Instituto Nacional de Matemática Pura e Aplicada

Fitting and Forecasting Brent and WTI future prices curve using Machine Learning

Autor: **Mario de Mello Figueiredo Neto**

Orientador: **Yuri Saporito**

Rio de Janeiro
Agosto de 2021

Dedico este trabalho à minha família.

Agradecimentos

Agradeço a todos aqueles que me ajudaram e contribuíram para o desenvolvimento deste trabalho.

Faço menção especial à minha mãe, Ana, que foi fundamental na minha criação, e ao meu pai, Mario, e ao meu padrasto, Alexandre, por todo o apoio e incentivo durante esses anos. Sou muito grato também aos meus irmãos, Gabriela e Frederico, pelo apoio e companheirismo incondicionais.

Agradeço também ao meu amigo Leonardo, que me acompanhou durante todo o curso, por todo o suporte e ajuda.

Expresso meus agradecimentos também ao professor Yuri Saporito, que me acompanhou e orientou por todos esses meses, e cuja dedicação e apoio foram essenciais para que eu confeccionasse esse trabalho.

Termino agradecendo a todos os demais professores do curso.

Abstract

In this work, we fit the dynamic Nelson-Siegel and the Schwartz-Smith three-factor models to the future prices curves of Brent and WTI. We also study the prediction of the one day ahead prices of the entire curve for each model, using two different approaches: Ordinary Least Squares (OLS) and Filtering Approach. For the dynamic Nelson-Siegel, we test the Random Walk, VAR and models based on LSTM, including the LSTM-KF (a filtering approach using an LSTM neural network). For the Schwartz-Smith model, we use the equations given by no-arbitrage assumptions. We also have tested a model that tries to predict the prices directly from the curve, which we refer as Basic Model. We show that the dynamic Nelson-Siegel was the best model for fitting the data, and that the LSTM method with the OLS Approach generated the best forecasts.

Key words: Dynamic Nelson-Siegel, Schwartz-Smith three-factor model, LSTM, LSTM Kalman Filter, Kalman Filter, Commodities Term Structure.

Resumo

Nesse trabalho, nós ajustamos os modelos Nelson-Siegel dinâmico e Schwartz-Smith de três fatores à curva de preços futuros do Brent e do WTI. Também estudamos a previsão de preços um dia à frente para toda a curva para cada modelo. Para isso, utilizamos duas abordagens diferentes: por OLS (minimzação do erro quadrático) e por Filtragem. Para o modelo Nelson-Siegel dinâmico, nós utilizamos o passeio aleatório, o VAR e modelos baseados em LSTM, incluindo uma versão de LSTM com filtragem (LSTM-KF). Para o modelo de Schwartz-Smith de três fatores, utilizamos as equações que são dadas pelo modelo, assumindo não arbitragem. Nós também testamos um método de previsão a partir da curva diretamente, a que referimos como Basic Model. Nós mostramos que o modelo de Nelson-Siegel dinâmico foi o melhor no ajuste de dados em comparação com o de Schwartz-Smith e que a utilização de LSTM na abordagem OLS gerou as melhores previsões.

Palavras-Chaves: Modelo de Nelson-Siegel dinâmico, modelo Schwartz-Smith de três fatores, LSTM, LSTM Kalman Filter, Curva a termo de commodities.

Contents

Introduction

In this work, we study the problem of fitting and forecasting commodities future prices curve from past data. We apply the Nelson-Siegel parametrization to the curve of Brent and WTI, employing the Dynamic Nelson-Siegel model, as proposed by [Diebold and Li, 2006], and compare it to the Schwartz-Smith three-factor model, which is an extension of the model developed by [Schwartz and Smith, 2000] and that was analysed in [Cortazar and Schwartz, 2003] and [Aiube and Samanez, 2014].

The literature around fitting commodities term structure concentrates mainly in no-arbitrage stochastic factor and parametric factor models, although the use of them for forecasting is not that common, and ever more rare is the usage of machine learning methods for this purpose.

In this work, we start by using an OLS (Ordinary Least Squares) approach to extract the factors on each timestamp. These factors are hidden variables that have less dimension than the original model, but retain the most information possible. We then fit different models to forecast these factors. We employ an LSTM (Long Short-Term Memory) neural network approach to estimate the time series and compare it to the Random Walk, in which the factors are assumed to keep the same value from one timestamp to the other, and VAR (vector autoregression) models.

We also test a filtering approach, in which the factors from the dynamic Nelson-Siegel model are the state variables. We consider different functions for the transition equations, specifically a VAR model and an LSTM network. This last approach will be regarded as an LSTM-KF (Long Short-Term Memory Kalman Filter) model during the rest of this work. As this is not a linear function, another type of filter is necessary. We apply the Unscented Kalman Filter (UKF), but also briefly discuss the Extended Kalman Filter (EKF), following [Durbin and Koopman, 2012].

The choice for the Nelson-Siegel parametrization was due to some reasons: the relative simplicity of the model allied with its great success in fitting term-structures, as we will discuss in the next chapter; the fact that the extracted factors are correlated with the main components that explain the data – as noted by [Diebold and Li, 2006] and also [Karstanje et al., 2017]; and finally because of the possibility to use flexible estimation methods, such as artificial neural networks.

We also fit the Schwartz-Smith three-factor model using both an OLS approach and the Kalman Filter technique, in order to compare its predictive power against the methods applied to the dynamic Nelson-Siegel. In both approaches, we make predictions using the equations derived from this model.

In terms of innovations, this work complements existing literature - for instance, [Aiube and Samanez, 2014], [Baruník and Malinská, 2016], [Freitas et al., 2000], [Grønborg and Lunde, 2016], [Karstanje et al., 2017] and [Schwartz and Smith, 2000] - in some ways: first by considering the LSTM in addition to the Nelson-Siegel parametrization, as well as the LSTM-KF, which is an advanced neural network filtering technique and finally by comparing the dynamic Nelson-Siegel to the Schwartz-Smith three-factor model.

In the next chapter, which is Chapter 1, we begin by exploring existing works in the area related both to fitting and estimating future prices curves, estimation of commodity prices in general using machine learning methods, and to works that employ neural networks combined with filtering techniques.

Then, in Chapter 2 we explore some characteristics of the data that we use in this work. In Chapter 3, we present the formulations of the models used to estimate the future prices curves. In Chapter 4, we enter in the details of the methods of estimation. After that, in Chapter 5, we present the results and in Chapter 6 we finish with the conclusions and discuss some of the possible future works.

Chapter 1

Literature Review

The task of modelling commodities future prices has received a lot of interest in the last few decades, specially to energy related products, and a large number of methods have been employed both for the modelling and forecasting of prices, taking or not into consideration the term structure.

We will make a brief exposition of some of the important works developed in this area, and that also influenced this work. We will also discuss some works that are not directly linked to commodities futures, but that lay the foundations of the estimation procedures that we utilize.

Some of the main models treated in the literature for the modelling of the curve of the futures are stochastic factor models based on no-arbitrage assumptions. One of the first works was developed by [Brennan and Schwartz, 1985], in which a one factor model for the spot price was assumed, considering a Geometric Brownian Motion. Later, [Gibson and Schwartz, 1990] developed the well-known two-factor model applied to crude oil. The factors considered were the spot price, following the Geometric Brownian Motion, and the convenience yield, which follows an Ornstein-Uhlenbeck reversal process.

In addition to this model, [Schwartz, 1997] also tested a three-factor model, in which the interest rate was also considered to be stochastic for crude oil, copper and gold.

Soon after, [Schwartz and Smith, 2000] developed the short/long term model for commodities future prices and applied to crude oil. They considered the spot price to be the sum of two distinct factors: a short term price deviation, following an Ornstein-Uhlenbeck reversal process, and a long term persistent price, following a Geometric Brownian Motion. The authors showed in their paper that this model was formally identical to the previous two-factor model developed in [Gibson and Schwartz, 1990], but claimed that this new model had greater interpretability, and that the factors were more orthogonal.

Following this paper, numerous other works used the short-long term model as base to their own works, such as [Lucia and Schwartz, 2002] and [Sørensen, 2002].

[Cortazar and Schwartz, 2003] proposed an expansion of this model, by including a third factor to the Schwartz-Smith model, in which the drift of the long-term equilibrium price follows a stochastic mean-reverting process.

This model was also later assessed by [Aiube and Samanez, 2014], in which they compare the two- and three-factor models to crude oil prices, showing that the latter fits better the term structure.

In a slight different direction, [Cortazar and Naranjo, 2006] applied an N-factor model, also to crude oil prices, concluding that three factors should be enough for fitting the term structure.

The inclusion of jumps in the short-term factors was considered by [Villaplana, 2003], in an application to electric energy prices, and by [Aiube et al., 2008], in an application to crude

oil prices. In the latter, the authors applied the Particle Filter, in replacement to the usual Kalman Filter, as the model was no longer Gaussian. They showed that the model was able to fit better to the term structure. They also considered the three-factor model expansion proposed in [Cortazar and Naranjo, 2006], showing that this model fitted better to the term structure, specially to the shorter and longer contracts.

[Hikspoors and Jaimungal, 2007] applied a diffusive two-factor model to crude oil prices, also considering the inclusion of jumps, and used it to precify spread options.

[Bhar and Lee, 2011] proposed yet another expansion to the Schwartz-Smith model, by first considering a second short term mean-reverting factor, and also by considering a time-varying market price of risk. They also applied the model to crude oil prices, and studied the implied risk premium, relating it to macroeconomic variables and important events to the crude oil market.

For a more detailed overview in the no-arbitrage models, we also recommend [Aiube, 2013], in which the main models are further explained.

Another class of models that have been increasingly applied to the modelling of commodities term structure is based on the Nelson-Siegel parametrization, first employed in [Nelson and Siegel, 1987] to fit the term structure of interest rates.

[Diebold and Li, 2006] and [Diebold et al., 2008] applied this parametrization dynamically to the term structure of interest rates, extracting the main factors – level, slope and curvature. The authors noted that this was basically a dynamic dimension reduction, but also without some of the drawbacks of usual dimension reduction methods, such as the principal component analysis. In order to do so, there have been proposed two main general approaches: extract the factors from the curve and then estimating this series independently using a VAR(1) or extract the factors as state variables using a Kalman filter; in this case, the transition equation also follows a VAR(1) process. They found that this methodology provided a better estimation than a random walk.

In their paper, [Diebold and Li, 2006] also talk about some of the extensions of the Nelson-Siegel model, including a no-arbitrage one. They argue, however, that this is not necessary for producing good forecasts. [Christensen et al., 2011] studied further the topic of no-arbitrage Nelson-Siegel extensions.

The dynamic Nelson-Siegel have been used by [West, 2012] to fit and estimate agriculture products prices – sugar and cotton. To account for seasonality, the authors used an extension of the method, including a seasonal factor as well.

[Grønborg and Lunde, 2016] used the dynamic Nelson-Siegel to forecast oil futures. In their work, the authors address the usability of this model to commodities future prices curve, since the model had been initially proposed to the interest rates term structure. The authors compare some of the stylized facts of crude oil futures to that of the interest rates, contemplating the main points that were accounted in [Diebold and Li, 2006]. They used daily time series and extracted the factors by using a series of cross-section regressions. To model the factors time series, instead of the VAR, the authors used a copula framework with a NIG-GARCH model. They showed that this generated better mean-squared errors (MSE) for both forecasting and direction forecasting of future prices compared to random walk, VAR and AR models.

[Baruník and Malinská, 2016] also employed the dynamic Nelson-Siegel to forecast the term structure of crude oil prices. In their paper, however, the authors employed a Focused Time Delay Neural Network (FTDNN) to estimate the factors time series. This is one of the few papers that, as far as we know, have combined the dynamic Nelson-Siegel with an Artificial Neural Network to forecast commodity term structure. In this paper, they created interpolated time series of constant days to maturity – 30, 60, 90, ... - using cublic splines. After the

extraction of the factors, the FTDNN was employed to forecast the prices on a horizon of 1, 3, 6 and 12 months ahead. They have found better results with the FTDNN in comparison to the random walk and VAR models.

Also based on the dynamic Nelson-Siegel, [Karstanje et al., 2017] has made a more profound analysis of commodities term structure. They used a version of the Nelson-Siegel model used in [Diebold et al., 2008] in which common factors across commodities were used to model the commonality between them. They fitted the dynamic Nelson-Siegel model adopting the one-step approach, where the factors are considered to be state variables and the model is estimated by Kalman Filter. For the transition equation, they considered a VAR(1) model, but using the difference of the level factor for stationarity reasons. The authors have applied this framework to the 24 commodities of the GSCI, using monthly prices. As some of these commodities present distinctful seasonal patters, they have used an extension to the model, adopting a seasonal factor whenever necessary. We point to some of the interesting results obtained in this work: first, it was shown that there are important common components in all the factors both under the perspective of commodity market and specific market sector; second that these components are partially explained by macroeconomic and fundamental variables; and third, that there is correlation between the level and slope factors to the spot price and convenience yield factors from the [Schwartz, 1997] three-factor model.

Still related to parametric models, more recently, [Kleppe et al., 2021] used the Svensson parametric curve (which is a four-factor extension of the Nelson-Siegel) to model crude oil futures. They have also employed a stochastic Wishart volatility model in their framework. They have used 24 monthly future WTI prices and concluded that the Svensson model displayed significantly better results compared to the Nelson-Siegel model when forecasting future prices.

Differently from the previous mentioned works, the usage of Machine Learning techniques to the modeling of commodity prices, specially energy related ones, account for a plethora of academic works. In these papers however, the term structure of the commodity is usually not taken into account.

For instance, [Jammazi and Aloui, 2012] use wavelet decomposition and neural network modeling to forecast crude oil prices, [Keynia, 2012] uses a composition neural network for forecasting eletricity prices, [Panapakidis and Dagoumas, 2016] also forecast eletricity prices via artificial neural networks, [Chen et al., 2017] forecast crude oil prices using a deep learning based model. Based on the LSTM (long short-term memory), [Peng et al., 2018] forecast eletricity prices. [Ding, 2018] use a decompose-ensemble methodology with AIC-ANN for crude oil forecasting. [Zhou et al., 2019] use a decomposition model named CEEMDAN (complete ensemble empirical mode decomposition with adaptive noise) and combine it with the usage of an extreme gradient boosting (XGBOOST) for prediction.

[Ghoddusi et al., 2019] reviewed the machine learning literature applied to the energy market, covering a large amount of different estimation methods.

Finally, we consider now some works that employ Kalman Filter – or an extension – combined with neural networks. This practice is not very common in the financial literature, but we refer here to some of the works and methods in this area.

We start by referring to [Wan and Nelson, 1997], in which the authors apply the Neural Dual Extended Kalman Filter to speech problems. We also refer to the famous paper by [Wan and Van Der Merwe, 2000], where the authors show the derivation of the Unscented Kalman Filter (UKF), first proposed in [Julier and Uhlmann, 1997]. In [Wan and Van Der Merwe, 2000], the authors compare the Extended Kalman Filter (EKF) with the Unscented Kalman Filter, showing that the latter has a considerable gain in fitting power. They also show the dual extended Kalman Filter and the dual Unscented Kalman Filter. In this case, the weights

also have a state-space representation. This model is studied in [Wan et al., 1999]. A similar approach is taken by [Freitas et al., 2000], in which they use sequential Monte Carlo methods in order to train the neural network.

[Zhang and Luh, 2005] apply an improved version of the EKF to predict market clearing price using a neural netowork.

[Dash et al., 2016] applied a dynamic neural network trained by an Unscented Kalman Filter to forecast eletricity prices.

[Krishnan et al., 2015] use a Deep Kalman Filter to make conterfactual inference to health data. They used a state-space model with normal transition equation, whose mean and variance were given by two LSTMs. The observation equation is set to follow a determined distribution of a function of the state variables, which itself is also an LSTM. In order to train the model, the authors use a variational inference technique, by minimizing a lower bound of the log-likelihood. With this method, they were also able to produce conterfactual inference, which was one of the main goals of the work.

Finally, [Coskun et al., 2017] use a kind of LSTM-KF to estimate pose estimation. They propose a state space model where the transition equation is given by a normal distribution, whose mean and variance are given by connected LSTM models. They also use the LSTM to calculate the observations variance matrix. To fit the model, they implemented the EKF to this state-space model, and used a loss function based on the filtering and forecasting errors to train the LSTM cells.

Chapter 2

Exploratory Analysis

In this work, we will analyze two different products: the ICE Brent Crude futures, referred as Brent, and the NYMEX West Texas Intermediate Crude Oil futures, referred as WTI. Crude oil is one of the most important and liquid commodities, making it a preferred one for studies in the literature. It has a great economical importance, being one of the world's most important sources of energy and used for transport, in industry, heating and many other applications.

Although Brent and WTI price series are very similar, we are including both of them to increase the total amount of data. Besides, it will also serve as a kind of robustness test: if the results for each series are not similar, it might imply a problem with the estimation method applied.

For further clarification, a future contract is a financial derivative in which both trading parties agree to buy/sell a specific amount of a certain product in a predetermined date - maturity - and price. Thus, the number of days to maturity is basically how many business days there are between the considered time and the maturity date determined in the contract. The first contract in a given time is the one with the shortest positive time to maturity. Because the first contract changes at the maturity date, we call this series a rolled contract. We can change the rolling rule as well, considering a certain time before maturity to roll the series, i.e., change the underlying contract. The second contract, third contract, and so on are the immediate next contracts in terms of maturity date in relation to the first contract.

Our data consisted of daily settlement prices for the first thirty six future contracts of Brent and WTI – whenever they were available – since 1985 up to the end of 2020. We also took into consideration the expiration date and calculated the number of business days until expiry for each contract. We have plotted the graph for the number of business days to expiry for all contracts in Figures 2.1 and 2.2.

After that, we studied which maturities had settlement prices available for each day. The goal was to set a reliable starting date, from which the number of contracts available would be stable. By doing this, we ensure that we will not have data problems during the estimation process. Below, we show the evolution of the number of contracts with settlement prices for each commodity in Figures 2.3 and 2.4.

Taking these charts into consideration, it was decided to set the start date of the analysis to the beginning of year 2000. Besides, we have resolved to use only the first 15 contracts. By doing this, we are certain that the number of contracts used during the estimation process does not change with time. What is more, we address more easily the problem of missing observations. Finally, this also remove contracts that are less liquid, have smaller volatility, and could possibly contribute to a worse fit of the data for the shortest contracts. The last step in the data analysis was to roll the contracts five business days before maturity. The reason

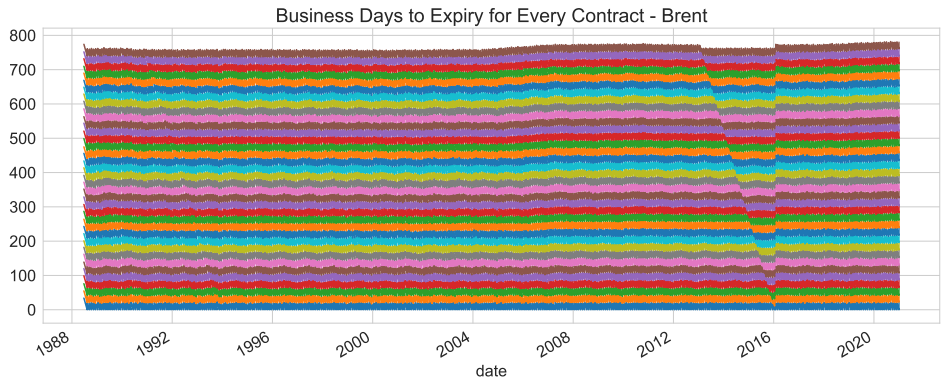

Figure 2.1: Business Days to Expiry for All Considered Contracts of Brent from 1988 to 2020. It is possible to see that, as expected, the time to maturity for each rolling contract does not change drastically over time.

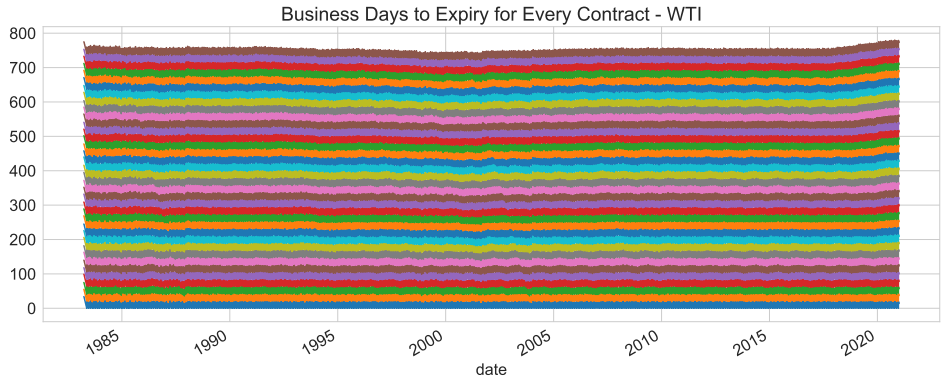

Figure 2.2: Business Days to Expiry for All Considered Contracts of WTI from 1985 to 2020. As with Brent, it is possible to see that, as expected, the time to maturity for each rolling contract does not change drastically over time.

is that the liquidity starts decreasing, and the price can suffer unexpected large variations, as was seen with the negative prices of WTI in the year 2020, which can also be seen in Figure 2.5 below. We also plot the prices for the rolled series in Figure 2.6.

