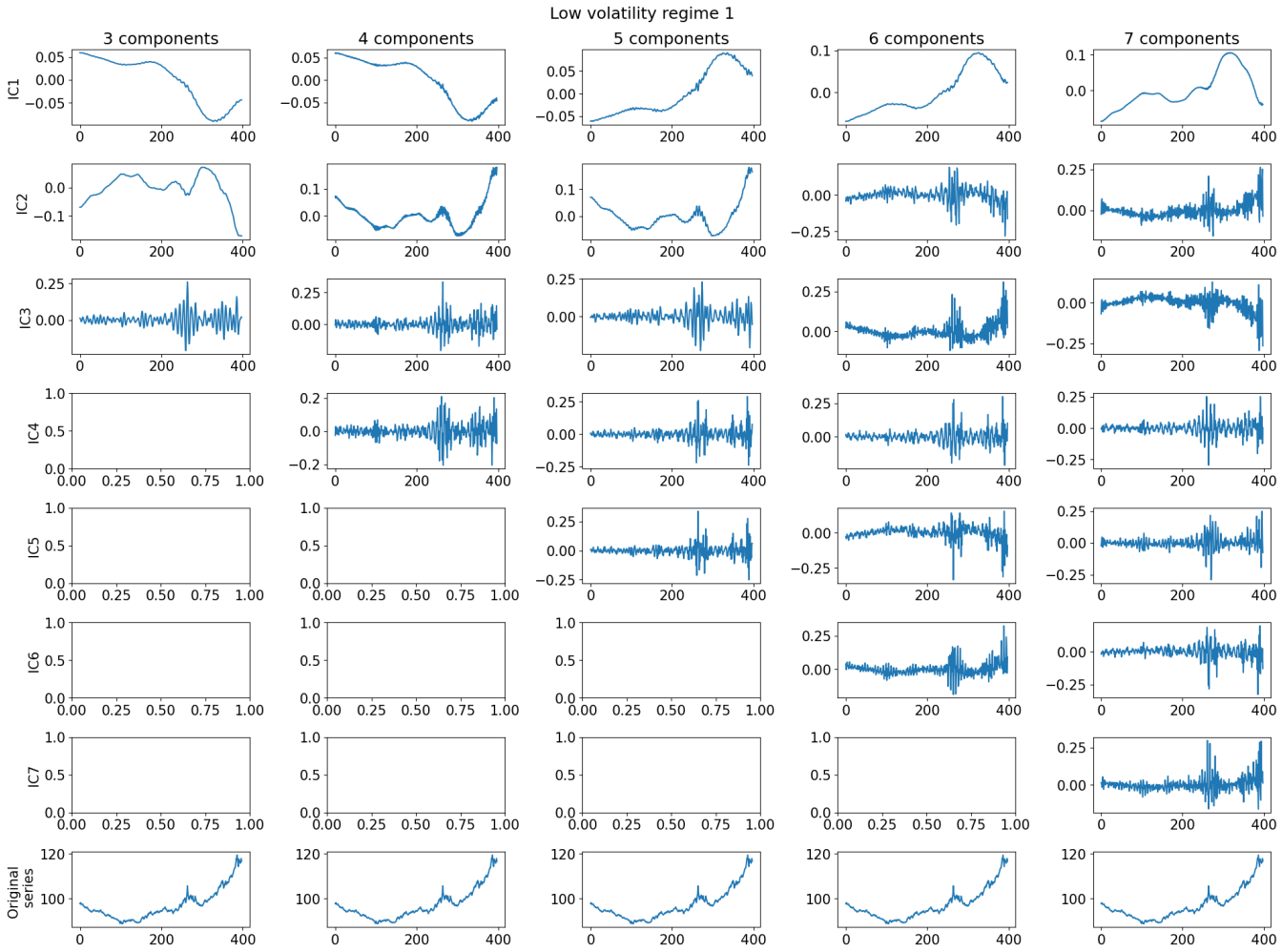


Figure 50: Low volatility regime ICA decomposition for different number of components.