The new first contract series generated for Brent - rolling five days before maturity - had an annualized daily return volatility of 36.4% since 2000, against the 36.8% of the previous series - rolling at the maturity. Thus, although there was not much loss of information in the series, we successfully avoided unnecessary volatility and noise, which were not the target of this study, as our main focus is to estimate and forecast the term structure as a whole. In the case of the WTI, the existence of the negative prices in 2020 distorted the volatility to a whopping 83.8%, against the 42.8% of the adjusted series.

After the adjustments were made, we ended up with 15 time series from the beginning of 2000 until the end of 2020 for both Brent and WTI. Below, we present some of the characteristics of the returns in Tables 2.1, 2.2, 2.3 and 2.4.

We can note that the distribution of Brent and WTI contracts returns are quite similar. It is also noticeable the fact that Brent returns have a slightest higher standard deviation than WTI returns. Besides that, we also point to the fact that the volatility of the contracts show

Figure 2.3: Number of Available Contracts - with settlement prices - for Brent from 1988 to 2020. We see that, with time, more contracts, with higher maturities were added.

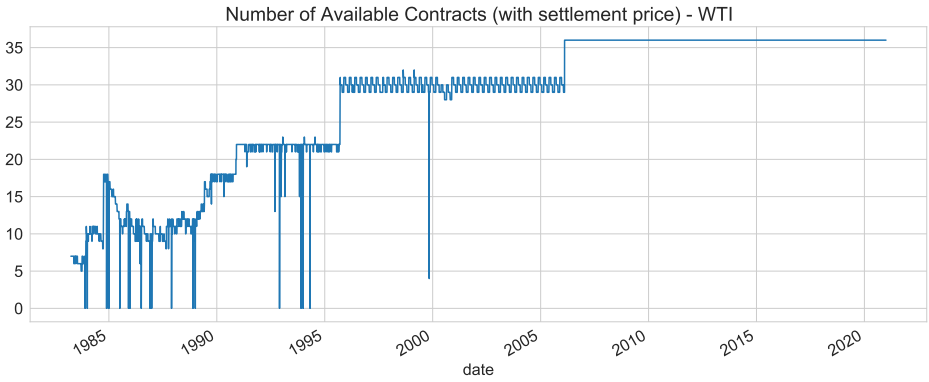

Figure 2.4: Number of Available Contracts - with settlement prices - for WTI from 1985 to 2020. The behaviour was very similar to what occured with Brent.

a decreasing trend towards higher expiration contracts for both Brent and WTI, which is a known fact in the literature as the Samuelson hypothesis - see [Aiube, 2013] for a more detailed explanation of the term structure of volatility in commodities.

Finally, we have plotted in Figure 2.7 the curves defined by these series on four different randomly selected dates: '2010-01-05', '2012-04-09', '2017-08-02' and '2019-06-20'. The x-axis corresponds to the expiration date of each contract.

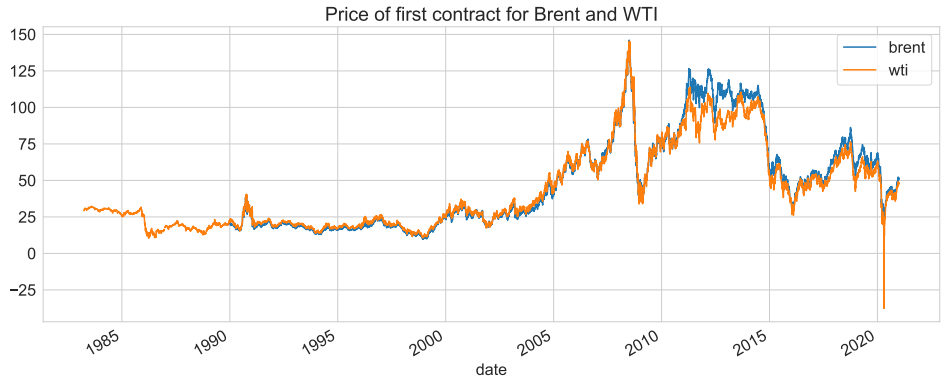

Figure 2.5: Price of the first contracts of Brent and WTI from 1985 to 2020. We take a special note on the negative price that the WTI series has achieved.

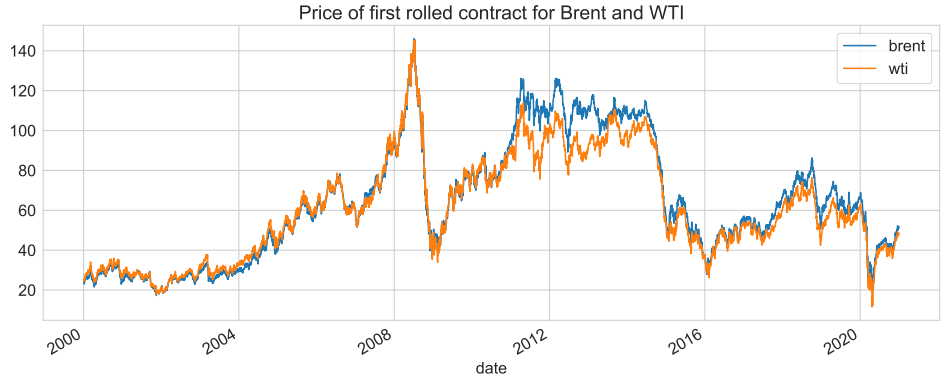

Figure 2.6: Price of the first rolled contracts of Brent and WTI from 1985 to 2020. In this case, the series are better behaved, which is clear by the fact that there are no more negative prices.

Contract	CO1	CO2	CO3	CO4	CO5	CO6	CO7	CO8
Count	5124	5124	5124	5124	5124	5124	5124	5124
Mean	0.03%	0.03%	0.03%	0.03%	0.03%	0.04%	0.04%	0.04%
Std	2.27%	2.16%	2.07%	2.00%	1.94%	1.89%	1.85%	1.80%
Min	-24.40%	-22.89%	-21.88%	-21.02%	-20.30%	-19.60%	-18.92%	-18.28%
25%	-1.11%	-1.05%	-1.01%	-0.98%	-0.96%	-0.94%	-0.92%	-0.90%
50%	0.09%	0.07%	0.07%	0.08%	0.08%	0.08%	0.08%	0.09%
75%	1.15%	1.11%	1.09%	1.07%	1.03%	1.01%	1.00%	0.98%
Max	21.02%	14.48%	13.63%	13.20%	12.58%	12.13%	11.82%	11.52%
Skewness	-0.22	-0.27	-0.27	-0.28	-0.28	-0.28	-0.27	-0.27
Kurtosis	9.81	7.77	6.95	6.50	6.22	5.92	5.66	5.46
Jarque Bera p-value	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2.1: Statistics of Brent Returns per Contract Rolled Series - Part 1.

Contract	CO9	CO10	CO11	CO12	CO13	CO14	CO15
Count	5124	5124	5124	5124	5124	5124	5124
Mean	0.04%	0.04%	0.04%	0.05%	0.05%	0.05%	0.05%
Std	1.77%	1.73%	1.70%	1.67%	1.65%	1.63%	1.60%
Min	-17.66%	-17.08%	-16.46%	-15.86%	-15.29%	-14.71%	-14.16%
25%	-0.88%	-0.86%	-0.84%	-0.82%	-0.81%	-0.78%	-0.77%
50%	0.08%	0.08%	0.09%	0.09%	0.09%	0.09%	0.09%
75%	0.97%	0.95%	0.95%	0.94%	0.93%	0.92%	0.90%
Max	11.03%	10.46%	9.99%	9.70%	9.76%	9.62%	9.51%
Skewness	-0.26	-0.26	-0.26	-0.25	-0.23	-0.22	-0.23
Kurtosis	5.24	5.03	4.85	4.63	4.40	4.25	4.19
Jarque Bera p-value	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2.2: Statistics of Brent Returns per Contract Rolled Series - Part 2.

Contract	CL1	CL2	CL3	CL4	CL5	CL6	CL7	CL8
Count	5021	5021	5021	5021	5021	5021	5021	5021
Mean	0.02%	0.03%	0.04%	0.04%	0.05%	0.05%	0.05%	0.05%
Std	2.62%	2.36%	2.22%	2.13%	2.05%	1.99%	1.94%	1.89%
Min	-43.37%	-28.88%	-24.20%	-22.79%	-21.95%	-21.06%	-20.20%	-19.38%
25%	-1.26%	-1.18%	-1.11%	-1.05%	-1.01%	-0.97%	-0.94%	-0.92%
50%	0.08%	0.09%	0.09%	0.10%	0.11%	0.12%	0.12%	0.12%
75%	1.26%	1.22%	1.17%	1.15%	1.11%	1.08%	1.06%	1.04%
Max	25.10%	22.16%	20.41%	18.70%	16.93%	15.40%	14.09%	12.91%
Skewness	-0.66	-0.48	-0.44	-0.44	-0.45	-0.45	-0.44	-0.43
Kurtosis	28.42	13.13	10.44	9.22	8.42	7.76	7.21	6.78
Jarque Bera p-value	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2.3: Statistics of WTI Returns per Contract Rolled Series - Part 1.

Contract	CL9	CL10	CL11	CL12	CL13	CL14	CL15
Count	5021	5021	5021	5021	5021	5021	5021
Mean	0.05%	0.05%	0.05%	0.05%	0.05%	0.05%	0.05%
Std	1.84%	1.80%	1.77%	1.73%	1.70%	1.68%	1.65%
Min	-18.57%	-17.86%	-17.18%	-16.50%	-15.87%	-15.32%	-14.83%
25%	-0.90%	-0.87%	-0.86%	-0.84%	-0.82%	-0.80%	-0.79%
50%	0.13%	0.12%	0.13%	0.13%	0.13%	0.13%	0.12%
75%	1.03%	1.01%	0.99%	0.96%	0.95%	0.94%	0.92%
Max	11.95%	11.15%	10.52%	10.47%	10.40%	10.36%	10.31%
Skewness	-0.42	-0.41	-0.40	-0.39	-0.38	-0.37	-0.37
Kurtosis	6.41	6.09	5.80	5.55	5.34	5.16	5.04
Jarque Bera p-value	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2.4: Statistics of WTI Returns per Contract Rolled Series - Part 2.

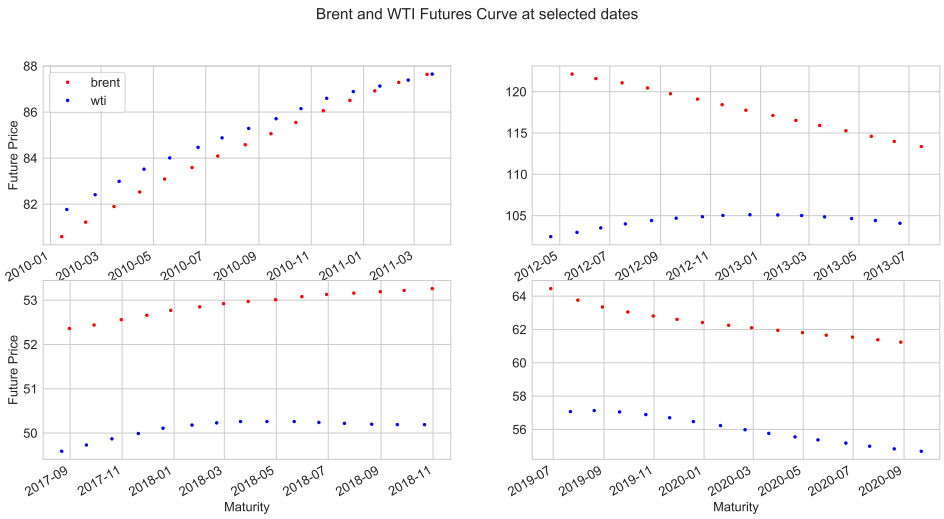

Figure 2.7: Term Structure of Future Prices for Brent and WTI for the first 15 maturities at different randomly selected dates. We see that, in general, both curves are very similar, although for some dates, this may not be the case.

Chapter 3

Term Structure

In this chapter, we overview the main models that will be used in this work for modelling commodities future prices.

3.1 Dynamic Nelson-Siegel

The Dynamic Nelson-Siegel model is based on the Nelson-Siegel parametrization [Nelson and Siegel, 1987] and was first used by [Diebold and Li, 2006] and [Diebold et al., 2008] to model the term structure of interest rates.

We start by presenting the parametric formula to calculate the price of a future contract. We mainly follow the notation used in [Karstanje et al., 2017].

Let $F_t(\tau)$ be the price of a contract with maturity $t+\tau$ at time t for a given commodity, with time measured in years. The maturity τ is dependent on time, but for the rest of this work, we will use the simpler notation τ. Consider that $t \in [0, T]$, and that $\tau \in \{\tau_1(t), \tau_2(t), ..., \tau_p(t)\}$ for each t, where p is a fixed and finite number that represents the number of maturities used and $\tau_p(t)$ is the maturity of contract p at time t. Then, the Nelson-Siegel parametrization of $F_t(\tau)$ is given as

$$\begin{aligned} F_t(\tau) &= L_t + S_t\left(\frac{1-\exp(-\lambda\tau)}{\lambda\tau}\right) + C_t\left(\frac{1-\exp(-\lambda\tau)}{\lambda\tau} - \exp(-\lambda\tau)\right) + \epsilon_t(\tau), \\ \epsilon_t &\sim N(0, \mathbf{H_t}). \end{aligned} \tag{3.1.1}$$

The error component ϵ_t follows a normal distribution with 0 mean and covariance matrix $\mathbf{H_t}$ with size $p \times p$.

The factors L_t, S_t and C_t vary with time and are not directly observed from the data, so that they need to be estimated. They are commonly referred to as the "level", "slope" and "curvature" factors, respectively, because of the shape of the coefficients associated with each of them. Besides both [Diebold and Li, 2006], for interest rates, and [Karstanje et al., 2017], for commodities, note that these factors are close to the first three principal components obtained by applying PCA (principal component analysis) to the future contract prices, which points that these factors are close to the best representation of the future prices with only three variables.

The parameter λ is considered to be fixed through time for simplification, although it could vary. It also may vary between different commodities. This parameter controls the decaying rate of the factor coefficients, hence it is associated with the number of contracts used in the estimation process, as well as the associated maturities. If the number of contracts or the

maturities change a lot with time, it might be a good practice to consider the λ parameter varying. Below, in Figures 3.1, 3.2 and 3.3, we show the shape of the factor coefficients versus τ and also show how they change with λ.

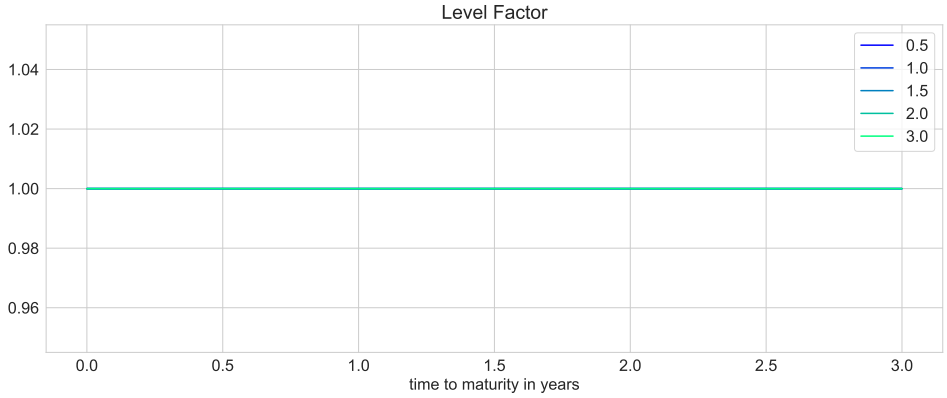

Figure 3.1: Level Factor versus Time to Maturity (τ) in years for increasing values of λ. We see that the loading for this factor is constant, weighting all contracts equally, independent of the value of λ.

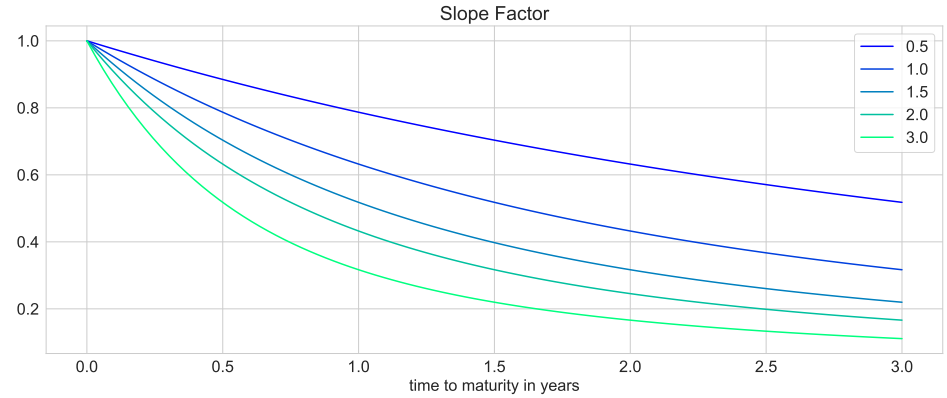

Figure 3.2: Slope Factor versus Time to Maturity (τ) in years for increasing values of λ. The higher the value of λ, the higher is the decay rate of the Slope Factor loading, weighting more the contracts with shorter times to maturity.

[Karstanje et al., 2017] utilizes a variation of the above formulation, considering the addition of a seasonal term and also by centralizing the regression variables - the factor coefficients. The authors note, however, that the seasonal term is not significant for some commodities, including WTI and Brent, which is a fact presumed by most of the discussed literature. Specifically in the case of crude oil, both the supply and demand are not much effected by the time of the year.

The dynamic nature of the model resides on the fact that the factors - L_t, S_t and C_t - are estimated for each timestamp, generating a time series across the entire period.

There are two main approaches to estimate this model: the OLS (ordinary least squares) and the filtering approach. In the OLS, the hidden factors are estimated using a linear regression of the future contracts observed prices versus the variables. The value of λ is optimized to minimize the total estimation error across the entire time series. After that, the extracted

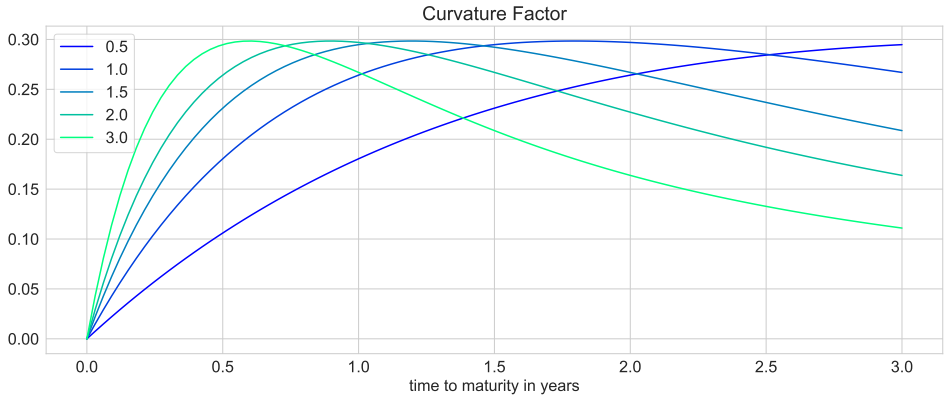

Figure 3.3: Curvature Factor versus Time to Maturity (τ) in years for increasing values of λ. The higher the value of λ, the higher is the decay rate of the Curvature Factor loading, weighting more the contracts with shorter times to maturity in relation to lower values of λ.

factors time series can be modelled by a number of different methods, some of which will be seen in Chapter 4. And with that, forecasts can be generated for the factors, automatically producing forecasts for the future contracts prices as well. This estimation process is thus composed of two independent steps. On another hand, in the filtering approach, both the factors - the state variables - and the future prices are estimated simultaneously, taking both estimation errors into account at the same time. This approach follows the well known Kalman Filter framework, in which the state variables are estimated in a forecast and update methodology. There are also multiple options for the transition equation for the state variables, including the random walk and the VAR.

In the following chapter we will enter in further details of the estimation procedures analysed in this work for all models described here.

3.2 Schwartz-Smith three-factor Model

The Schwartz-Smith three-factor model for commodities is an extension of the well-known Schwartz-Smith two factor model introduced by [Schwartz and Smith, 2000]. This model is based on no arbitrage assumptions for future contracts prices.

We will present the dynamics of the factors proposed in the three-factor model, as well as the equation for the pricing of the future contract based on the risk-neutral measure. For this part, we follow the notation used in [Aiube and Samanez, 2014], with small adaptations. For more details on the derivations of the equations, we also refer to [Aiube and Samanez, 2014].

The three factors considered in this model are defined as the following: χ_t is the short-term price variation factor, ξ_t is the long-term price factor and μ_t is the drift of ξ_t. Let S_t be the spot price. Consider again that $t \in [0, T]$, where 0 is the beginning of the time interval and T is its length. Then, the dynamics of the model can be written as:

$$\begin{aligned} \ln(S_t) &= g(t) + \chi_t + \xi_t, \\ d\chi_t &= -\kappa_\chi \chi_t dt + \sigma_\chi dW_t^\chi, \\ d\xi_t &= \mu_t dt + \sigma_\xi dW_t^\xi, \\ d\mu_t &= \kappa_\mu(\bar{\mu} - \mu_t)dt + \sigma_\mu dW_t^\mu, \end{aligned} \tag{3.2.1}$$

where $g(t)$ is a deterministic function - where a seasonal component could be inserted, $\kappa_\chi > 0$ and $\kappa_\mu > 0$ are the speed of reversion of the Ornstein-Uhlenbeck process followed by χ_t and μ_t respectively, $\sigma_\chi > 0$, $\sigma_\xi > 0$, $\sigma_\mu > 0$, $\mathbf{W}_t = (W_t^\chi, W_t^\xi, W_t^\mu)^T$ is a three dimensional Brownian process with $dW_t^\chi dW_t^\xi = \rho_{\chi\xi}dt$, $dW_t^\chi dW_t^\mu = \rho_{\chi\mu}dt$ and $dW_t^\xi dW_t^\mu = \rho_{\xi\mu}dt$, with $\rho_{\chi\xi}, \rho_{\chi\mu}, \rho_{\xi\mu} \in [-1, 1]$.

Under a risk-neutral measure $\mathbb{Q}$, the dynamics for the factors are given by:

$$\begin{aligned} d\chi_t &= -\kappa_\chi \left(-\frac{\lambda_\chi}{\kappa_\chi} - \chi_t \right) dt + \sigma_\chi d\widetilde{W}_t^\chi, \\ d\xi_t &= (\mu_t - \lambda_\xi)dt + \sigma_\xi d\widetilde{W}_t^\xi, \\ d\mu_t &= \kappa_\mu \left(\bar{\mu} - \frac{\lambda_\mu}{\kappa_\mu} - \mu_t \right) dt + \sigma_\mu d\widetilde{W}_t^\mu, \end{aligned} \tag{3.2.2}$$

where $(\lambda_\chi, \lambda_\xi, \lambda_\mu)$ is a constant market price of risk. For simplification, we can also define $\hat{\mu} = \bar{\mu} - \frac{\lambda_\mu}{\kappa_\mu}$.

Note that we skip the definitions of the measures $\mathbb{P}$ and $\mathbb{Q}$, as well as the processes defined by $\widetilde{W}_t$ for simplification, as these are well known in the context of finance, and do not compromise understanding of the topics of this work. For further understanding on "no-arbitrage" models and risk-neutral measures, we encourage the reader to find detailed explanations in [Aiube, 2013].

From Equations (3.2.1) and (3.2.2), [Aiube and Samanez, 2014] derives the formulas for calculating the price of the future contract and also the mean vector and variance-covariance matrix of the factors under the historical measure $\mathbb{P}$. These formulations are given below by Equations (3.2.3) to (3.2.5). They will be important for the model estimation, as it will be seen in the next chapter.

For $t \in [0, T]$, we can write:

$$\begin{aligned} \ln(F_t(\tau)) =& g(t+\tau) + \chi_t \exp(-\kappa_\chi \tau) + \xi_t + \mu_t \frac{1 - \exp(-\kappa_\chi \tau)}{\kappa_\mu} + C(\tau), \\ C(\tau) =& (\hat{\mu} - \lambda_\xi)\tau - \frac{\lambda_\chi}{\kappa_\chi}(1 - \exp(-\kappa_\chi \tau)) + \frac{\sigma_\chi^2}{4\kappa_\chi}(1 - exp(-2\kappa_\chi \tau)) + \\ &+ \frac{1}{2}\sigma_\xi^2 \tau - \hat{\mu}\frac{1 - \exp(-\kappa_\chi \tau)}{\kappa_\mu} + \\ &+ \frac{\sigma_\mu^2}{2\kappa_\mu^2}\left[\tau + \frac{1}{2\kappa_\mu}(1 - \exp(-2\kappa_\mu \tau)) - \frac{2}{\kappa_\mu}(1 - \exp(-\kappa_\mu \tau)) \right] + \\ &+ \frac{\rho_{\xi\mu}\sigma_\xi\sigma_\mu}{\kappa_\mu}\left[\tau - \frac{1}{\kappa_\mu}(1 - \exp(-\kappa_\mu \tau)) \right] + \frac{\rho_{\chi\xi}\sigma_\xi\sigma_\chi}{\kappa_\chi}(1 - \exp(-\kappa_\chi \tau)) + \\ &+ \frac{\rho_{\chi\mu}\sigma_\chi\sigma_\mu}{\kappa_\mu}\left[\frac{1}{\kappa_\chi}(1 - \exp(-\kappa_\chi \tau)) - \frac{1}{\kappa_\chi - \kappa_\mu}(\exp(-\kappa_\mu \tau) - \exp(-\kappa_\chi \tau) \right]. \end{aligned} \tag{3.2.3}$$

$$\mathbb{E}^{\mathbb{P}}\left(\begin{bmatrix} \chi_t \\ \xi_t \\ \mu_t \end{bmatrix} \right) = \begin{bmatrix} \chi_0 \exp(-\kappa_\chi t) \\ \xi_0 + \bar{\mu}t + (\mu_0 - \bar{\mu})\frac{1-\exp(-\kappa_\mu t)}{\kappa_\mu} \\ \bar{\mu} + (\mu_0 - \bar{\mu})\exp(-\kappa_\mu t) \end{bmatrix}. \tag{3.2.4}$$

$$Cov^{\mathbb{P}}\left(\begin{bmatrix}\chi_t\\ \xi_t\\ \mu_t\end{bmatrix}\right) = \begin{bmatrix} Var(\chi_t) & Cov(\chi_t,\xi_t) & Cov(\chi_t,\mu_t)\\ Cov(\xi_t,\chi_t) & Var(\xi_t) & Cov(\xi_t,\mu_t)\\ Cov(\mu_t,\chi_t) & Cov(\mu_t,\xi_t) & Var(\mu_t)\end{bmatrix}, \tag{3.2.5a}$$

$$Var^{\mathbb{P}}(\chi_t) = \frac{\sigma_\chi^2}{2\kappa_\chi}(1-\exp(-2\kappa_\chi t)), \tag{3.2.5b}$$

$$Var^{\mathbb{P}}(\xi_t) = \sigma_\xi^2 t + \frac{\sigma_\mu^2}{\kappa_\mu^2}\left[t + \frac{1-exp(-2\kappa_\mu t)}{2\kappa_\mu} - \frac{2\left(1-exp(-\kappa_\mu t)\right)}{\kappa_\mu}\right] + \tag{3.2.5c}$$

$$+ \frac{2\rho_{\xi\mu}\sigma_\xi\sigma_\mu}{\kappa_\mu}\left[t - \frac{1-\exp(\kappa_\mu t)}{\kappa_\mu}\right], \tag{3.2.5d}$$

$$Var^{\mathbb{P}}(\mu_t) = \frac{\sigma_\mu^2}{2\kappa_\mu}(1-\exp(-2\kappa_\mu t)), \tag{3.2.5e}$$

$$Cov^{\mathbb{P}}(\chi_t,\xi_t) = \frac{\rho_{\chi\xi}\sigma_\xi\sigma_\chi}{\kappa_\chi}(1-\exp(-\kappa_\chi t)) + \tag{3.2.5f}$$

$$+ \frac{\rho_{\chi\mu}\sigma_\chi\sigma_\mu}{\kappa_\mu}\left[\frac{1-\exp(-\kappa_\chi t)}{\kappa_\chi} - \frac{\exp(-\kappa_\mu t)-\exp(-\kappa_\chi t)}{\kappa_\chi-\kappa_\mu}\right], \tag{3.2.5g}$$

$$Cov^{\mathbb{P}}(\chi_t,\mu_t) = \frac{\rho_{\chi\mu}\sigma_\chi\sigma_\mu}{\kappa_\chi+\kappa_\mu}(1-\exp(-(\kappa_\chi+\kappa_\mu)t)), \tag{3.2.5h}$$

$$Cov^{\mathbb{P}}(\xi_t,\mu_t) = \frac{\rho_{\xi\mu}\sigma_\xi\sigma_\mu}{\kappa_\mu}(1-\exp(-\kappa_\mu t)) + \frac{\sigma_\mu^2}{\kappa_\mu}\left[\frac{1-\exp(-\kappa_\mu t)}{\kappa_\mu} - \exp(-\kappa_\mu t)\right]. \tag{3.2.5i}$$

Similar to the previously defined dynamic Nelson-Siegel model, the estimation for the Schwartz-Smith three-factor model can be done either by the OLS or filtering approaches. The main difference in this case is that the dynamics for the factors is already defined. That means that the model for the second step of the OLS approach is fixed, as well as the transition equation for the filtering approach.

Chapter 4

Estimation Methods

In this chapter, we detail the estimation methods considered to fit and forecast the models outlined in the previous chapter. For the more well-known and studied methods, only a brief summary will be provided, while, for the more complex models analysed, we will provide a more thorough description. We initially give a general treatment to each method, the algorithm and finally the specific application to each of the two models.

4.1 Conventions

For the subsequent sections, we will adopt some variable conventions, depicted below:

$\mathbf{y}_t$ is the response variable which we want to model and forecast, and whose values are directly observed. In the case of the dynamic Nelson-Siegel model, it will be the future prices for a determined commodity, whereas in the case of the Schwartz-Smith three-factor model, it will be the logarithm of the future prices. The subscript t corresponds to the time position of that value. That is, for the first value observed a value of 0 is assigned, then $1, 2, 3, ..., n$ in order of observation of the data, where n is the size of the entire series $\mathbf{y}_t$. In the case of this work, it corresponds to the number of tradable days since the beginning of the observations. For each t, $\mathbf{y}_t$ is a vector of dimension $p \times 1$, where p is the number of future contracts considered in that timestamp.

$\boldsymbol{\alpha}_t$ is the vector of factors that represents the model. The index follow the same logic described above. For each t, $\boldsymbol{\alpha}_t$ is a vector of dimension $m \times 1$, where m is the number of factors.

Θ is the set of all parameters - both for the term structure model and the estimation method employed - that need to be estimated, either by maximization of likelihood, minimization of error or by other methods, as will be seen in more details in the following sections.

In order to make the text easier to understand, we consider two disjoint subsets of Θ: Θ_M and Θ_E, such that $\Theta = \Theta_M \cup \Theta_E$. Θ_M will include the parameters relative to the term structure, while Θ_E will encompass the parameters relative to the estimation method. Besides, we will also consider two disjoint subsets of Θ_E, namely Θ_H and Θ_F, such that $\Theta_E = \Theta_H \cup \Theta_F$, where the first will correspond to the hyperparameters of the estimation method, while the latter will correspond to the parameters that will be fitted.

4.2 Data Processing

Before describing the estimation procedures, we will give the general guidelines for the data processing applied.

For all models, we divide the data into train, validation and test, in a proportion of roughly $0.8 : 0.1 : 0.1$. For each timestamp from the validation or test data, all the information used to make a prediction must already be known. As different models may produce predictions with different lengths, we choose to make the size of the test and validation equal across all models, so that we can compare all of them with the exact same time frame.

Given this division of the data, the process of determining the values of Θ_H will be based on choosing multiple sets of values and comparing the validation data results for each one of them. As for Θ_F, given Θ_H, it will be calculated based on the estimation method equations using test data.

Although the models in Chapter 3 are defined in continuous time, in practice we consider discrete time for estimation. Specifically, we consider each timestamp to represent a different trading day. It is important to note, however, that whenever time is directly used in a formula (i.e. it is not an index), it will be measured in years.

For some of the methods of estimation used, we have differentiated the data to produce a stationary time series - at least in the sense of the ADF (Augmented Dick Fuller) test, as shown in Section 5.2. And for the models using LSTM (Long Short-Term Memory), we also have scaled the data to be in the range $[-1, 1]$. We do that because, as will be seen, the output of an LSTM is also given in this range. To scale the data, we apply the following formula, where $\mathbf{x}_i^*$ is the i-th value of $\mathbf{x}$ scaled, for any given input $\mathbf{x}$, and $\mathbf{x_{train}}$ is the subset of $\mathbf{x}$ which corresponds to the train data:

$$\begin{aligned} \mathbf{x}_i^* &= \frac{\mathbf{x}_i - a}{b}, \\ a &= \frac{\max\{\mathbf{x_{train}}\} + \min\{\mathbf{x_{train}}\}}{2}, \\ b &= \frac{\max\{\mathbf{x_{train}}\} - \min\{\mathbf{x_{train}}\}}{2}. \end{aligned} \tag{4.2.1}$$

In matrix form, for the case that the variable has more than one feature - let's say m features, we can write:

$$\begin{aligned} \mathbf{x_i^*} &= D(\mathbf{b})^{-1}(\mathbf{x_i} - \mathbf{a}), \\ \mathbf{a} &= \begin{bmatrix} (\max\{\mathbf{x_{train,1}}\} + \min\{\mathbf{x_{train,1}}\})/2 \\ \vdots \\ (\max\{\mathbf{x_{train,m}}\} + \min\{\mathbf{x_{train,m}}\})/2 \end{bmatrix}, \\ \mathbf{b} &= \begin{bmatrix} (\max\{\mathbf{x_{train,1}}\} - \min\{\mathbf{x_{train,1}}\})/2 \\ \vdots \\ (\max\{\mathbf{x_{train,m}}\} - \min\{\mathbf{x_{train,m}}\})/2 \end{bmatrix}, \\ D(\mathbf{b}) &= \begin{bmatrix} \mathbf{b}_1 & 0 & \dots & 0 \\ 0 & \mathbf{b}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{b}_m \end{bmatrix}, \end{aligned} \tag{4.2.2}$$

where $\mathbf{x_i^*}$ is the i-th row of scaled $\mathbf{x}$, which will have dimension $1 \times m$. Basically, each feature scaling is done completely separate, so we may use the first notation for the multi-dimensional case as well for simplicity.

Note from the equations that, to scale the data, we also follow the train and test data framework, that is, we only use data from the train set.

Also note that we will use an asterisk whenever we want to indicate that the corresponding variable is scaled.

4.3 OLS Approach

As the name suggests, in this approach, the vector $\boldsymbol{\alpha}_t$ is estimated via linear regression for each time step. Hence, we can write the model as following:

$$\begin{aligned} \mathbf{y}_t &= \mathbf{Z_t}(\Theta_M)\boldsymbol{\alpha}_t + \mathbf{d}_t(\Theta_M) + \boldsymbol{\epsilon}_t, \\ \boldsymbol{\epsilon}_t &\sim N(0, \mathbf{H_t}), \end{aligned} \tag{4.3.1}$$

where $\mathbf{Z_t}(\Theta_M)$ is a deterministic matrix of shape $p \times m$, whose values depend on the parameters Θ_M and $\mathbf{d}_t(\Theta_M)$ is a deterministic vector of shape $p \times 1$ that also depends on Θ_M. This matrix corresponds to the regression variables. The use of $\mathbf{d}_t(\Theta_M)$ is merely for simplicity of representation, in practice, for a given Θ_M, we can run the regression with $\mathbf{y}_t - \mathbf{d}_t(\Theta_M)$. The component error $\boldsymbol{\epsilon}_t$, which has shape $p \times 1$, follows a normal distribution with mean 0 and variance matrix given by $\mathbf{H_t}$, which has shape $p \times p$.

The estimation of Θ_M is denoted by $\hat{\Theta}_M$, which is calculated by minimizing the total error across the entire train time series, written as follows:

$$\hat{\Theta}_M = \underset{\Theta_M}{\operatorname{argmin}} \left(\sum_{t=0}^{n_{train}-1} \sum_{j=0}^{p-1} \epsilon_{t,j}(\Theta_M)^2 \right). \tag{4.3.2}$$

For the dynamic Nelson-Siegel model, we can write:

$$\Theta_M = \lambda, \tag{4.3.3a}$$

$$\mathbf{y_t} = \begin{bmatrix} F_t(\tau_{t,1}) \\ \vdots \\ F_t(\tau_{t,p}) \end{bmatrix}, \tag{4.3.3b}$$

$$\mathbf{Z_t}(\lambda) = \begin{bmatrix} 1 & \left(\frac{1-exp(-\lambda\tau_{t,1})}{\lambda\tau_{t,1}}\right) & \left(\frac{1-exp(-\lambda\tau_{t,1})}{\lambda\tau_{t,1}} - exp(-\lambda\tau_{t,1})\right) \\ \vdots & \vdots & \vdots \\ 1 & \left(\frac{1-exp(-\lambda\tau_{t,p})}{\lambda\tau_{t,p}}\right) & \left(\frac{1-exp(-\lambda\tau_{t,p})}{\lambda\tau_{t,p}} - exp(-\lambda\tau_{t,p})\right) \end{bmatrix}, \tag{4.3.3c}$$

$$\boldsymbol{\alpha}_t = \begin{bmatrix} L_t \\ S_t \\ C_t \end{bmatrix}, \tag{4.3.3d}$$

$$\mathbf{d}_t = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.3.3e}$$

where $\tau_{t,j}$ is the time to maturity in years of the j-th contract at time t, as constructed in Chapter 2, and p is the total number of contracts used in the estimation.

As for the Schwartz-Smith three-factor model, based on Equations (3.2.1) and (3.2.3), we can write:

$$\Theta_M = \{\kappa_\chi, \kappa_\mu, \hat{\mu}, \bar{\mu}, \lambda_\xi, \lambda_\chi, \sigma_\chi, \sigma_\xi, \sigma_\mu, \rho_{\xi\mu}, \rho_{\chi\mu}, \rho_{\chi\xi}\}, \tag{4.3.4a}$$

$$\mathbf{y_t} = \begin{bmatrix} \ln\left(F_t(\tau_{t,1})\right) \\ \vdots \\ \ln\left(F_t(\tau_{t,p})\right) \end{bmatrix}, \tag{4.3.4b}$$

$$\mathbf{Z_t}(\Theta_M) = \begin{bmatrix} \exp(-\kappa_\chi \tau_{t,1}) & 1 & \frac{1-\exp(-\kappa_\chi \tau_{t,1})}{\kappa_\mu} \\ \vdots & \vdots & \vdots \\ \exp(-\kappa_\chi \tau_{t,p}) & 1 & \frac{1-\exp(-\kappa_\chi \tau_{t,p})}{\kappa_\mu} \end{bmatrix}, \tag{4.3.4c}$$

$$\boldsymbol{\alpha}_t = \begin{bmatrix} \chi_t \\ \xi_t \\ \mu_t \end{bmatrix}, \tag{4.3.4d}$$

$$\mathbf{d}_t = \begin{bmatrix} C(\tau_{t,1}) \\ \vdots \\ C(\tau_{t,p}) \end{bmatrix}, \tag{4.3.4e}$$

where we considered $g \equiv 0$.

The algorithm for obtaining the estimation is simple. We start by generating N random initial values $\Theta_{M0}^{(j)}, j = 1, ..., N$ within a parameter domain, respecting the theoretical bounds for each parameter, as well as the practical values obtained in the literature. The reason is that the optimization for Θ_M is not necessarily guaranteed to produce a global minimum, and by choosing a lot of different random initial values we might increase the accuracy of the optimization.

For each initial value $\Theta_{M0}^{(j)}$, we run an optimization algorithm to minimize the sum of errors, which are calculated directly from Equation (4.3.1) by linear regression. Multiple methods are available for minimizing the output of a function by changing the parameter. All the optimizations run in this work were produced by the `optimize.minimize` function from the Python package `scipy`. After running the minimization to each $\Theta_{M0}^{(j)}$, we choose the best result that will be our final $\hat{\Theta}_M$. With that, we can calculate the values of $\boldsymbol{\alpha}_t$ for each timestamp.