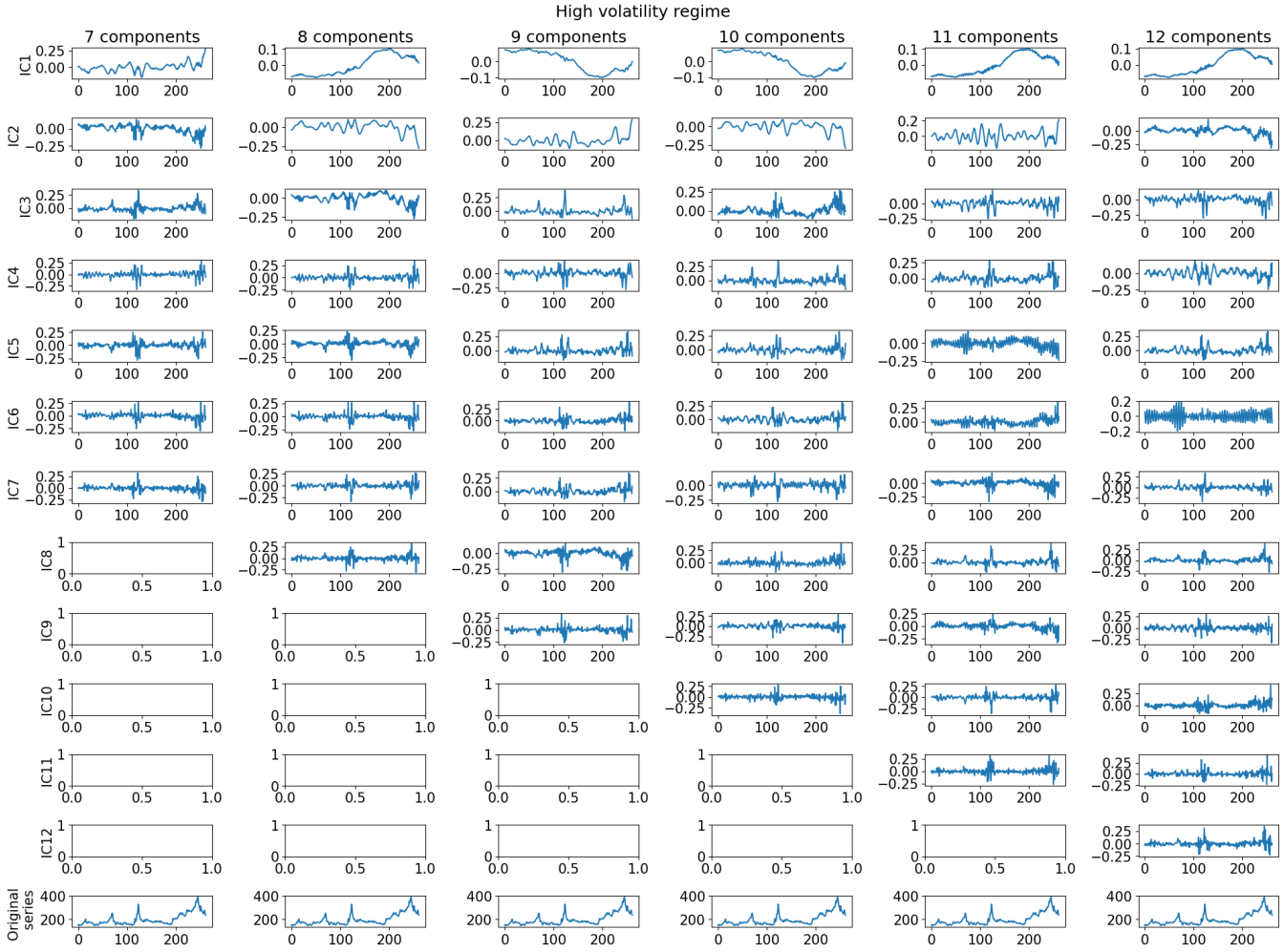


Figure 51: High volatility regime decompositions for different numbers of components