After the values of $\boldsymbol{\alpha}_t$ are retrieved, they are modelled by Equation (4.3.5) below, where $\mathbf{T_{\Delta t}}$ and $\mathbf{c_{\Delta t}}$ will vary depending on the estimation process applied, which will depend on the model considered, and Δt is the difference between two time steps measured in years. For the Schwartz-Smith three-factor model, we produce estimates for $\boldsymbol{\alpha}_t$ by applying the equations given in Section 3.2. This will be described in the next section. Whereas for the dynamic Nelson-Siegel model, we analyze three different methods, which will be described in Sections 4.3.2.1 to 4.3.2.3. Furthermore, $\boldsymbol{\eta}_t$ is an error component with covariance matrix given by $\mathbf{Q_t}$ - which is a parameter that will be fitted, so that $\mathbf{Q_t} \in \Theta_F$ for all methods, and that may depend on the remaining parameters. We will consider that this error component follows a normal distribution for simplicity:

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \mathbf{T_{\Delta t}}(\Theta, \boldsymbol{\alpha}_t) + \mathbf{c}_{\Delta t} + \boldsymbol{\eta}_t, \\ \boldsymbol{\eta}_t &\sim N(0, \mathbf{Q_t}). \end{aligned} \tag{4.3.5}$$

With this model, we can generate a forecast $\hat{\boldsymbol{\alpha}}_{t+1}$ for $\boldsymbol{\alpha}_{t+1}$ and then can calculate the forecast $\hat{\mathbf{y}}_{t+1}$:

$$\hat{\boldsymbol{\alpha}}_{t+1} = \mathbf{T}_{\boldsymbol{\Delta t}}(\hat{\Theta}, \boldsymbol{\alpha}_t) + \mathbf{c}_{\Delta t}, \tag{4.3.6}$$

$$\hat{\mathbf{y}}_{t+1} = \mathbf{Z}_{\mathbf{t+1}}(\hat{\Theta})\hat{\boldsymbol{\alpha}}_{t+1} + \mathbf{d}_{t+1}(\hat{\Theta}) \tag{4.3.7}$$

where $\hat{\Theta}$ is the estimated value of Θ. In order to estimate the value of Θ, we must estimate both Θ_M - using Equation 4.3.2 - and Θ_E. The estimation of Θ_E will depend on the estimation method considered. However, for all methods, Θ_F will be estimated using only test data, while the estimation of Θ_H will depend on validation data.

In the next sections, we will discuss the estimation methods that will be considered for the Schwartz-Smith three-factor model and for the dynamic Nelson-Siegel model.

4.3.1 Schwartz-Smith three-factor model

For the Schwartz-Smith three-factor model, $\mathbf{T}_{\boldsymbol{\Delta t}}$ and $\mathbf{c}_{\boldsymbol{\Delta_t}}$ from Equation (4.3.5) are given in Equation (4.3.8) below, derived from equation (3.2.4).

$$\begin{aligned} \mathbf{T}_{\boldsymbol{\Delta t},\mathbf{SS}}(\Theta, \boldsymbol{\alpha}_t) &= \begin{bmatrix} \exp(-\kappa_\chi \Delta t) & 0 & 0 \\ 0 & 1 & \frac{1-\exp(-\kappa_\mu \Delta t)}{\kappa_\mu} \\ 0 & 0 & \exp(-\kappa_\mu \Delta t) \end{bmatrix} \boldsymbol{\alpha}_t, \\ \mathbf{c}_{\Delta t,SS} &= \begin{bmatrix} 0 \\ \bar{\mu}\left(\Delta t - \frac{1-\exp(-\kappa_\mu \Delta t)}{\kappa_\mu}\right) \\ \bar{\mu}(1-\exp(-\kappa_\mu \Delta t)) \end{bmatrix}. \end{aligned} \tag{4.3.8}$$

In this case, there are no additional parameters to be estimated, i.e., $\Theta_E = \Theta_F = \emptyset$. Furthermore, the estimation of $\mathbf{Q_t}$ is irrelevant in this case, as it is not used in Equation 4.3.7.

4.3.2 Dynamic Nelson-Siegel model

For the dynamic Nelson-Siegel model, we study different models for $\boldsymbol{\alpha}$ which will be analyzed in the following sections. We start with the simpler models - Random Walk and VAR, common in literature as benchmark for the models that apply machine learning, then proceed to the models applying LSTM.

4.3.2.1 Baseline: Random Walk

The Baseline model will serve as a benchmark for all the other methods that will be seen throughout this work, as it provides a forecast without any additional information other than the immediate previous value, thus being the simplest forecast we can calculate.

This model can be defined as follows:

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \boldsymbol{\alpha}_t + \boldsymbol{\eta}_t, \\ \boldsymbol{\eta}_t &\sim N(0, \mathbf{Q_t}). \end{aligned} \tag{4.3.9}$$

In this case, we also have $\Theta_E = \Theta_F = \emptyset$.

With that, the estimated value for $\boldsymbol{\alpha}$ in the next timestamp is given by:

$$\hat{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_t. \tag{4.3.10}$$

The estimation of $\mathbf{y}$ will be calculated using Equation (4.3.7).

4.3.2.2 VAR

The vector autoregressive (VAR) model is one of the most used models for multi-dimensional time series. There is a vast literature that explores the details of this method, and in this work we will follow [Lütkepohl, 2005].

In this model, the last P values of the time series are used to predict the next value in a linear way. In this manner, it already incorporates much more information from the time series than the baseline model. Besides, it is very simple and easy to interpret, being an excellent benchmark for more complex models.

For this model, we differentiate the factors in order to obtain stationarity (in the results, we also show stationarity tests for these time series).

With some adjustments in the notation - used for clarity, we can write the VAR(P) model (VAR of order P - in upper case to avoid confusion with p, the size of the observation vector) as the following:

$$\begin{aligned} \boldsymbol{\Delta\alpha}_t &= \sum_{j=1}^{P} \mathbf{A}_j \boldsymbol{\Delta\alpha}_{t-j} + \mathbf{d} + \boldsymbol{\eta}_t, \\ \boldsymbol{\eta}_t &\sim N(0, \mathbf{Q}), \\ \boldsymbol{\alpha}_t &= \boldsymbol{\alpha}_{t-1} + \boldsymbol{\Delta\alpha}_t, \end{aligned} \tag{4.3.11}$$

where $\mathbf{A}_j$ are fixed $m \times m$ coeffiecient matrices and $\mathbf{d}$ is a constant $m \times 1$ intercept vector. In this model, $\boldsymbol{\eta}_t$ is considered to be a white noise process with variance given by $\mathbf{Q}$. For simplicity, we will consider that it follows a normal distribution as well.

In this case, $\Theta_F = \{\mathbf{A_j}, j = 1, ..., P\} \cup \{\mathbf{Q}, \mathbf{d}\}$ and $\Theta_H = \{P\}$.

We will not enter in the details for estimating the model, as this can be seen in the aforementioned literature. Once the parameters Θ_F of the model - $\mathbf{A_j}$, $j = 1, ..., P$, $\mathbf{Q}$ and $\mathbf{d}$ - are estimated, either statically or dynamically, we can make predictions for $\mathbf{y}_t$ applying formula (4.3.7) and (4.3.12) below:

$$\hat{\boldsymbol{\alpha}}_{t+1} = \boldsymbol{\alpha}_t + \widehat{\boldsymbol{\Delta\alpha}}_{t+1}. \tag{4.3.12}$$

The number of lags will be selected based on the validation data. We will further clarify this in Section 5.3.2.

4.3.2.3 LSTM

Introduction

The use of machine learning methods to forecast financial time series have gained increased importance in the last decades, and is a very vast theme. Multiple methods are available for such purpose. [Masini et al., 2021] and [Lim and Zohren, 2021] gave a broader view of some of the most important methods in deep learning that are being used in the literature of time

series forecasting. In this work, we focus mainly on the LSTM (Long Short Term Memory) network, which is a type of Recurrent Neural Network (RNN).

Recurrent Neural Networks have the ability to process inputs sequentially, while feeding the output to the next time step. The LSTM is a specific type of RNN introduced by [Hochreiter and Schmidhuber, 1997] that addresses an important problem with the first RNNs that is the vanishing or exploding gradient, which is a well-known problem in the literature (see [Hochreiter and Schmidhuber, 1997] and [Goodfellow et al., 2016]). Because of these features, LSTMs have found a lot of success in many applications, as pointed by [Goodfellow et al., 2016], such as speech and handwritten recognition, but also with time series forecasting, as pointed by [Masini et al., 2021] and [Lim and Zohren, 2021].

[Olah, 2015] and [Phi, 2018] detailed introductory guides on the inner workings of LSTMs, which we use as a base for this section. Introductory video-lectures can be found in [Heaton, 2020]. [Goodfellow et al., 2016] gave a deeper theoretical background on RNNs, LSTMs and other deep learning techniques.[1]

We will briefly describe how an LSTM cell works, but we will mainly focus in describing the practical details pertinent to this work. We will define three main different LSTM models that we will use for forecasting future contracts prices. One of them is also the one used in the Filtering approach framework.

In Figure 4.1, we display the basic inner working of a LSTM cell. In the image, there are shown the operations that are made inside the cell. $s_t^{(j)}$ is the Cell State, $h_t^{(j)}$ is the Hidden State and $X_t^{(j)}$ is the input. Notice that $h_t^{(j)}$ is the output of the cell, but it is also an input for the subsequent time step. In this case, the index j correspond to the time step of the algorithm, which will be made clear.

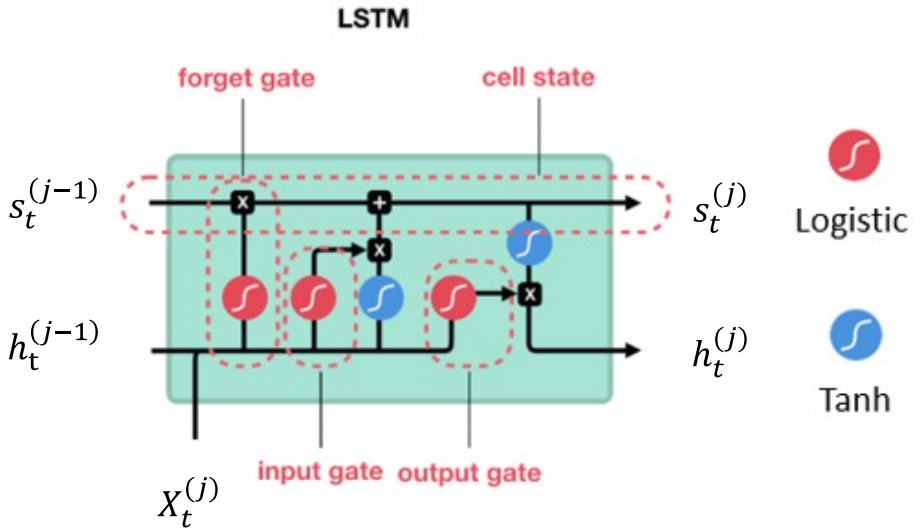

Figure 4.1: LSTM Cell - Adapted from [Phi, 2018].

The LSTM cells are chained together in order to feed the data sequentially for each time step. This could also be seen as a single cell with a recurrent calculation, hence we shall call this chain as LSTM hidden cell, which forms the LSTM hidden layer. This hidden cell may

[1]For a more practical, and implementation directed guide, we refer to https://www.tensorflow.org/tutorials/structured_data/time_series.

contain a lot of different units. The number of units - u - is basically the (first) dimension of $h_t^{(j)}$.

First, we explain how a single LSTM cell operates - at a single time step. The output of the previous time step $h_t^{(j-1)}$ and the current input $X_t^{(j)}$ will be combined through matrix multiplication and then addition and pass through three different gates. These gates are nothing more than Logistic Functions (e.g. "sigmoid") that either result in a positive value or zero, acting like a switch that lets a piece of the information, or none at all, pass to next node.

The forget gate chooses which information from the previous cell state to maintain, decrease or forget. This choice is effectively done by doing a point-wise multiplication between the result of the Logistic Function applied to the combination of $[h_t^{(j-1)}, X_t^{(j)}]$ and $s_t^{(j-1)}$. Depending on the values of $h_t^{(j-1)}$ and $X_t^{(j)}$, different kinds of information may be chosen to be kept or forgotten.

The input gate chooses which information from the combination of $H_t^{(j-1)}$ and $X_t^{(j)}$ to keep or discard. This is made by doing a point-wise multiplication between the result of the Logistic Function applied to the combination of $[H_t^{(j-1)}, X_t^{(j)}]$ and the scaled value of the combination of $[H_t^{(j-1)}, X_t^{(j)}]$ - this scaling is done using a tanh function that compresses the values between -1 and 1.

Finally, the output gate decides which information from the output - that is generated by the scaling of the sum of the output of the forget gate $f^{(j)}$ and the input gate $i^{(j)}$ - will be kept or discarded for generating the final output of the cell.

We summarize this operation in Equation (4.3.13), adapted from [Masini et al., 2021]:

$$f^{(j)} = Logistic(W_f X_t^{(j)} + U_f h_t^{(j-1)} + b_f), \tag{4.3.13a}$$
$$i^{(j)} = Logistic(W_i X_t^{(j)} + U_i h_t^{(j-1)} + b_i), \tag{4.3.13b}$$
$$o^{(j)} = Logistic(W_o X_t^{(j)} + U_o h_t^{(j-1)} + b_o), \tag{4.3.13c}$$
$$p^{(j)} = Tanh(W_c X_t^{(j)} + U_c h_t^{(j-1)} + b_c), \tag{4.3.13d}$$
$$s^{(j)} = (f^{(j)} \otimes s^{(j-1)}) + (i^{(j)} \otimes p^{(j)}), \tag{4.3.13e}$$
$$h^{(j)} = o^{(j)} \otimes Tanh(s^{(j)}), \tag{4.3.13f}$$

where $\{W_f, U_f, b_f, W_i, U_i, b_i, W_o, U_o, b_o, W_c, U_c, b_c\}$ are the parameters that need to be estimated, that is, the trainable variables of the LSTM hidden cell. We omit here the equations for backpropagation, as we deem unnecessary.

To understand how the whole hidden cell operates, we start with the time series that we want to predict, say $\boldsymbol{\Delta\alpha_t}$. Hence, the input at time t for the LSTM cell, denoted as X_t, can be written as:

$$X_t = \left[\boldsymbol{\Delta\alpha_{t-P}} \quad \boldsymbol{\Delta\alpha_{t-P+1}} \quad \dots \quad \boldsymbol{\Delta\alpha_{t-1}}\right], \tag{4.3.14}$$

where P is the number of time steps used in the estimation, similarly to the VAR(P) model. For each time step, $X_t^{(j)} = \boldsymbol{\Delta\alpha_{t-P+j}}, j = 0, ..., P-1$.

In other words, for each time stamp t, we consider P time steps of the input variable into our LSTM hidden cell, that consists of a repeating LSTM cell that operates sequentially in the input. This can be seen in the following picture, adapted from [Olah, 2015], which represents the repeating structure of the LSTM layer.

After describing the operation of a single LSTM hidden layer, we will present the general model framework in which this layer will be inserted. From this general model framework, we derive the models that will be used for this work application.

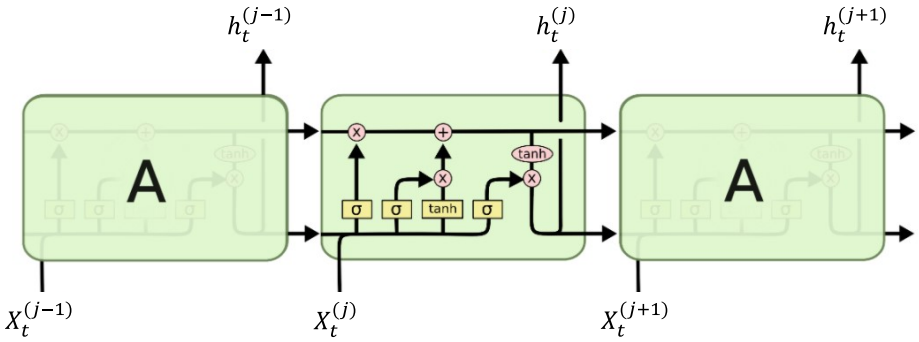

Figure 4.2: LSTM Diagram - Adapted from [Olah, 2015].

Before that, we make an important consideration about the input X of the LSTM hidden cell. As done in most cases [2], we will make the input X_t to be stationary, and also to be scaled between -1 and 1. The scaling formula was already given in Equation (4.2.1). We will refer to a variable scaled this way by including an asterisk. Thus, instead of using $\boldsymbol{\Delta\alpha_t}$, as in the example above, we would use $\boldsymbol{\Delta\alpha_t}^*$ as the input X_t.

LSTM Models

We then proceed to describe the general model framework, which is composed by three different components. The first component is the input, it comprises not only what data will be fed into the Deep Learning model, but also how this data is pre-processed and how it is divided into train and test data. The second component is comprised by the Deep Learning hidden layers, containing LSTM layers, dense layers, dropout layers, and data processing layers. And finally the third component is the output, which also determines how the loss will be calculated for the model fitting.

From this framework, we develop three different models that will be named as: the Basic Model, the Direct State Model and the Indirect State Model. The Basic Model will be considered later on the Chapter, as it does not belong to the OLS approach. For all models, the loss function considered is the mean squared error.

For the Direct State Model, the input at a time t and time step j is the $m \times 1$ vector given by the following Equation:

$$X_t^{(j)} = \boldsymbol{\Delta\alpha}_{t-P+j}^*. \tag{4.3.15}$$

In this case, X_t has a dimension of $P \times m$, where the number of features will be m.

Thus, we can write:

$$\begin{aligned} \widehat{\boldsymbol{\Delta\alpha}}_{t+1}^* &= \texttt{DirectStateModel}(X_t), \\ \hat{\boldsymbol{\alpha}}_{t+1} &= D(\mathbf{b}_{\Delta\alpha})\widehat{\boldsymbol{\Delta\alpha}}_{t+1}^* + \mathbf{a}_{\Delta\alpha} + \boldsymbol{\alpha}_t, \end{aligned} \tag{4.3.16}$$

where $\mathbf{a}_{\boldsymbol{\Delta}\alpha}$ and $\mathbf{b}_{\boldsymbol{\Delta}\alpha}$ are given by the scaling formula (4.2.1) for $\boldsymbol{\Delta\alpha}$, considering only train data. $\hat{\mathbf{y}}_t$ can then be derived using the Equation (4.3.7).

We show the basic layer graph for the Direct State Model in Figure 4.3. We start by the Input Layer following to stacked LSTM hidden layers. In the example, we utilize two layers,

[2] See for instance the guide https://www.tensorflow.org/tutorials/structured_data/time_series

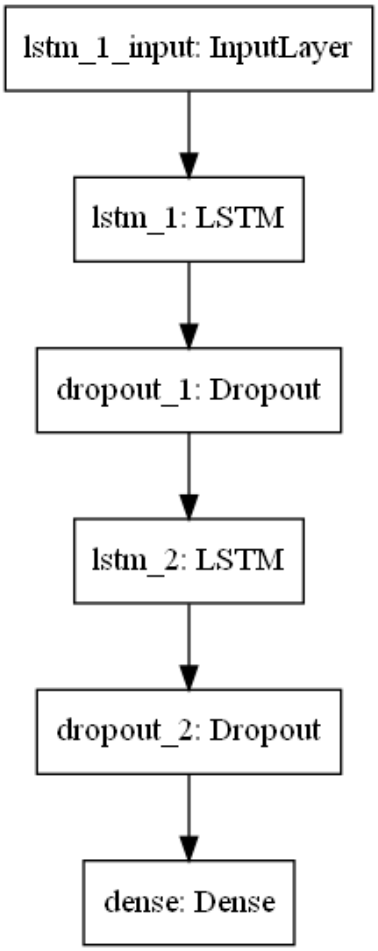

Figure 4.3: Direct State Model Layer Graph.

but more layers could be considered. It is important to take into account however that the input for the subsequent layers will have the dimension equal to number of units of the previous layer. The number of units is also a variable that needs to be tuned in the model. After each LSTM hidden layer, we also place a Dropout Layer to prevent over-fitting of the model. After the stacked LSTM hidden layers, we finish with a Dense Layer with a number of neurons equal to the length of the input, which will be our final output.

We now describe the Indirect State Model. The main difference from the Direct State Model is that, instead of trying to forecast the state variable in the next timestamp, we try to forecast the value of the observation. This is basically done by scaling the response of the Dense Layer - which outputs the scaled difference of the state variable - with a matrix $\mathbf{Z_t^{adj}}$ and comparing it to an adjusted observation $\mathbf{y}_t^{adj}$. The reason for this is that the LSTM model we utilize outputs the scaled variation of the factors and not the factors themselves, so that it is needed to convert this scaled variation into a value comparable to an adjusted observation so that we can calculate a loss to train the Neural Network.

We define $\mathbf{Z_t^{adj}}$ and $\mathbf{y}_t^{adj}$ in such a way that the graph would be the least complex possible. We start from the equation from the first difference of the factors:

$$\boldsymbol{\Delta\alpha}_t = \boldsymbol{\alpha}_t - \boldsymbol{\alpha}_{t-1}. \tag{4.3.17}$$

From Equation (4.2.2), we can write:

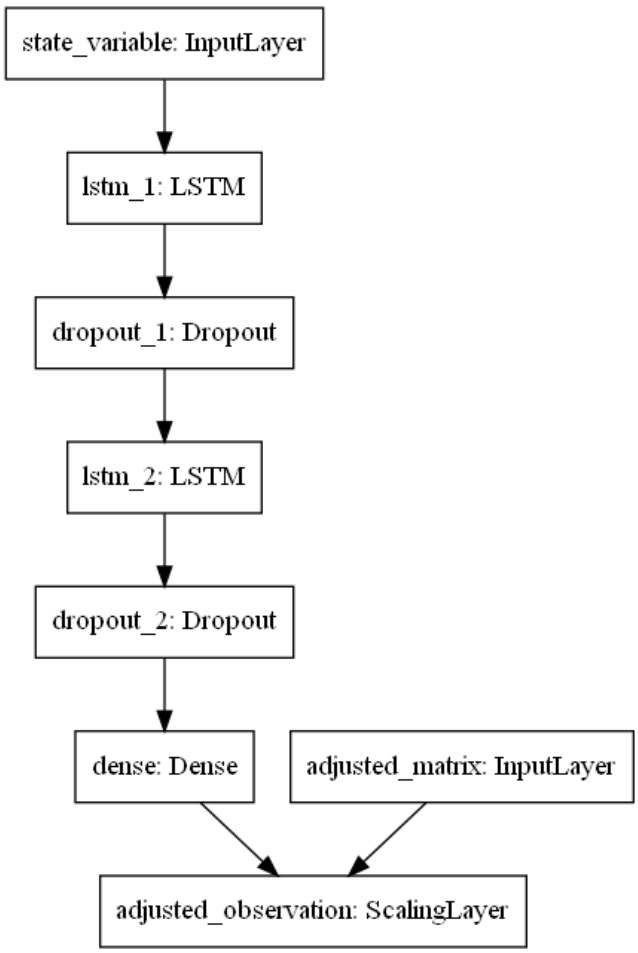

Figure 4.4: Indirect State Model Layer Graph.

$$\boldsymbol{\Delta\alpha}_t = D(\mathbf{b}_{\boldsymbol{\Delta\alpha}})\boldsymbol{\Delta\alpha}_t^* + \mathbf{a}_{\boldsymbol{\Delta\alpha}}. \tag{4.3.18}$$

At the same time, rewriting Equation (4.3.1) and applying Equation (4.3.18), we get:

$$\begin{aligned} (\mathbf{y}_t - \mathbf{d}_t) &= \mathbf{Z_t}\boldsymbol{\alpha}_t + \boldsymbol{\epsilon}_t, \\ (\mathbf{y}_t - \mathbf{d}_t) - \mathbf{Z_t}\boldsymbol{\alpha}_{t-1} - \mathbf{Z_t}\mathbf{a}_{\boldsymbol{\Delta\alpha}} &= \mathbf{Z_t}D(\mathbf{b}_{\boldsymbol{\Delta\alpha}})\boldsymbol{\Delta\alpha}_t^* + \boldsymbol{\epsilon}_t. \end{aligned} \tag{4.3.19}$$

Finally, using Equation (4.3.19), we can define Z_t^{adj} and $\mathbf{y}_t^{adj}$:

$$\mathbf{Z}_t^{adj} = \mathbf{Z_t}D(\mathbf{b}_{\boldsymbol{\Delta\alpha}}), \tag{4.3.20}$$

$$\mathbf{y}_t^{adj} = (\mathbf{y}_t - \mathbf{d}_t) - \mathbf{Z_t}\boldsymbol{\alpha}_{t-1} - \mathbf{Z_t}\mathbf{a}_{\boldsymbol{\Delta\alpha}}. \tag{4.3.21}$$

From Equations (4.3.19), (4.3.20) and (4.3.21), we write the final equation:

$$\mathbf{y}_t^{adj} = Z_t^{adj}\boldsymbol{\Delta\alpha}_t^* + \boldsymbol{\epsilon}_t. \tag{4.3.22}$$

It is possible to see that this equation is very similar to the definition of the OLS. In addition, $\mathbf{Z}_t^{adj}$ and $\mathbf{y}_t^{adj}$ can be directly calculated. In practice, we add another layer, which we will call Scaling Layer, that will multiply the output of the Dense Layer by $\mathbf{Z}_t^{adj}$. The result of this

multiplication will be compared to the output $\mathbf{y}_t^{adj}$, which we also provide to the model during the fitting phase. This framework is particularly interesting to use with the filter, as we can use the filtered state to create the input and compare with the actual observation in the next timestamp with only the inclusion of a matrix multiplication layer.

In Figure 4.4 above, we show the Layer Graph of the Indirect State Model. There is a second input corresponding to $\mathbf{Z}_t^{adj}$, and the addition of the Scaling Layer.

As the output of the Indirect State Model for timestamp t, we return both the predicted value $\widehat{\mathbf{\Delta\alpha}}^*_{t+1}$ and $\widehat{\mathbf{y}^{adj}}_{t+1}$, although only the second output is used to calculate the loss function. Calling $\texttt{IndirectStateModel}(X_t)_1$ the first output of the Indirect State Model, we can write, similarly to the Direct State Model prediction Equation (4.3.16):

$$\begin{aligned} \widehat{\mathbf{\Delta\alpha}}^*_{t+1} &= \texttt{IndirectStateModel}(X_t)_1, \\ \hat{\boldsymbol{\alpha}}_{t+1} &= D(\mathbf{b}_{\Delta\alpha})\widehat{\mathbf{\Delta\alpha}}^*_{t+1} + \mathbf{a}_{\Delta\alpha} + \boldsymbol{\alpha}_t, \end{aligned} \tag{4.3.23}$$

where $\mathbf{a}_{\Delta\alpha}$ and $\mathbf{b}_{\Delta\alpha}$ are given by the scaling formula (4.2.1) for $\mathbf{\Delta\alpha}$, considering only train data. $\hat{\mathbf{y}}_t$ can then be derived using the Equation (4.3.7).

For both the Direct State Model and the Indirect State Model, the fitting parameters will be the weights of each Deep Learning layers (including the LSTM layer and the dense layer). As for the hyperparameters Θ_H, we consider the number of time steps, number of layers, number of units, number of epochs,the batch size and the initialization. To determine these values, we will consider the validation data. As the LSTM converges to different models, depending on the initialization values, we will tune this also based on the validation data. This process will be further explained in Section 5.3.2.

4.4 Filtering Approach

In this approach, $\mathbf{y}_t$ and $\boldsymbol{\alpha}_t$ are modelled using a state-space representation. We consider $\mathbf{y}_t$ to be the output variable and $\boldsymbol{\alpha}_t$ to be the state variable. We follow the notation used in [Durbin and Koopman, 2012], with some necessary adaptations. The most general model we are going to consider can be written as:

$$\begin{aligned} \mathbf{y}_t &= \mathbf{Z_t}(\Theta_M)\boldsymbol{\alpha}_t + \mathbf{d}_t(\Theta_M) + \boldsymbol{\epsilon}_t, \\ \boldsymbol{\epsilon}_t &\sim N(0, \mathbf{H_t}), \end{aligned} \tag{4.4.1}$$

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \mathbf{T_t}(\Theta, \boldsymbol{\alpha}_t) + \mathbf{c}_t + \boldsymbol{\eta}_t, \\ \boldsymbol{\eta}_t &\sim N(0, \mathbf{Q_t}), \end{aligned} \tag{4.4.2}$$

for $t = 0, ..., n-1$, where the first equation is the measurement equation and the second one is the transition equation. Notice that $\mathbf{Z_t}$ is a matrix that depends on Θ_M, while T_t is a function that depends on both Θ and $\boldsymbol{\alpha}_t$. In the simplest case, it is a linear function and the model can be treated with Kalman Filter, that will be seen in the following section. In more complex models, $\mathbf{Z_t}$ can also be generalized as in (4.4.3). Also note that both errors were considered to be normally distributed, but that is a simplification, more general distributions could have been used.

Define $\mathbf{a_t}$, $\mathbf{a_{t|t}}$, $\mathbf{P_t}$ and $\mathbf{P_{t|t}}$ in the following way:

$$\begin{aligned} \mathbf{a_t} &= \mathbb{E}(\boldsymbol{\alpha_t}|\mathbf{y_1}, ..., \mathbf{y_{t-1}}), \\ \mathbf{a_{t|t}} &= \mathbb{E}(\boldsymbol{\alpha_t}|\mathbf{y_1}, ..., \mathbf{y_t}), \\ \mathbf{P_t} &= Var(\boldsymbol{\alpha_t}|\mathbf{y_1}, ..., \mathbf{y_{t-1}}), \\ \mathbf{P_{t|t}} &= Var(\boldsymbol{\alpha_t}|\mathbf{y_1}, ..., \mathbf{y_t}), \end{aligned} \tag{4.4.3}$$

In the linear case in $\boldsymbol{\alpha}_t$, we can write:

$$\boldsymbol{\alpha}_{t+1}|\boldsymbol{\alpha}_t \sim N(\mathbf{a_t}, \mathbf{P_t}), \tag{4.4.4}$$

where $\mathbf{a_t}$ and $\mathbf{P_t}$ will be estimated using the Kalman Filter, or other techniques if necessary. We consider that $\boldsymbol{\alpha}_0 \sim N(\mathbf{a_0}, \mathbf{P_0})$, where a_0 and P_0 will be estimated as well.

The estimation of $\Theta_M \cup \Theta_F$ is generally given by maximization of log-likelihood, but we will also discuss another method for our more complex model in Section 4.4.5.2. Either way, for this calculation we still follow the train and test data framework, that is, for test data predictions, we can only use Θ estimated from data available at that timestamp. We can also calculate this value dynamically by refitting the model as more data becomes available.

In the next three sections we provide the equations and algorithms to estimate this model under different circumstances. For more details about the calculations, more general models and computational aspects, we suggest the interested reader to see the reference [Durbin and Koopman, 2012].

4.4.1 Kalman Filter

We can use the Kalman Filter in the case that $\mathbf{T_t}$ is a linear function on $\boldsymbol{\alpha}_t$. Thus, the transition equation will be written as:

$$\mathbf{T_t}(\Theta, \boldsymbol{\alpha}_t) = \mathbf{T_t}(\Theta)\boldsymbol{\alpha}_t. \tag{4.4.5}$$

The algorithm of the Kalman Filter is given in Equation (4.4.6) below, taken from [Durbin and Koopman, 2012], where Θ is suppressed for simpler notation:

$$\mathbf{v}_t = \mathbf{y}_t - \mathbf{Z_t}\mathbf{a}_t - \mathbf{d}_t, \tag{4.4.6a}$$
$$\mathbf{F_t} = \mathbf{Z_t}\mathbf{P_t}{\mathbf{Z_t}}^T + \mathbf{H_t}, \tag{4.4.6b}$$
$$\mathbf{a}_{t|t} = \mathbf{a}_t + \mathbf{P_t}{\mathbf{Z_t}_t}^{T-1}\mathbf{v}_t, \tag{4.4.6c}$$
$$\mathbf{a}_{t+1} = \mathbf{T_t}\mathbf{a}_{t|t} + \mathbf{c}_t, \tag{4.4.6d}$$
$$\mathbf{P_{t|t}} = \mathbf{P_t} - \mathbf{P_t}{\mathbf{Z_t}}^T{\mathbf{F_t}}^{-1}\mathbf{Z_t}\mathbf{P_t}, \tag{4.4.6e}$$
$$\mathbf{P_{t+1}} = \mathbf{T_t}\mathbf{P_{t|t}}{\mathbf{T_t}}^T + \mathbf{Q_t}, \tag{4.4.6f}$$

for $t = 0, ..., n-1$.

To estimate Θ_M and Θ_F, we maximize the loglikelihood function given in Equation (4.4.7), also called prediction error decomposition. In this case we consider $t = 0, ..., n_{train} - 1$, as already stated earlier.

$$\log L(\Theta) = -\frac{n_{train}p}{2}\log(2\pi) - \frac{1}{2}\sum_{t=0}^{n_{train}-1}(\log|\mathbf{F_t}(\Theta)| + \mathbf{v}_t(\Theta)^T{\mathbf{F_t}}^{-1}(\Theta)\mathbf{v_t}(\Theta)). \tag{4.4.7}$$

4.4.2 Extended Kalman Filter

The Extended Kalman Filter can be applied to the model defined in Equations (4.4.1) and (4.4.5) when $T_t(\boldsymbol{\alpha}_t)$ is not a linear function. In this case, what this method does is to basically approximate the transition equation linearly and then apply the usual Kalman Filter. We will not enter in the calculation details, as they can be seen in the reference [Durbin and Koopman, 2012].

For the Extended Kalman Filter algorithm, we consider that $T_t(\boldsymbol{\alpha}_t)$ from Equation (4.4.3) is a differentiable function. Let $\dot{\mathbf{T}}_t = \frac{\partial T_t}{\partial \boldsymbol{\alpha}_t}(\mathbf{a}_{t|t})$. Then, the algorithm for the Extended Kalman Filter is written as:

$$\mathbf{v}_t = \mathbf{y}_t - \mathbf{Z_t}\mathbf{a}_t - \mathbf{d}_t, \tag{4.4.8a}$$
$$\mathbf{F_t} = \mathbf{Z_t}P_t{\mathbf{Z_t}}^T + \mathbf{H_t}, \tag{4.4.8b}$$
$$\mathbf{a}_{t|t} = \mathbf{a}_t + \mathbf{P_t}{\mathbf{Z_t}}^T{\mathbf{F_t}}^{-1}\mathbf{v}_t, \tag{4.4.8c}$$
$$\mathbf{a}_{t+1} = \mathbf{T_t}(\mathbf{a}_{t|t}) + \mathbf{c}_t, \tag{4.4.8d}$$
$$\mathbf{P_{t|t}} = \mathbf{P_t} - \mathbf{P_t}{\mathbf{Z_t}}^T{\mathbf{F_t}}^{-1}\mathbf{Z_t}\mathbf{P_t}, \tag{4.4.8e}$$
$$\mathbf{P_{t+1}} = \dot{\mathbf{T}}_\mathbf{t}\mathbf{P_{t|t}}{\dot{\mathbf{T}}_\mathbf{t}}^T + \mathbf{Q_t}, \tag{4.4.8f}$$

for $t = 0, ..., n-1$.

4.4.3 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is also an approximation, but with a different methodology from the Extended Kalman Filter. It is based on the Unscented Transformation (see [Julier and Uhlmann, 1997]). It uses a set of so-called sigma points and respective sigma weights that are used to approximate the density of $T_t(\mathbf{a}_{t|t})$. As pointed in [Wan and Van Der Merwe, 2000], the Unscented Kalman Filter is accurate up to the third order in Taylor series expansion, in contrast to the first order accuracy achieved by the Extended Kalman Filter.

In our study, this method is the preferred to use with the LSTM-KF, which will be defined in Section 4.4.5.2. Not only does it provide a more accurate result, but it also prevents us from estimating the jacobian of the LSTM function for every point. In fact, we only need to apply the function to $2m+1$ total points, whose calculations can be made parallely. Thus, it actually runs faster then the Extended Kalman Filter.

Below we show the algorithm for the Unscented Kalman Filter. We follow the notation of [Durbin and Koopman, 2012], where the derivations can be found.

We start by applying the Cholesky decomposition to $\mathbf{P_t}$, so that $\mathbf{P_t} = \bar{\mathbf{P}}_t\bar{\mathbf{P}}_t^T$. Let $\bar{\mathbf{P}}_{t,i}$ be the i-th column of $\bar{\mathbf{P}}_t$, then we can define the sigma points $\{\mathbf{x}_0, ..., \mathbf{x}_{2m+1}\}$ and sigma weights $\{w_0, ..., w_{2m+1}\}$ as:

$$\begin{aligned}
\mathbf{x_{t,0}} &= \mathbf{a_t}, & w_0 &= \frac{k}{m+k},\\
\mathbf{x_{t,i}} &= \mathbf{a_t} + \sqrt{m+k}\bar{\mathbf{P}}_{t,i}, & w_i &= \frac{1}{2(m+k)},\\
\mathbf{x_{t,i+m}} &= \mathbf{a_t} - \sqrt{m+k}\bar{\mathbf{P}}_{t,i}, & w_{i+m} &= \frac{1}{2(m+k)}.
\end{aligned} \tag{4.4.9}$$

After that, we define $\bar{\mathbf{y}}_t$, $\mathbf{P}_{\alpha v,t}$ and $\mathbf{P}_{vv,t}$ such that:

$$\begin{aligned}
\bar{\mathbf{y}}_t &= \sum_{i=0}^{2m} w_i(\mathbf{Z_t}\mathbf{x_{t,i}} + \mathbf{d_t}),\\
\mathbf{P}_{\alpha v,t} &= \sum_{i=0}^{2m} w_i(\mathbf{x_{t,i}} - \mathbf{a_t})(\mathbf{Z_t}\mathbf{x_{t,i}} - \bar{\mathbf{y}}_t),\\
\mathbf{P}_{vv,t} &= \sum_{i=0}^{2m} w_i(\mathbf{Z_t}\mathbf{x_{t,i}} - \bar{\mathbf{y}}_t)(\mathbf{Z_t}\mathbf{x_{t,i}} - \bar{\mathbf{y}}_t)^T + \mathbf{H_t},
\end{aligned} \tag{4.4.10}$$

for $t = 0, ..., n-1$. We can now calculate $\mathbf{a}_{t|t}$ and $P_{t|t}$ as:

$$\begin{aligned}
\mathbf{a}_{t|t} &= \mathbf{a}_t + \mathbf{P}_{\alpha \mathbf{v},\mathbf{t}}{\mathbf{P}_{\mathbf{vv},\mathbf{t}}}^{-1}(\mathbf{y}_t - \bar{\mathbf{y}}_t),\\
\mathbf{P_{t|t}} &= \mathbf{P_t} - \mathbf{P}_{\alpha v,t}\mathbf{P}^{-1}_{vv,t}\mathbf{P}^T_{\alpha v,t}.
\end{aligned} \tag{4.4.11}$$

To make the prediction step, we first recalculate the sigma points and sigma weights. First we apply the Cholesky decomposition to $\mathbf{P}_{t|t}$, such that $\mathbf{P}_{t|t} = \bar{\mathbf{P}}_{t|t}\bar{\mathbf{P}}^T_{t|t}$. Then, the new sigma points and weights are obtained in the following way:

$$\begin{aligned}
\mathbf{x_{t,0}} &= \mathbf{a_{t|t}}, & w_0 &= \frac{k}{m+k},\\
\mathbf{x_{t,i}} &= \mathbf{a_{t|t}} + \sqrt{m+k}\bar{\mathbf{P}}_{t|t,i}, & w_i &= \frac{1}{2(m+k)},\\
\mathbf{x_{t,i+m}} &= \mathbf{a_{t|t}} - \sqrt{m+k}\bar{\mathbf{P}}_{t|t,i}, & w_{i+m} &= \frac{1}{2(m+k)}.
\end{aligned} \tag{4.4.12}$$

With these new defined sigma points and sigma weights, we can proceed to calculate $\mathbf{a}_{t+1}$ and P_{t+1} as:

$$\begin{aligned}
\mathbf{a}_{t+1} &= \sum_{i=0}^{2m} w_i\mathbf{T_t}`(\mathbf{x_{t,i}}),\\
\mathbf{P_{t+1}} &= \sum_{i=0}^{2m} w_i \left((\mathbf{T_t}(\mathbf{x_{t,i}}) - \mathbf{a}_{t+1})(\mathbf{T_t}(\mathbf{x_{t,i}}) - \mathbf{a}_{t+1})^T + \mathbf{Q_t}\right),
\end{aligned} \tag{4.4.13}$$

for $t = 0, ..., n-1$.

4.4.4 Schwartz Smith three-factor model

For the Schwartz-Smith three-factor model, $\mathbf{Z_t}$ and $\mathbf{d_t}$ are the same as in Equation (4.3.4), while $\mathbf{T_t}$ and $\mathbf{c_t}$ are given in Equation (4.3.8). $\mathbf{Q_t}$ is given by the covariance formula (3.2.5) replacing t for Δt. The only difference is that Θ also contains $\mathbf{H}$ which is a constant matrix such that $\mathbf{H_t} = \mathbf{H}, \forall t$, as written in Equation (4.4.14) below. We also choose $\mathbf{H}$ to be a diagonal matrix for simplification (in order to reduce the number of parameters to estimate in the model).

$$\Theta_E = \Theta_F = \{\mathbf{H}\}. \tag{4.4.14}$$

To estimate the model, we can apply the Kalman Filter algorithm explained in Section 4.4.1.

4.4.5 Dynamic Nelson-Siegel model

For the dynamic Nelson-Siegel model, $\mathbf{Z_t}$ and $\mathbf{d_t}$ are the same as in Equation (4.3.3). As for $\mathbf{T_t}$, $\mathbf{c_t}$ and $\mathbf{Q_t}$, they can vary depending on the model used. They will be discussed in the next subsections. Finally, Θ_F will also include $\mathbf{H}$ - which we also consider to be constant - and the parameters for the transition equation model.