A.3 Forecasting comparison different regimes

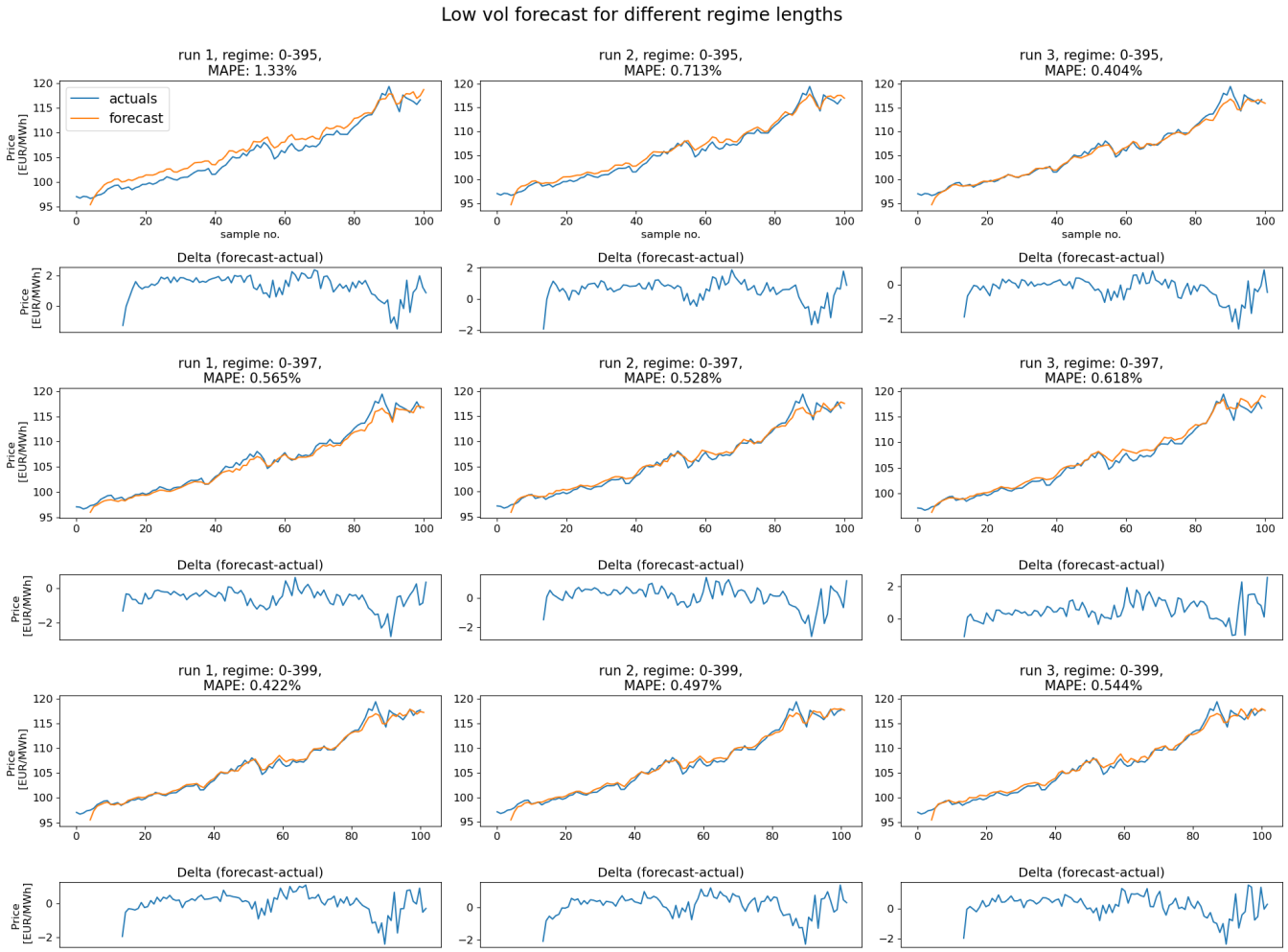


Figure 52: Low volatility forecasting performance over multiple runs for different regime definitions, with 5 component ICA.