We will consider two different transition equation models for the dynamic Nelson-Siegel. The first one is the VAR(1), in which $\mathbf{T_t}$ is a linear function. And the second one is the LSTM, in which $\mathbf{T_t}$ is the output of an Indirect State Model, as defined in (4.3.2.3). These models will be discussed in more details in Sections 4.4.5.1 and 4.4.5.2 respectively.

4.4.5.1 Kalman Filter with VAR

In Kalman Filter with VAR model, the transition equation is given by (4.4.5), and thus the model can be estimated using the Kalman Filter algorithm described in (4.4.6).

This model has been used by [Karstanje et al., 2017] to estimate the dynamic Nelson-Siegel model for multiple commodities. We follow a similar path to [Karstanje et al., 2017], and choose to model the level factor as first differences, in order to make it stationary. We won't, however, take into account, market or sector components for these factors, as in the aforementioned work.

In order to write the model with ΔL_t instead of L_t, we need to use a different specification for $\boldsymbol{\alpha_t}$, as written below:

$$\boldsymbol{\alpha}_t = \begin{bmatrix} L_t \\ L_{t-1} \\ S_t \\ C_t \end{bmatrix} \quad \text{and} \quad \boldsymbol{\alpha}_0 = \begin{bmatrix} L_0 \\ L_0 \\ S_0 \\ C_0 \end{bmatrix}. \tag{4.4.15}$$

The transition matrix $\mathbf{T_t}$ is given in Equation (4.4.16) below and the vector $\mathbf{c_t}$ is a vector with zeros:

$$\mathbf{T_t} = \begin{bmatrix} 1+\phi_1 & -\phi_1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & \phi_2 & 0 \\ 0 & 0 & 0 & \phi_3 \end{bmatrix}, \tag{4.4.16}$$

where ϕ_1, ϕ_2 and ϕ_3 are parameters to be estimated.

As we have modified $\boldsymbol{\alpha_t}$, we must also modify $\mathbf{Z_t}$, by including a column of zeros in the second column.

Finally, for the matrix $\mathbf{Q_t}$, we write:

$$\mathbf{Q_t} = \begin{bmatrix} \sigma^2_{\eta_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{\eta_2} & 0 \\ 0 & 0 & 0 & \sigma^2_{\eta_3} \end{bmatrix}. \tag{4.4.17}$$

Hence, $\Theta_F = \{\phi_1, \phi_2, \phi_3, \sigma^2_{\eta_1}, \sigma^2_{\eta_2}, \sigma^2_{\eta_3}, \mathbf{H}\}$.

4.4.5.2 LSTM Kalman Filter

Introduction

The LSTM Kalman Filter (LSTM-KF) is the most complex model that will be considered in this work. It combines an LSTM Deep Learning model with a Filtering framework in order to simultaneously estimate the state variables and the observations.

We will briefly describe the models used by [Krishnan et al., 2015] and [Coskun et al., 2017]. Both authors use LSTMs in conjunction with a state space model.

[Krishnan et al., 2015] developed a model which they call Deep Kalman Filter. The goal was to be able to produce counterfactual inference in a medical setting. The authors have built a generative model, whose formulation we repeat in Equation below:

$$\begin{aligned} \boldsymbol{\alpha}_0 &\sim N(\boldsymbol{\mu}_0, \Sigma_0), \\ \boldsymbol{\alpha}_t &\sim N(G(\boldsymbol{\alpha}_{t-1}, \mathbf{u}_{t-1}, \Delta_t), S(\boldsymbol{\alpha}_{t-1}, \mathbf{u}_{t-1}, \Delta_t)), \\ \mathbf{y}_t &\sim \Pi(F(\boldsymbol{\alpha}_t)). \end{aligned} \tag{4.4.18}$$

In this equation, the observations are distributed according to a certain distribution Π (e.g. Bernoulli) whose parameters are given by a function F, which depends on the current state $\boldsymbol{\alpha}_t$, while the state variable is distributed by a Normal distribution, whose mean is modeled by a function G and variance by a function S. These two functions depend on the previous state $\boldsymbol{\alpha}_{t-1}$, the previous actions $\mathbf{u}_{t-1}$ and the time difference between current and last time Δ_t. The authors considered that F, G and S are the output of three LSTM models, although other neural networks architectures could be used. Finally, u_t is a sequence of actions.

In this manner, the LSTMs are learning how to infer the value of the state variable, while incorporating external actions and the observations values. The prediction and update equations are, therefore, not used in this case.

To estimate the model, the authors sequentially perform these steps until convergence: sample an observation vector $\mathbf{y}$; use the variational approximation $q_\phi(\boldsymbol{\alpha}|\mathbf{y}, \mathbf{u})$ to infer the values of α into $\hat{\boldsymbol{\alpha}}$; then compute the estimates $\hat{\mathbf{y}}$ using the distribution of $p_\theta(\mathbf{y}|\hat{\boldsymbol{\alpha}})$; finally compute the gradients of the lower bound of the loglikelihood of the variational model with respect to the parameters, which they update using Adam optimizer.

For more details on the model, especially the derivation of all the equations used, we refer to the original paper.

In [Krishnan et al., 2015] model, as we already stated, there is no use of the prediction and update framework of the filter, which is one of the characteristics that we want for our model.

On the other hand, [Coskun et al., 2017] develops a model that fully utilises the prediction and update framework. They model the transition equation as an LSTM, and the variance matrices of the errors are also modeled by LSTMs. There isn't a state space model for the weights of the models. Rather, the weights are considered to be fixed and are trained by assuming a loss function that depends on the prediction errors and on the update errors as well.

In Equation (4.4.19) we present, with some adaptations in the notation, the model used by [Coskun et al., 2017].

$$\begin{aligned} \boldsymbol{\alpha}_t &= f(\boldsymbol{\alpha_{t-1}}) + \boldsymbol{\eta}_t, \\ \mathbf{y}_t &= \boldsymbol{\alpha}_t + \boldsymbol{\epsilon}_t, \\ \boldsymbol{\eta}_t &\sim N(0, \mathbf{Q_t}), \\ \boldsymbol{\epsilon}_t &\sim N(0, \mathbf{H_t}). \end{aligned} \tag{4.4.19}$$

In their model, f is a LSTM, which they name as $LSTM_f$. The estimates of Q_t and H_t,

$\hat{Q}_t$ and $\hat{H}_t$, are the output of $LSTM_Q$ and $LSTM_H$, respectively. To estimate the model they apply the Extended Kalman Filter.

The algorithm they apply is the following: they start from the previous prediction $\mathbf{a}_t$ and feed it to the $LSTM_f$, which outputs an intermediate value $\mathbf{a}'_t$ that is fed to $LSTM_Q$ to produce $\hat{Q}_t$. Then, they feed the observation $\mathbf{y}_t$ to $LSTM_H$ which outputs $\hat{H}_t$. Finally, $\mathbf{a}'_t$, $\hat{\mathbf{Q}}_t$ and $\hat{\mathbf{H}}_t$ are used in the equations of the Extended Kalman Filter, by replacing $\mathbf{a}_t$ with $\mathbf{a}'_t$, $\mathbf{Q_t}$ with $\hat{\mathbf{Q}}_t$ and $\mathbf{H_t}$ with $\hat{\mathbf{H}}_t$. At last, from the custom loss function proposed by the authors, they use the backpropagation algorithm to tune the weights of all the LSTMs.

The authors apply this model for human pose estimation, finding very good results. For more details on this model and underlying equations, we refer to the original paper.

Model Description: LSTM-KF

Lastly, we describe the model that we have implemented to estimate the Nelson-Siegel factors across time, and ultimately the future contracts prices. This model has similar components to the ones described thus far, but has unique features and considerations. We will present a detailed explanation of the algorithm for the LSTM-KF using the UKF.

We start from the formulations given in the beginning of the section: Equations (4.4.1) and (4.4.3), that we rewrite below:

$$\begin{aligned} \mathbf{y}_t &= \mathbf{Z_t}(\Theta)\boldsymbol{\alpha}_t + \mathbf{d}_t + \boldsymbol{\epsilon}_t, \\ \boldsymbol{\epsilon}_t &\sim N(0, \mathbf{H_t}), \end{aligned} \tag{4.4.20}$$

$$\begin{aligned} \boldsymbol{\alpha}_{t+1} &= \mathbf{T_t}(\Theta, \boldsymbol{\alpha}_t) + \mathbf{c}_t + \boldsymbol{\eta}_t, \\ \boldsymbol{\eta}_t &\sim N(0, \mathbf{Q_t}). \end{aligned} \tag{4.4.21}$$

For the LSTM-KF, we model $\mathbf{T_t}$ using the Indirect Basic Model LSTM (ILSTM), defined in Section 4.3.2.3. Using Equations (4.3.23) and (4.3.15), we can write the following general algorithm:

1. Start with $\mathbf{a_0}$ and $\mathbf{P_0}$, just as with the normal Kalman Filter.

2. For each timestamp t, we proceed to update the value of the state variable and variance matrix using the UKF update Equations (4.4.9), (4.4.10) and (4.4.11), obtaining the filtered states $\mathbf{a}_{t|t}$ and $\mathbf{P_{t|t}}$.

3. Using the observation $\mathbf{y}_t$, calculating $\mathbf{Z_t^{adj}}$ and $\mathbf{y}_t^{adj}$, and using $\Delta\mathbf{a}^*_{t-j|t-j}, j = 0, ..., P-1$, we fit the ILSTM model for a certain number of epochs. Input and Output pairs can be given independently to the fit function of the LSTM, so that we can make additional fits to the model, as more data becomes available. The values $\Delta\mathbf{a}^*_{t-j|t-j}, j = 1, ..., P$ can be calculated from the series of $\mathbf{a}_{t|t}$ until the current timestamp. To scale the data, we use the series of $\boldsymbol{\alpha}_t$ extracted from the OLS Approach, for $t \in [0, n_{train}]$. In practice, what our ILSTM model is doing is the following: given the series of updated (or filtered) values of the state variable $\mathbf{a}_{t|t}$ until this timestamp, we want to predict the value $\mathbf{a}_{t+1}$ that will produce $\hat{\mathbf{y}}_{t+1}$ with the least error possible. Of course, in order to calculate this error and use it to fit the model, we need the value of $\mathbf{y}_{t+1}$, that will only be available in the next timestamp. That is why we fit the model for this error only in this next timestamp. Notice also that, because of the need of a certain number of timesteps to fit the model, it

is not possible to apply this step in the beginning of the loop, while $t \leq P$. Besides, the Fit Step does not need to be executed for every timestep, it is possible to apply this step only after a certain number of timesteps, accumulating the data.

4. After fitting the LSTM model, we use it to produce the next estimate $\mathbf{a}_{t+1}$ and also the estimate of the variance matrix $\mathbf{P_{t+1}}$, by applying the prediction formulas of the UKF, given by Equations (4.4.12) and (4.4.13). In summary, we generate $2m+1$ sigma-points, and use them to calculate $\Delta\mathbf{a}^*_{t|t}$, which will be used together with $\Delta\mathbf{a}^*_{t-j|t-j}, j = 1, ..., P-1$ and $\mathbf{Z^{adj}_{t+1}}$ as inputs for the LSTM Model, that will output $2m+1$ values that, after re-scaled, will be multiplied by the sigma-weights to generate $\mathbf{a}_{t+1}$ and consequently P_{t+1}. As we also need a certain number of timesteps to produce this estimate, during the initial timestamps, we consider that $\mathbf{a}_{t+1} = \mathbf{a}_{t|t}$ and $\mathbf{P_{t+1}} = \mathbf{P_{t|t}} + \mathbf{Q_t}$.

5. We re-do the last 3 steps for $t = 0$ until $t = n_{train}$. We can repeat this more times in this range to better fit the model. After that, we proceed to apply the model for the rest of the timestamps until $t = n$, without repeating, as we don't want future data to contaminate the predictions generated during the test phase.

The diagram for the LSTM-KF algorithm is displayed in Figure 4.5 below.

After running a subset of the model with the UKF and the EKF, we preferred to use the UKF for two reasons: first, it has been shown in the literature to generate better estimates, as already discussed; and, second, because it is faster, as the calculation of the Jacobian of the LSTM takes much more time than predicting the $2m+1$ sigma-points. Besides, we can speed up these predictions by doing them simultaneously inside the model. We have not tested the Particle Filter, as it needs a lot of particles and thus a lot of predictions from the LSTM model, which would be computationally very expensive, although it could also have been applied in this context, especially by parallelizing the model with GPUs.

As a side speed consideration, we point to the fact that, in both cases, we can fit the model in batches by accumulating data and refitting the model only from time to time. Utilizing this technique with batches of size 10, for instance, the model's performance in relation to computation time has improved 5-fold.

We tried to, instead of repeating the loop for the train model lots of times, speed up the process by warming-up the LSTM Model using the OLS Approach data for a certain number of epochs, and then plugging it to the LSTM-KF. By doing this, we would reduce a lot the number of repetitions necessary to fit the model correctly. However, this has negatively affected the forecasting ability of the model, and thus not used.

In this model, the fitting parameters Θ_F will include the fitting parameters of the ILSTM model, $\mathbf{H}$ and $\mathbf{Q}$ - which will, once again, be considered constant. As for the hyperparameters Θ_H, in addition to the hyperparameters of the ILSTM model - number of time steps, number of layers, number of units, number of epochs and batch size - we also have the fit window - which is the size of the batch of data to accumulate before running the Fit Step - the fit epochs - which is the number of epochs used during the Fit Step and the total epochs - which is the number of times the filter run through the train set.

Finally, we make considerations regarding the estimation of the parameters λ, $\mathbf{H}$ and $\mathbf{Q}$ of the model. Although it is possible to maximize the log-likelihood to find these parameters, in practice this would consume too much time, as a single run of the filter can take up to 6 minutes for the data set we are considering. Taking this into account, we need to set the values of these parameters by ourselves. For λ, we use the value obtained in the OLS Approach. For the estimation of $\mathbf{H}$ and $\mathbf{Q}$, we tried two options. The first one was to calculate the variance

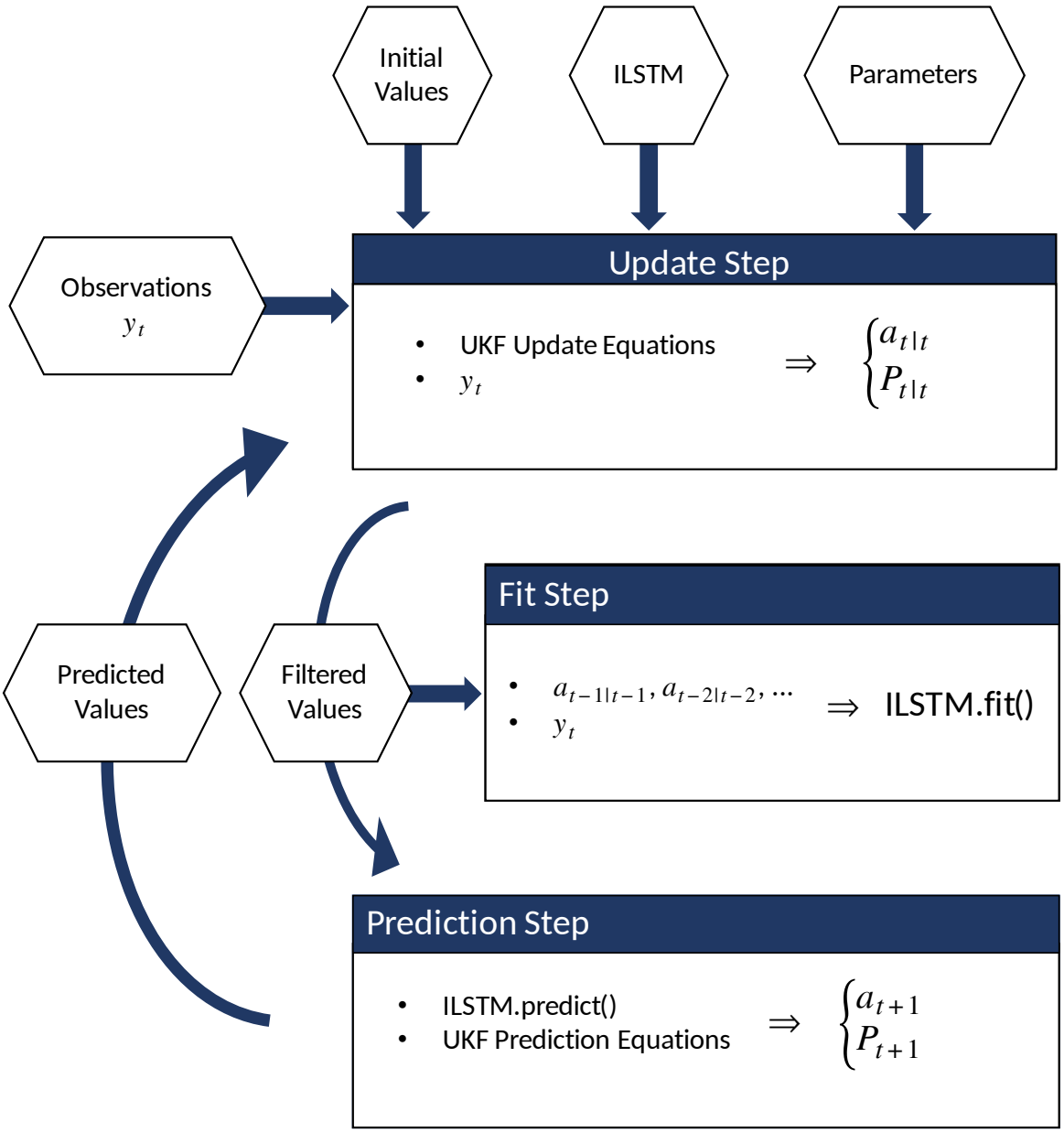

Figure 4.5: Diagram of LSTM-KF.

matrix of errors of the observations and state variables calculated from the OLS Approach with the LSTM Indirect State Model, using only train data, and then multiplying by a scalar. We have tested different values for these scalars, in order to produce a good fitting for the model. The second one is to use the matrices estimated with the normal Kalman Filter. A third option would be to consider these matrices to be the output of other LSTMs, as considered in [Krishnan et al., 2015] and [Coskun et al., 2017].

4.5 Basic Model

We finally describe the Basic Model, which has a similar design to the Direct State Model, described earlier. With the Basic Model, we want to test whether forecasting the term structure without modeling it with hidden factors can generate better results. It is not directly used with

the OLS Approach, but will serve as a benchmark for the other models. The main problem with the Basic Model is its high dimensionality, which adds a lot of noise to it, and increases a lot the number of parameters to be trained.

The Basic Model has the same structure of the Direct State Model, with the difference being the Input Component, or Input Layer and the Output Layer as a consequence. In the Basic Model, the input at a time t and time step j is the $2p \times 1$ vector given by the following equation:

$$X_t^{(j)} = \begin{bmatrix} \Delta \mathbf{y}^*_{t-P+j,\tau_1} \\ \Delta \mathbf{y}^*_{t-P+j,\tau_2} \\ \vdots \\ \Delta \mathbf{y}^*_{t-P+j,\tau_p} \\ \tau_1^* \\ \tau_2^* \\ \dots \\ \tau_p^* \end{bmatrix}. \tag{4.5.1}$$

Hence, X_t has a dimension of $P \times 2p$ where $2p$ is the number of features and P is the number of time steps. Notice that we also feed the values of the times to maturity of the contracts alongside the variation of the price, as this is an important value for the price calculation that, in the other models, is already taken into account. We also highlight the fact that the scaling of Δy and τ are made independently from each other. In fact, the scaling is independent for each of the $2p$ features. This is also true for the Direct State Model and the Indirect State Model.

The output of the model will be denoted by $\widehat{\Delta \mathbf{y}}_t$. We use an asterisk whenever we want to indicate that the corresponding variable is scaled. We can then write:

$$\begin{aligned} \widehat{\Delta \mathbf{y}}^*_{t+1} &= \texttt{BasicModel}(X_t), \\ \hat{\mathbf{y}}_{t+1} &= D(\mathbf{b}_{\Delta \mathbf{y}}) \widehat{\Delta \mathbf{y}}^*_{t+1} + \mathbf{a}_{\Delta \mathbf{y}} + \mathbf{y}_t, \end{aligned} \tag{4.5.2}$$

where $\mathbf{a}_{\Delta \mathbf{y}}$ and $\mathbf{b}_{\Delta \mathbf{y}}$ are given by the scaling formula (4.2.2) for $\Delta \mathbf{y}$, considering only training data, while $D(.)$ is given in Equation (4.2.2).

The layer graph is the same as for the Direct Stade Model.

In the case of the Basic Model, $\Theta_M = \emptyset$. The fitting parameters Θ_F will be the fitting parameters of the underlying LSTM model, while the hyperparameters Θ_H will be the hyperparameters of the underlying LSTM model.

Chapter 5

Results

5.1 Introduction

In this chapter we present the results for fitting and forecasting the models discussed in Chapter 3 and using the methods described in Chapter 4.

For each case, as it will be further explained in details, the choice for parameters is based on error metrics in train and validation data.

To compare the models, we have used two main metrics: Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We make the comparison for both the entire curve at once and for each vertex at a time.

We will start by presenting the results of the fitting of the dynamic Nelson-Siegel and the Schwartz-Smith three-factor models, calculating the fit error and exploring the characteristics of the state variables. We will also show the comparison between the two models. We have found out that the first model was better at fitting the data, with practically half the RMSE. Besides, two of the state variables of the models are highly correlated.

After that, we will give further details on the forecasting of each model and will present the comparison of the forecasting errors, while discussing the results. We will also briefly explore the results of a rolling validation. The dynamic Nelson-Siegel model with LSTM and ILSTM were the best models considering RMSE, MAE and robustness.

For all the results obtained in this chapter, the year of 2020 was not considered due to possible noise in the data because of the pandemics. With that, the total data used for Brent has length of $n = 5120$ days, while for WTI the total data length was $n = 5021$. We separated 20% of data for validation and testing, so that $n_{validation} = 512$ and $n_{test} = 512$ for Brent and $n_{validation} = 502$ and $n_{test} = 502$ for WTI.

5.2 Model Fitting and Factors Analysis

5.2.1 Dynamic Nelson-Siegel model

In order to calculate the parameters of the dynamic Nelson-Siegel, we have generated 10 random initial values between 0 and 5 for λ and optimized the error from the least squares regression over the train data. For all of the initial values, the optimization process ended with the same value for λ.

The calculated value of λ for Brent was approximately 1.98 and for WTI was approximately 2.7. We plot in Figures 5.1 and 5.2 the RMSE of the least squares regression across all contracts,

obtained by running the regression with different values of λ. We also plot in red the calculated value using the method mentioned.

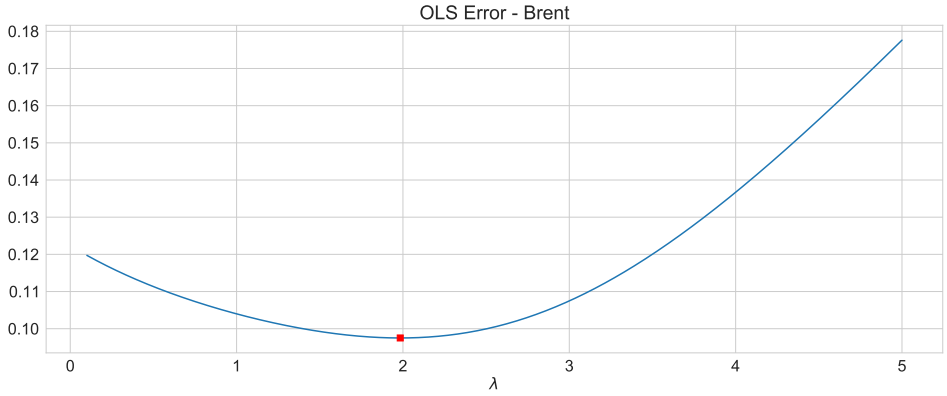

Figure 5.1: OLS Error vs λ for Brent. It is clear that the optimization was successful.

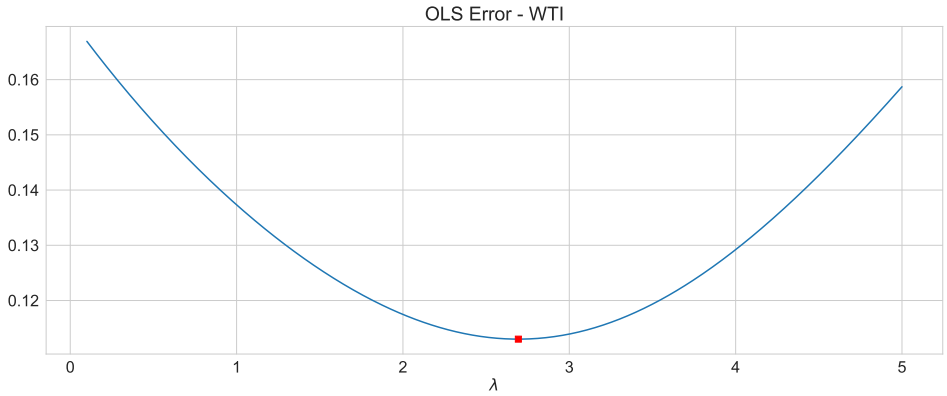

Figure 5.2: OLS Error vs λ for WTI. It is clear that the optimization was successful.

Although we fixed the value of λ, its dynamic behaviour can be seen in Figures 5.3 and 5.4, where we plotted the calculated value of λ for a rolling window of 800 days (a little more than 3 years). This indicates that we might improve our model by considering λ to be dynamic as well. An interesting note about the two dynamic λ time series is that they show a very similar behaviour. The total correlation between them was 0.96.

After fixing the value of λ, we calculated the dynamic Nelson-Siegel factors using the OLS approach. These values can be seen in Figures 5.5 and 5.6 for Brent and WTI respectively.

The total fit RMSE across the entire time series (including the out of sample data) was approximately 0.091 for Brent and 0.107 for WTI.

Using the Kalman Filter, we obtained very similar results, but slightly worse. The RMSE was approximately 0.107 for Brent and 0.110 for WTI. In Figures 5.7 and 5.8 we show the factors time series.

In Figure 5.9 and 5.10 we plot the fitted curve using OLS against the actual curve at randomly selected dates.

After that, we briefly explore some characteristics of the state variables time series obtained by the OLS approach - which were very similar to the ones obtained with the Kalman Filter.

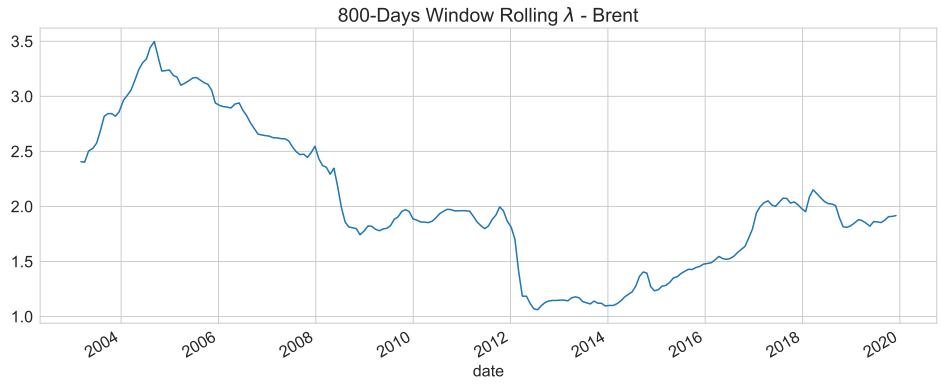

Figure 5.3: 800-Day Window Rolling value of λ for Brent. It shows that the value of λ may change depending on the test set, which suggests that a dynamic treatment may be advantageous.

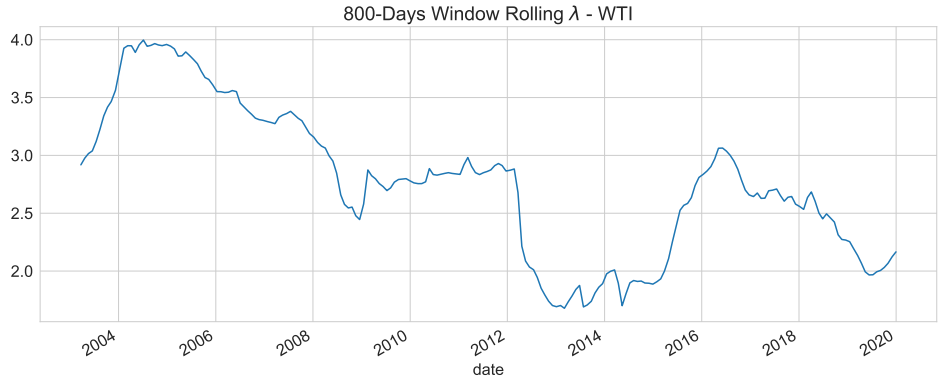

Figure 5.4: 800-Day Window Rolling value of λ for WTI. It shows a similar behaviour to Brent.

We started by applying the Augmented Dickey-Fuller test to the Level, Slope and Curvature factors, as well as to their first difference series, which we refer as Delta Level, Delta Slope and Delta Curvature. The results can be seen in Table 5.1.

For both Brent and WTI, the test suggests that only the Level factor has a unit root and is non-stationary. We decided to use the first difference of the factors for the forecasting, although we could have used the Slope and Curvature factors as they are. Additionally, it could also be possible to apply a fractional derivative to the Level factor. More specifically, we found out that a fractional derivative of 0.4 was already enough to make the data stationary. This analysis will not be the focus here, but could make part of an expansion study.

In Figures A.1, A.2, A.3, A.10, A.11 and A.12, we show the ACF (auto-correlation function) and PACF (partial auto-correlation function) plots of the Delta Level, Delta Slope and Delta Curvature for Brent and WTI. In general, it is possible to see that there are some lags there are slightly significant, specially among the first four legs. We also show these plots for the square of the factors in figures A.4, A.5, A.6, A.13, A.14 and A.15. In this case, a lot of lags showed a great degree of significance, specially for the Level factor. This indicates that there is a strong auto-regressive pattern for the volatility of the factors, and that we could apply a model with a stochastic volatility.

Finally, we also show the Q-Q (quantile-quantile) plots of the Delta Level, Delta Slope and

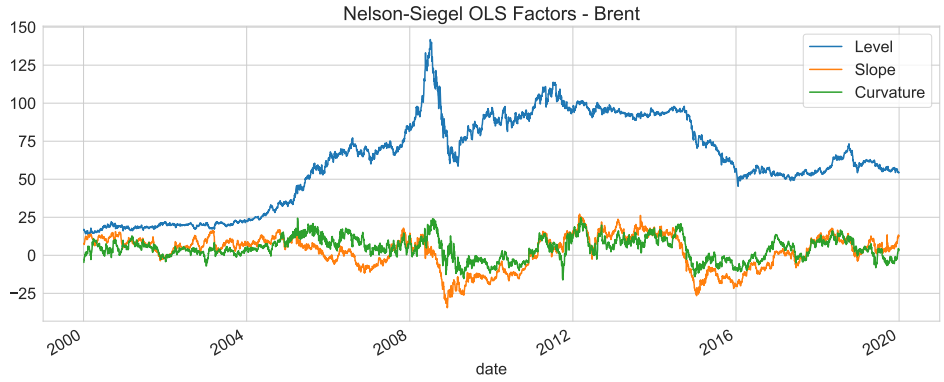

Figure 5.5: Time series of the OLS factors ($\boldsymbol{\alpha}$) from the dynamic Nelson-Siegel for Brent.

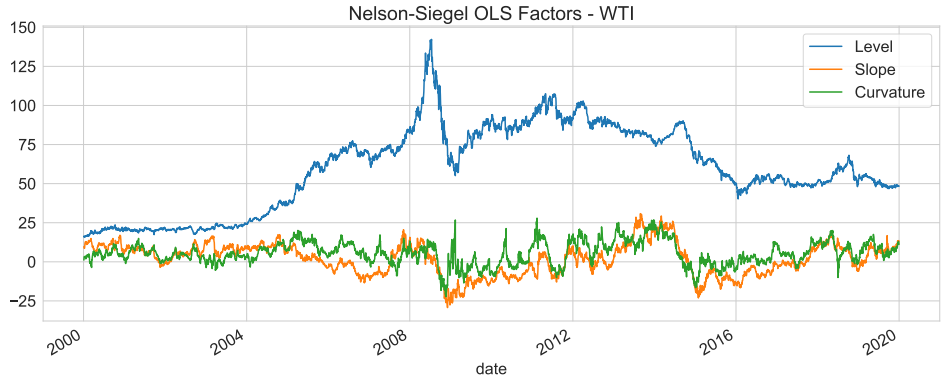

Figure 5.6: Time series of the OLS factors ($\boldsymbol{\alpha}$) from the dynamic Nelson-Siegel for WTI.

Delta Curvature in Figures A.7, A.8, A.9, A.7, A.8 and A.9 for Brent and WTI. They all display a similar shape, with heavy tails. This indicates that we could improve the model by considering a different distribution for the errors, for example including a jump process.

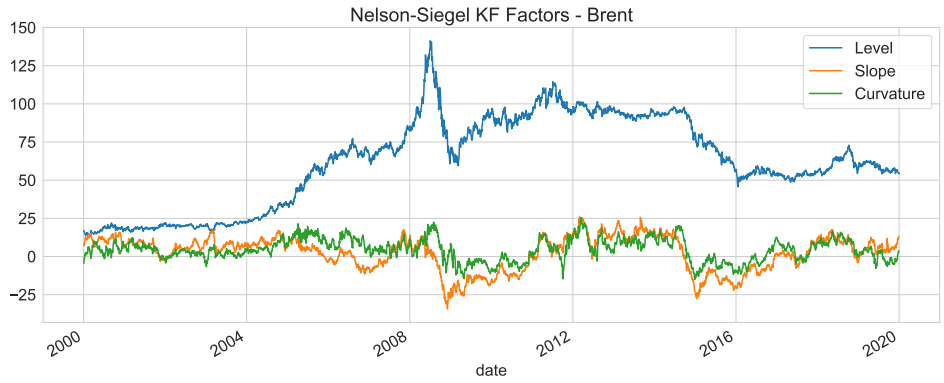

Figure 5.7: Time series of the KF factors ($\boldsymbol{\alpha}$) from the dynamic Nelson-Siegel for Brent.

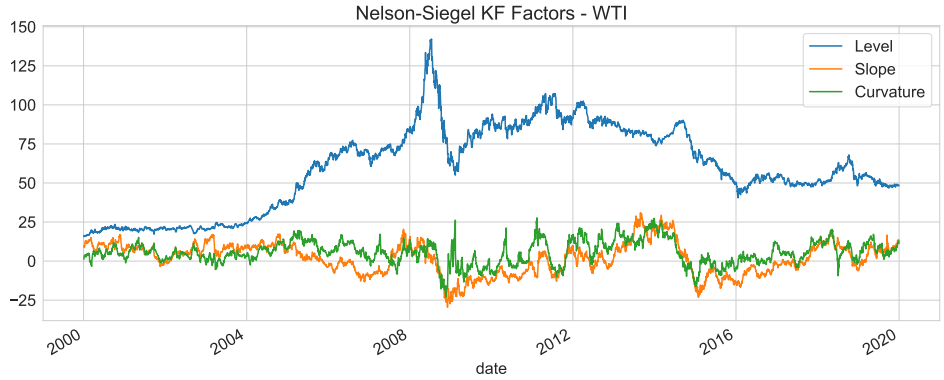

Figure 5.8: Time series of the OLS factors ($\boldsymbol{\alpha}$) from the dynamic Nelson-Siegel for WTI.

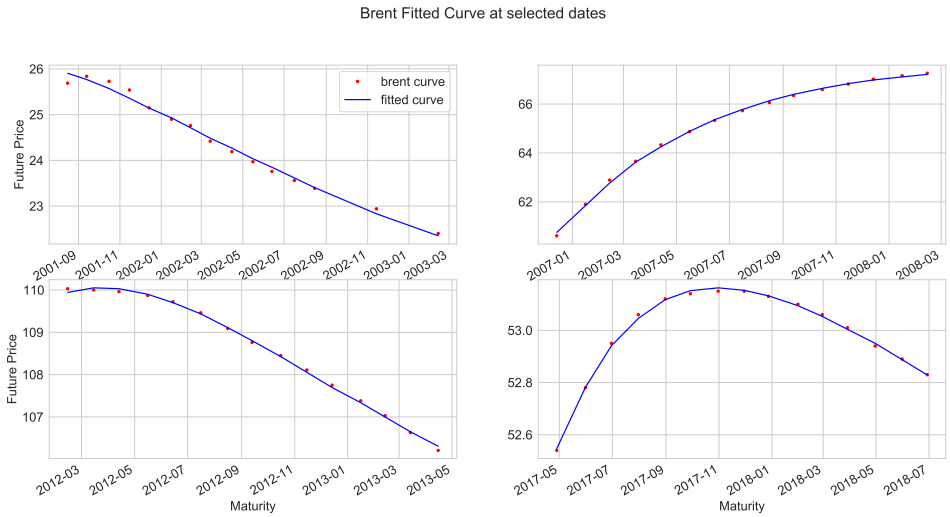

Figure 5.9: Dynamic Nelson-Siegel OLS Approach fitted curves versus real curve for Brent for randomly selected dates. These are the same dates selected in Section 2. We can see that the curve fitted satisfactorily the prices.

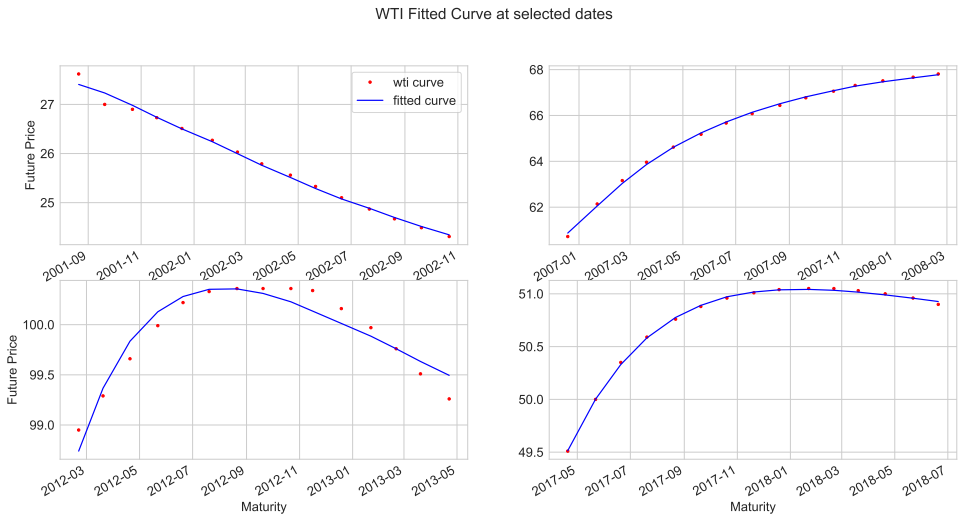

Figure 5.10: WDynamic Nelson-Siegel OLS Approach fitted curves versus real curve for WTI for randomly selected dates. These are the same dates selected in Section 2. We can see that the curve fitted satisfactorily the prices, except for the third one.

Series	Brent		WTI	
	ADF-Statistic	p-value	ADF-Statistic	p-value
Level	-1.71	0.428	-1.85	0.358
Slope	-3.12	0.025	-3.64	0.005
Curvature	-5.23	0.00	-5.60	0.00
Delta Level	-20.95	0.00	-12.45	0.00
Delta Slope	-23.75	0.00	-50.95	0.00
Delta Curvature	-19.00	0.00	-15.40	0.00

Table 5.1: ADF Test for Brent and WTI OLS dynamic Nelson-Siegel factors

5.2.2 Schwartz-Smith three-factor model

For the Schwartz-Smith three-factor model, we have also applied the OLS and Kalman Filter to fit the data. Starting with the OLS approach, we generated ten random samples of the parameters in ranges that we judged optimal, considering the usual values obtained in the literature. For each sample we have run the optimization to find the best parameters. Initially we were getting errors during the linear regression part due to numerical instability, specially for some ranges of values for WTI. This was overcome by calculating the linear regression using the Moore-Penrose pseudo-inverse matrix.

In the case of the Kalman Filter method, the optimization would sometimes result in error using the L-BFGS-B algorithm so the Powell algorithm was preferred here. Similarly to the OLS approach, we have also generated ten random samples for the parameters and run the optimization for each of them, keeping the one that maximized the log-likelihood over the train data.

We ran each method multiple times, until the RMSE of the test data stabilized, choosing the final result with smaller error.

In Figures 5.11 and 5.12 we show the factors extracted using the OLS and Kalman Filter respectively for Brent. And in Figures 5.13 and 5.14 we show the factors for WTI.

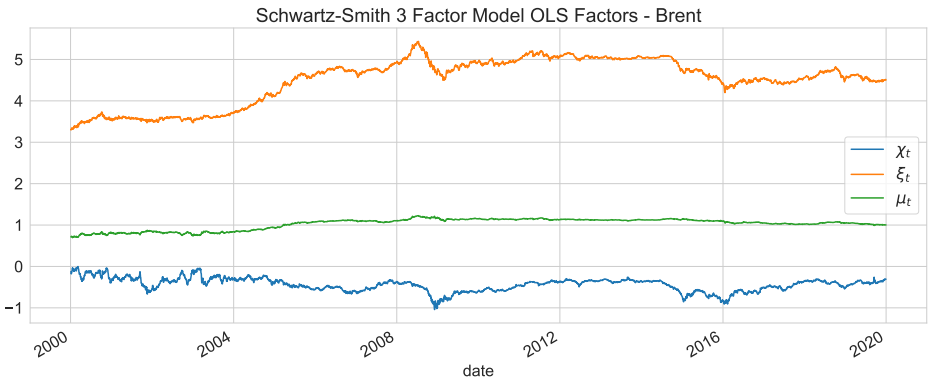

Figure 5.11: Time series of the OLS factors ($\boldsymbol{\alpha}$) from the Schwartz-Smith three-factor model for Brent.