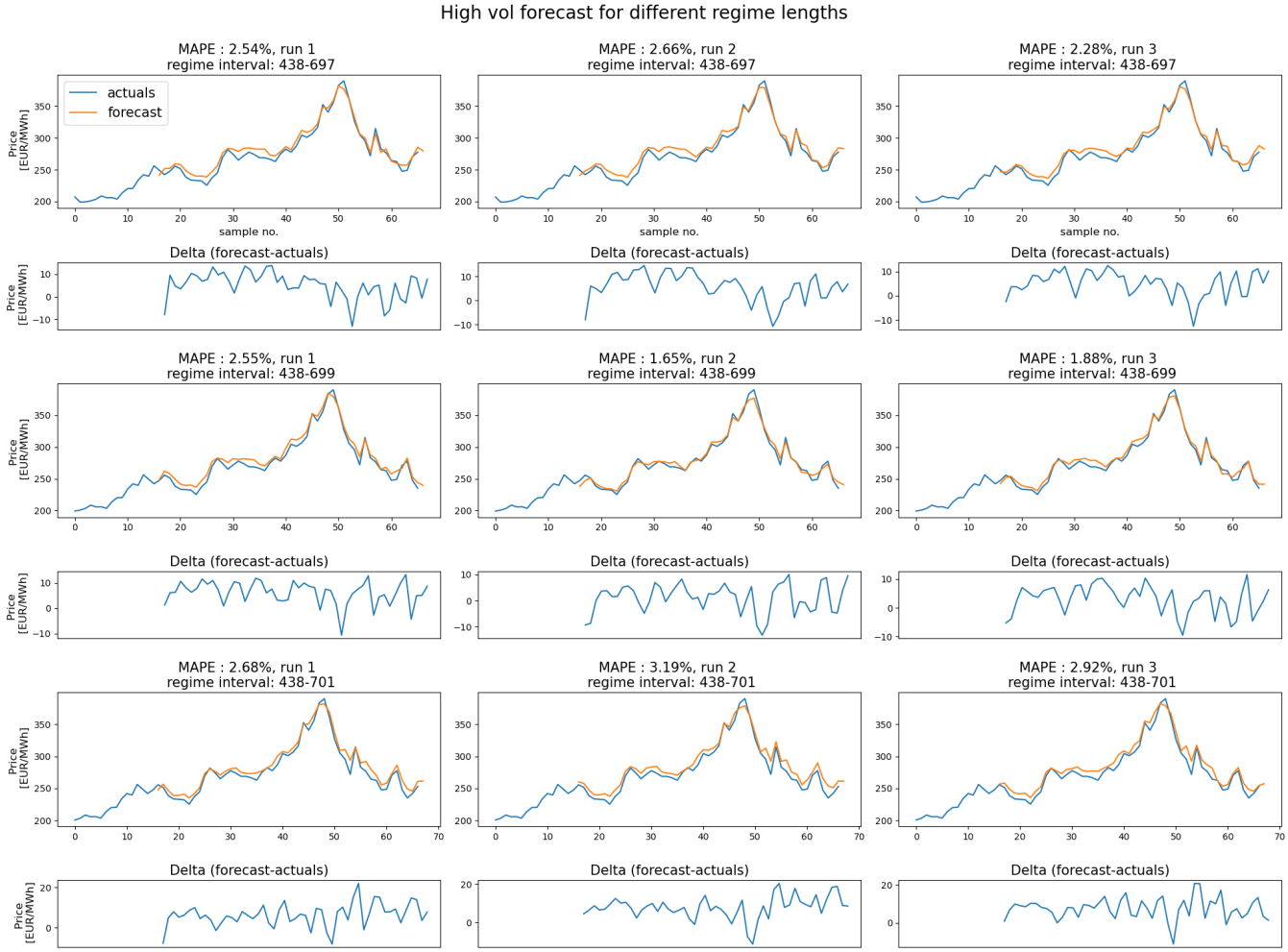


Figure 53: High volatility forecasting performance over multiple runs for different regime definitions, with 8 component ICA.

A.4 Forecasting comparison different numbers of components

Figure 54: Low volatility forecasting performance over multiple runs for different number of components.

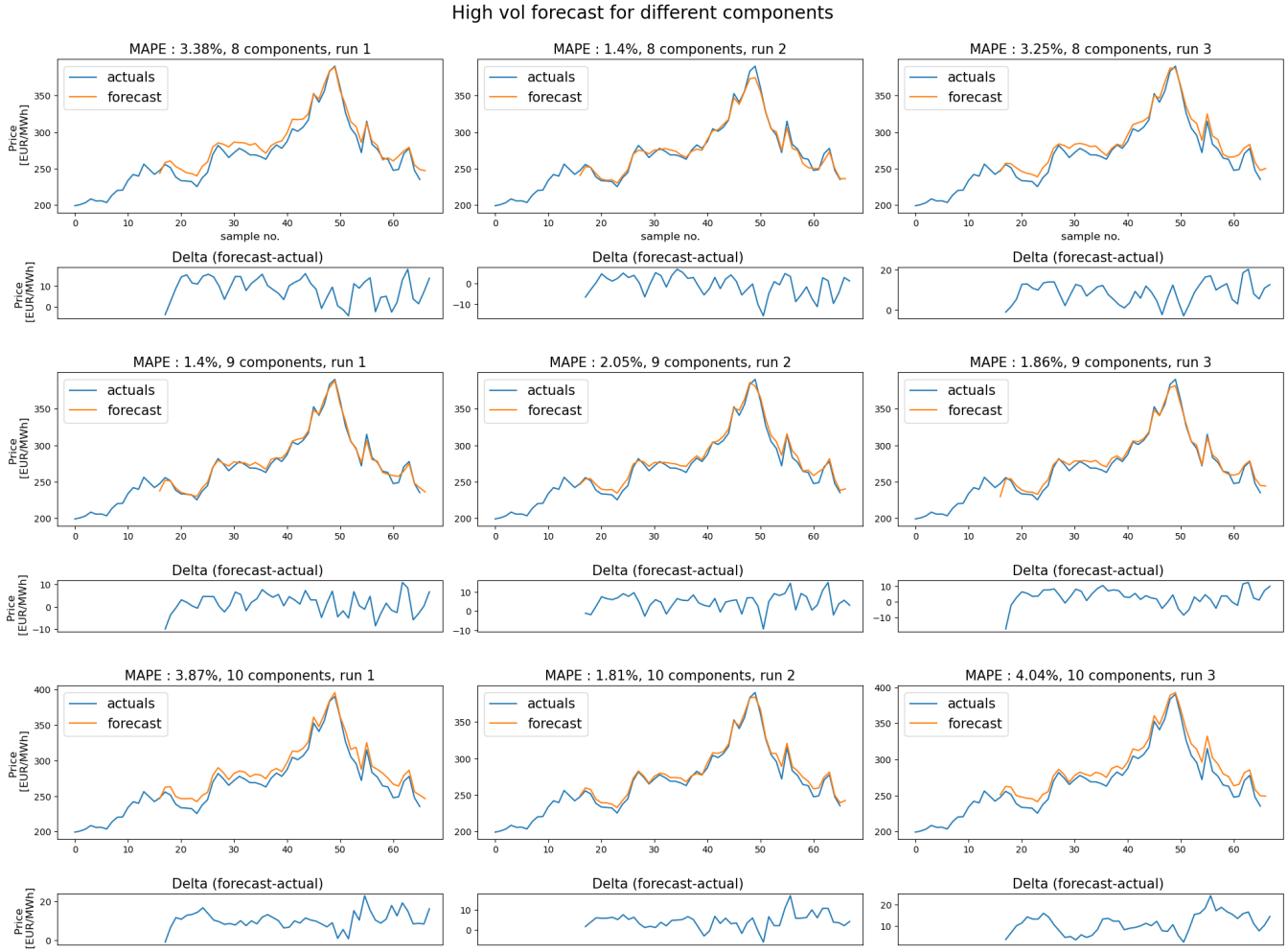


Figure 55: High volatility forecasting performance over multiple runs for different number of components

A.5 Hyperparameter tuning results

	Low volatility regime	High volatility regime
Number of GRU cells	MAPE score	MAPE score
64	0.64%	4.24%
32	0.38%	1.4%
16	0.55%	5.51%
8	2.30%	29.0%
4	2.75%	28.9%

Table 20: All forecasting performances for the number of GRU cells tuning.

Low volatility regime		High volatility regime	
Rolling window size	MAPE score	Rolling window size	MAPE score
1	2.98%	-	-
2	0.59%	-	-
3	0.38%	10	2.61%
7	1.09%	15	1.4%
10	1.22%	20	2.54%
12	1.68%	25	2.65%
20	1.97%	30	5.58%

Table 21: All forecasting performances for the rolling window size tuning.

	Low volatility regime	High volatility regime
Network architecture	MAPE score	MAPE score
One layer, 32 cells	0.38%	1.4%
Two layers, 32 cells	0.50%	6.88%
Three layers, 32 cells	0.61%	7.90%
Two layers, 16 cells	1.00%	6.69%
Three layers, 16 cells	0.58%	7.73%

Table 22: All forecasting performances for neural network architectures.

	Low volatility regime	High volatility regime
Train-val-test split	MAPE score	MAPE score
0.7-0.05-0.25	0.38%	1.4%
0.50-0.05-0.45	9.49%	11.7%
0.60-0.10-0.30	8.29%	3.28%
0.70-0.15-0.15	0.74%	19.3%
0.50-0.30-0.20	12.1%	35.6%
0.80-0.05-0.15	0.89%	4.12%

Table 23: All forecasting performances for different train, validation and test sets.

A.6 Full ICA decomposition TTF and NBP

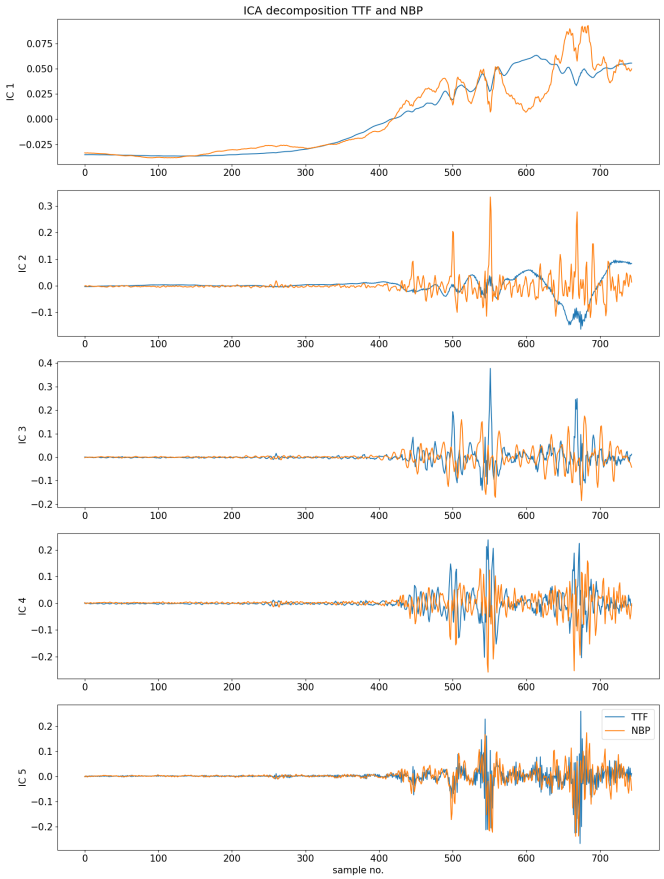


Figure 56: Independent component comparison for the TTF and NBP ICA decompositions.

Nomenclature

(E)EMD (Ensemble) Empirical Mode Decomposition

(TG)ARCH (Threshold Generalised) Autoregressive conditional heteroskedasticity

ACE Alternating Conditional Estimation

ADMM Alternate Direction Method of Multipliers

ANN Artificial Neural Network

AR Autoregression

ARIMA Autoregressive Integrated Moving Average

ARMA Autoregressive Moving Average

ARNN Autoregressive Neural Network

CNN Convolutional Neural Network

DA Day-ahead gas price

DLLR Dynamic Local Linear Regression

DWD Discrete Wavelet Decomposition

ETS Error, Trend, Seasonality model

EUA European Emission Allowances

FAR Functional Autoregressive

FFT Fast Fourier Transform

GRUNN Gated Recurrent Unit Neural Network

ICA Independent Component Analysis

IMF Intrinsic Mode Function

LLR Local Linear Regression

LOESS Locally estimated scatter plot smoothing

LSTM Long Short Term Memory

MA Month-ahead gas price

MA Moving average model

MAD Mean Absolute Deviation

ML Machine Learning

MLP Multilayer Perceptron

RF Random forest

SSM State Space Model

SVM Support Vector Machine

SVR Support Vector Regression

TBATS Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend component, Seasonal component model

TTF Title Transfer Facility

VAR Vector autoregression

VMD Variational Mode Decomposition

References

Abrishami, H. and Varahrami, V. (2011). Different methods for gas price forecasting. *Cuadernos de Economia*, 34(96):137–144.

Ataei, M., Bussmann, M., Shaayegan, V., Costa, F., Han, S., and Park, C. B. (2021). NPLIC: A machine learning approach to piecewise linear interface construction. *Computers & Fluids*, 223:104950.

Berrisch, J. and Ziel, F. (2022). Distributional modeling and forecasting of natural gas prices. *Journal of Forecasting*, 41(6):1065–1086.

Bopp, A. (1990). An analytical approach to forecasting natural gas prices. *Amercan Gas Association Forecasting Review*, 2:33–52.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Čeperić, E., Žiković, S., and Čeperić, V. (2017). Short-term forecasting of natural gas prices using machine learning and feature selection algorithms. *Energy*, 140:893–900.

Chen, Y., Xu, X., and Koch, T. (2020). Day-ahead high-resolution forecasting of natural gas demand and supply in Germany with a hybrid model. *Applied Energy*, 262.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Dragomiretskiy, K. and Zosso, D. (2014). Variational mode decomposition. *IEEE Transactions on Signal Processing*, 62(3):531–544.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola··, A., and Vapoik, V. (1996). Support Vector Regression Machines. *Advances in Neural Information Processing Systems*, 9:155–161.

E, J., Ye, J., He, L., and Jin, H. (2019). Energy price prediction based on independent component analysis and gated recurrent unit neural network. *Energy*, 189.

Geng, J. B., Ji, Q., and Fan, Y. (2017). The relationship between regional natural gas markets and crude oil markets from a multi-scale nonlinear Granger causality perspective. *Energy Economics*, 67:98–110.

Hamie, H., Hoayek, A., and Auer, H. (2020). Modeling Post-Liberalized European Gas Market Concentration—A Game Theory Perspective. *Forecasting*, 3(1):1–16.

Herault, J. and Jutten, C. (1986). Space or time adaptive signal processing by neural network models. In *AIP Conference Proceedings*, pages 206–211. AIP.

Herrera, G. P., Constantino, M., Tabak, B. M., Pistori, H., Su, J. J., and Naranpanawa, A. (2019). Long-term forecast of energy commodities price using machine learning. *Energy*, 179:214–221.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hotelling, H. (1931). The Economics of Exhaustible Resources. *Journal of Political Economy*, 39(2):137–175.

Hsieh, Y. (1990). Natural Gas Price Projection: A Methodology. *American Gas Association Forecasting Review*, Volume 2.

Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Snin, H. H., Zheng, Q., Yen, N. C., Tung, C. C., and Liu, H. H. (1998). The empirical mode decomposition and the Hubert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 454(1971):903–995.

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430.

Jin, J. and Kim, J. (2015). Forecasting natural gas prices using wavelets, time series, and artificial neural networks. *PLoS ONE*, 10(11).

Karabiber, O. A. and Xydis, G. (2020). Forecasting day-ahead natural gas demand in Denmark. *Journal of Natural Gas Science and Engineering*, 76.

Lu, H., Ma, X., Ma, M., and Zhu, S. (2021). Energy price prediction using data-driven models: A decade review.

Mishra, P. (2012). Forecasting natural gas price-time series and nonparametric approach. In *Proceedings of the World Congress on Engineering 2012*, pages 1–8, London.

Mouchtaris, D., Sofianos, E., Gogas, P., and Papadimitriou, T. (2021). Forecasting Natural Gas Spot Prices with Machine Learning. *Energies*, 14(18):5782.

Naderi, M., Khamehchi, E., and Karimi, B. (2019). Novel statistical forecasting models for crude oil price, gas price, and interest rate based on meta-heuristic bat algorithm. *Journal of Petroleum Science and Engineering*, 172:13–22.

Nguyen, H. T. and Nabney, I. T. (2010). Short-term electricity demand and gas price forecasts using wavelet transforms and adaptive models. *Energy*, 35(9):3674–3685.

Oja, E. and Zhijian Yuan (2006). The FastICA Algorithm Revisited: Convergence Analysis. *IEEE Transactions on Neural Networks*, 17(6):1370–1381.

Patrick Heather (2020). European Traded Gas Hubs: the supremacy of TTF. Technical report, The Oxford Institute for Energy Studies, Oxford.

Petrovich, B. (2013). European gas hubs: how strong is price correlation? Technical report, Oxford Institute for Energy Studies, Oxford, United Kingdom.

Qin, Q., Xie, K., He, H., Li, L., Chu, X., Wei, Y. M., and Wu, T. (2019). An effective and robust decomposition-ensemble energy price forecasting paradigm with local linear prediction. *Energy Economics*, 83:402–414.

Ram, M., Taklif, A., and Faridzad, A. (2019). Prediction of Natural Gas Price in European Gas Hubs Using Artificial Neural Network. *Petroleum Business Review*, 3(2):1–14.

Reiter, D. F. and Economides, M. J. (1999). Prediction of Short-term Natural Gas Prices Using Econometric and Neural Network Models. In *All Days*. SPE.

Salehnia, N., Falahi, M. A., Seifi, A., and Mahdavi Adeli, M. H. (2013). Forecasting natural gas spot prices with nonlinear modeling using Gamma test analysis. *Journal of Natural Gas Science and Engineering*, 14:238–249.

Shi, X. (2016). Development of Europe's gas hubs: Implications for East Asia. *Natural Gas Industry B*, 3(4):357–366.

Siddiqui, A. W. (2019). Predicting Natural Gas Spot Prices Using Artificial Neural Network. In *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–6. IEEE.

Su, M., Zhang, Z., Zhu, Y., and Zha, D. (2019a). Data-Driven Natural Gas Spot Price Forecasting with Least Squares Regression Boosting Algorithm. *Energies*, 12(6):1094.

Su, M., Zhang, Z., Zhu, Y., Zha, D., and Wen, W. (2019b). Data Driven Natural Gas Spot Price Prediction Models Using Machine Learning Methods. *Energies*, 12(9):1680.

Székely, G. J. and Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances.

Székely, G. J., Rizzo, M. L., and Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *Annals of Statistics*, 35(6):2769–2794.

Tang, L., Wu, Y., and Yu, L. (2018). A randomized-algorithm-based decomposition-ensemble learning methodology for energy price forecasting. *Energy*, 157:526–538.

Tang, Y., Wang, Q., Xu, W., Wang, M., and Wang, Z. (2019). Natural Gas Price Prediction with Big Data. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 5326–5330. IEEE.

Wang, J., Lei, C., and Guo, M. (2020). Daily natural gas price forecasting by a weighted hybrid data-driven model. *Journal of Petroleum Science and Engineering*, 192:107240.

Wu, Z. and Huang, N. E. (2009). Ensemble Empirical Mode Decomposition: A Noise-Assisted Data Analyis. *Advances in Adaptive Data Analysis*, 01(01):1–41.

Xie, Z. and Wen, H. (2019). Composite Quantile Regression Long Short-Term Memory Network. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11730 LNCS, pages 513–524. Springer Verlag.

Yang, X.-S. (2010). A New Metaheuristic Bat-Inspired Algorithm. In Juan R. González, David Alejandro Pelta, Carlos Cruz, Germán Terrazas, and Natalio Krasnogor, editors, *Nature Inspired Cooperative Strategies for Optimization (NICSO)*, volume 284, pages 65–74. Springer Berlin.

Yu, L., Yang, Z., and Tang, L. (2018). Quantile estimators with orthogonal pinball loss function. *Journal of Forecasting*, 37(3):401–417.