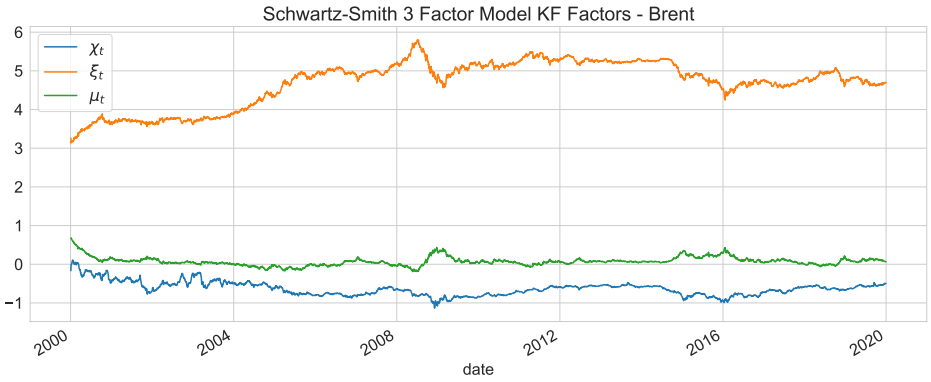

Figure 5.12: Time series of the KF factors ($\boldsymbol{\alpha}$) from the Schwartz-Smith three-factor model for Brent.

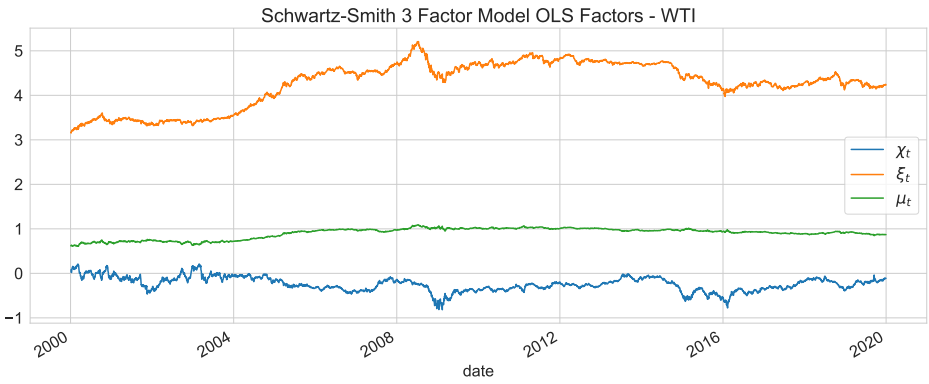

Figure 5.13: Time series of the OLS factors ($\boldsymbol{\alpha}$) from the Schwartz-Smith three-factor model for WTI.

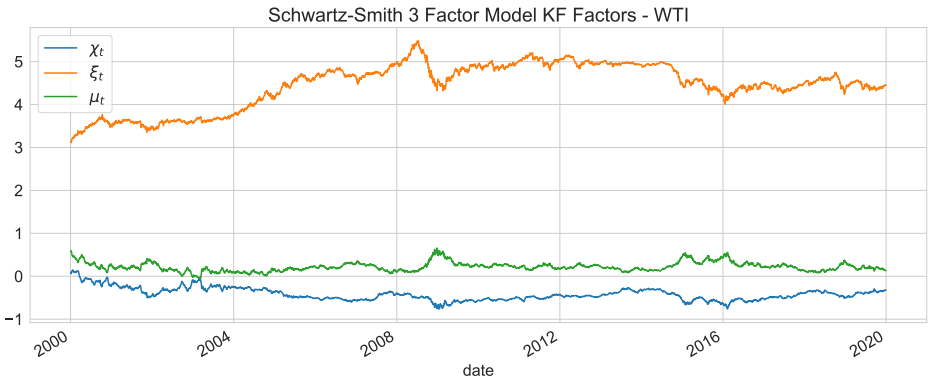

Figure 5.14: Time series of the KF factors ($\boldsymbol{\alpha}$) from the Schwartz-Smith three-factor model for WTI.

The total fit RMSE for Brent was 0.200 using the OLS approach and 0.188 using the Kalman Filter. As for the WTI, the fit error was 0.278 and 0.251 using the OLS and Kalman Filter respectively. As it happened with the other model, the OLS was better at fitting the data.

Differently from what occurred with the dynamic Nelson-Siegel, one of the factors - μ_t - extracted using the OLS was very different from the one extracted with the Kalman Filter. It is also interesting to note that the factors ξ_t and χ_t are very similar to the level and slope factors from the dynamic Nelson-Siegel model. In the next Section we will compare the two models in more details.

5.2.3 Model Comparison

We start by comparing the total fit RMSE for each of the models used: dynamic Nelson-Siegel with OLS or Kalman Filter (KF) and Schwartz-Smith three-factor model (SS for simplicity) with OLS or KF. In Figures 5.15 and 5.16 we show the total fit RMSE for Brent and WTI respectively.

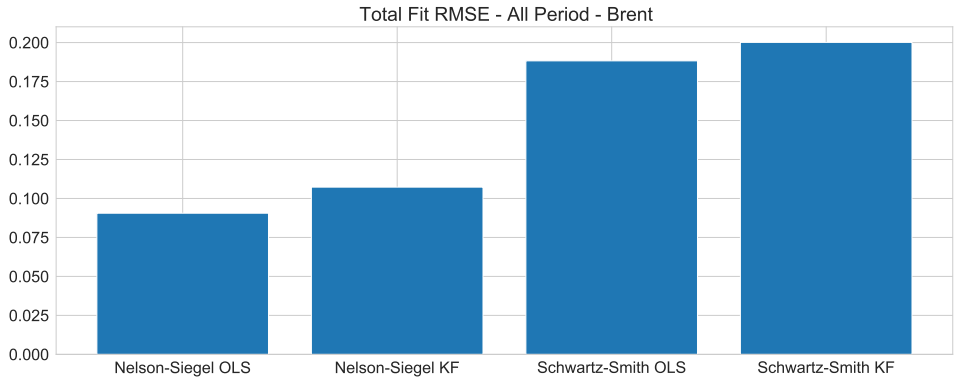

Figure 5.15: Total Fit RMSE - All Period - Brent.

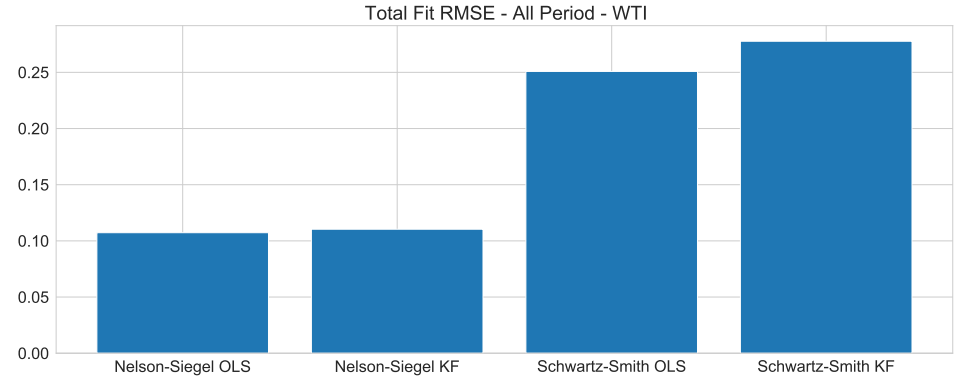

Figure 5.16: Total Fit RMSE - All Period - WTI.

In terms of fitting the data, the dynamic Nelson-Siegel exhibited much better results for both Brent and WTI. This can also be seen in the comparison of the fit RMSE for each one of the 15 rolling contracts in Figures 5.17 and 5.18 for Brent and WTI respectively.

The largest difference between the two models occurs in the first and last contracts. Specially the Schwartz-Smith model with Kalman Filter has exhibited a very large fit RMSE for the first contract of 0.8 approximately. The dynamic Nelson-Siegel model has shown a more constant behaviour along the term structure. Taking into consideration that two of the factors are very similar between the two models, it seems that the curvature factor considered in the dynamic Nelson-Siegel model was much better in improving the model than μ_t. Furthermore, the inclusion of another curvature factor as in the Svensson model [Svensson, 1994] could improve even more the accuracy.

Lastly, we present the correlation of all the factors for Brent and WTI in Figures 5.19 and 5.20. For clarity, we do not include the factors of the dynamic Nelson-Siegel obtained via Kalman Filter because they were approximately equal to the ones obtained via OLS.

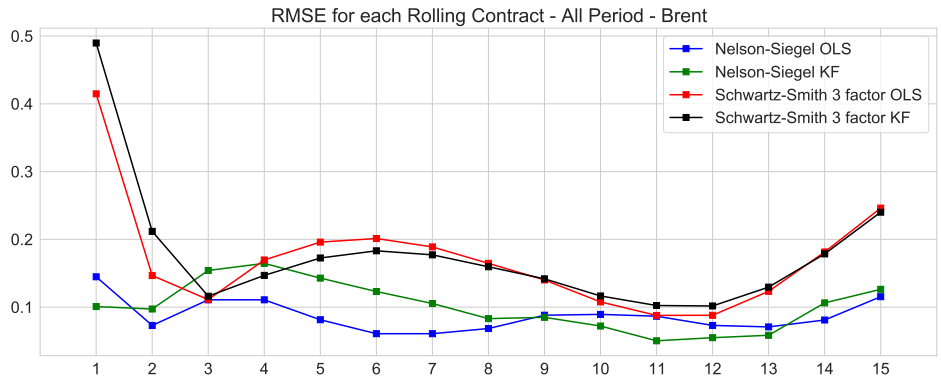

Figure 5.17: Fit RMSE for each Rolling Contract - All Period - Brent.

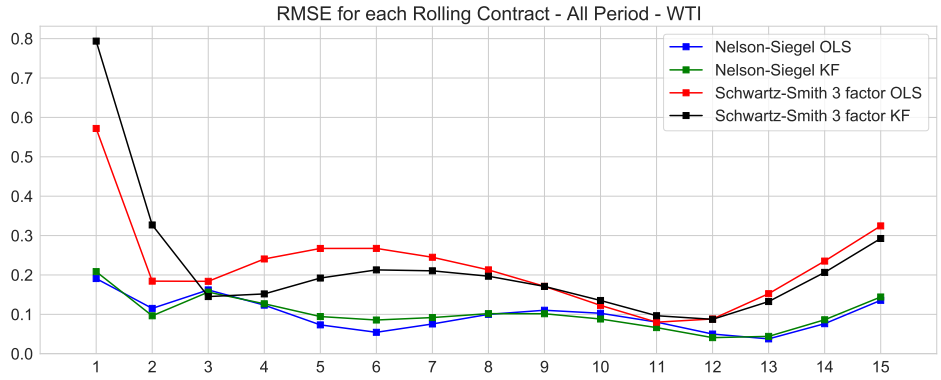

Figure 5.18: Fit RMSE for each Rolling Contract - All Period - WTI.

The results are very similar for Brent and WTI and support some of the conclusions achieved by inspecting the series. First, in the case of the Schwartz-Smith, χ_t and ξ_t presented a correlation of 0.94 and 0.99 between the OLS and KF, but the μ_t factor was completely uncorrelated. The factor μ_t obtained via OLS was very similar to the ξ_t factor with a correlation of 0.97, but also displayed a negative correlation of -0.61 with χ_t. On the other hand, the factor μ_t obtained via Kalman Filter was not much correlated to any of the other factors of the Schwartz-Smith.

Comparing both models, we reach the same conclusion that the level and slope curvature are very similar to ξ_t and χ_t respectively with correlations in the range of 0.75 to 0.96. This roughly mean that the level is explained by the long term price - ξ_t - and that the slope is explained partly by short term price fluctuations - χ_t. This was similar to what [Karstanje et al., 2017] found when comparing the dynamic Nelson-Siegel and the Gibson-Schwartz two-factor model (which is equivalent to the Schwartz-Smith two-factor model). It is also interesting to see that the curvature factor shows significant correlation of 0.42 with χ_t from the OLS and -0.54 with μ_t from the Kalman Filter. Besides, μ_t from the KF also showed a significant correlation of -0.45 with the slope factor.

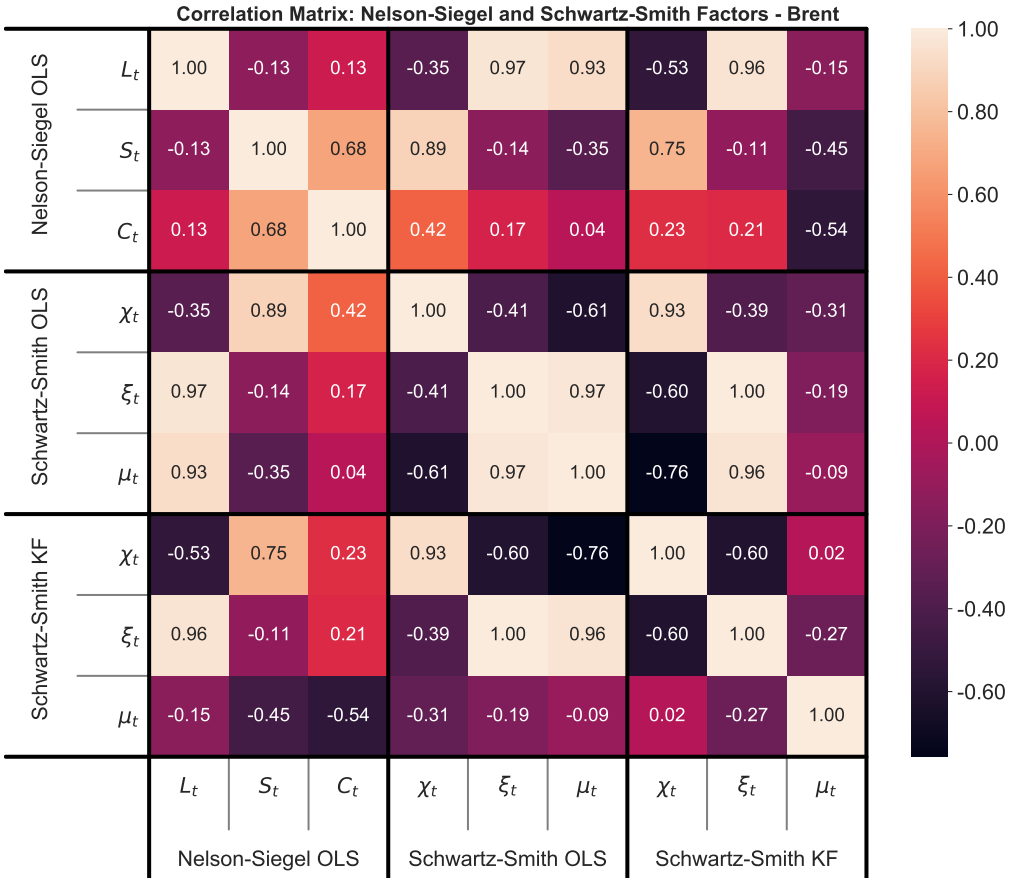

Figure 5.19: Correlation between dynamic Nelson-Siegel and Schwartz-Smith OLS and KF factors - Brent.

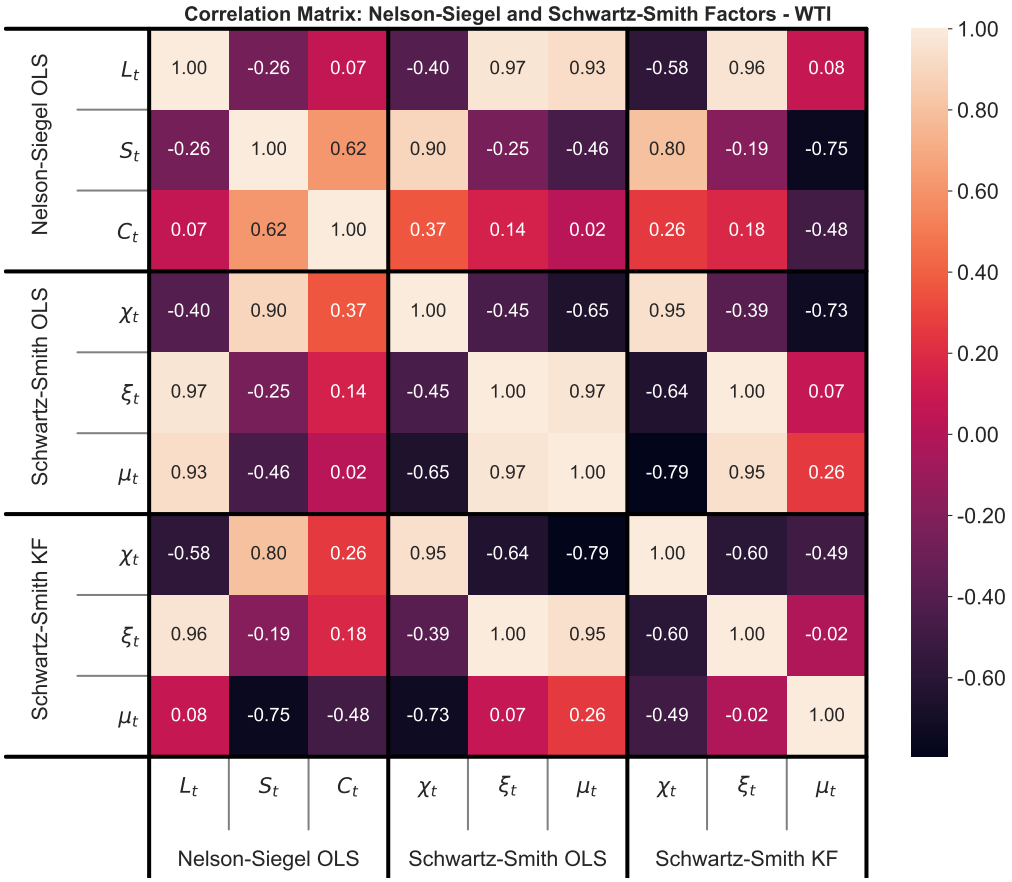

Figure 5.20: Correlation between dynamic Nelson-Siegel and Schwartz-Smith OLS and KF factors - WTI.

5.3 Forecasting

5.3.1 Introduction

In this section, there will be presented the one day ahead forecasting results using all the methods described. For the dynamic Nelson-Siegel, we have used Random Walk (NS-RW): which will serve as the Benchmark; VAR (NS-VAR); Kalman Filter (NS-KF); LSTM (NS-LSTM); ILSTM (NS-ILSTM: LSTM with Indirect State Model) and finally LSTM-KF (NS-LSTM-KF). We will not include the dynamic version of VAR, as the results were all similar or slightly worse than the original version. We will also show the results for the forecasting with the Schwartz-Smith three-factor model using the OLS approach (SS-OLS) and the Kalman Filter (SS-KF). With these results, we can compare how well does the parametric model do against a more traditional risk-neutral based approach. Finally, we also include the results for the forecasting with the Basic Model (BM), which will represent how well can a LSTM model forecast the curve without extracting factors first, or, in other words, if there is any actual gain in doing this extraction before-hand. Note that we have included in parenthesis the label for each one of the methods to identify them in the tables and charts that will be displayed. We also show below in Table 5.2 a summary containing the description of each label for clarity.

Model	Estimation Procedure	Label
Dynamic Nelson-Siegel	Random Walk	NS-RW
	VAR	NS-VAR
	Kalman Filter	NS-KF
	LSTM - Direct State Model	NS-LSTM
	LSTM - Indirect State Model	NS-ILSTM
	LSTM with Kalman Filter	NS-LSTM-KF
Schwartz-Smith three-factor	OLS Approach	SS-OLS
	Filtering Approach	SS-KF
Basic Model	LSTM - Basic Model	BM

Table 5.2: Labels for the models and estimation procedures

5.3.2 Model Tuning

Before going directly to the results, we will also make some comments on the methods with hyperparameters in regards to tuning the models, i.e., how were these values calculated. The remaining methods are straightforward and were already explained in previous chapters.

In the Appendix, we display the detailed hyperparameters estimated for each model.

5.3.2.1 Dynamic Nelson-Siegel with VAR

Starting with the NS-VAR, the only hyper-parameter we need to choose is the number of lags. After training each one of the VAR models, we used the validation data to decide which number of lags was best. Using the training result would be catastrophic because of the possibility of overfit. This can be seen in Figures 5.21 and 5.22 below - for Brent and WTI, respectively.

We plot the train and validation forecasting RMSE versus the number of lags. Here, 0 lags means the NS-RW. It is possible to see that the training error decreases monotonically, but the validation error gets actually worse than the NS-RW from 4 lags onward. The same behaviour

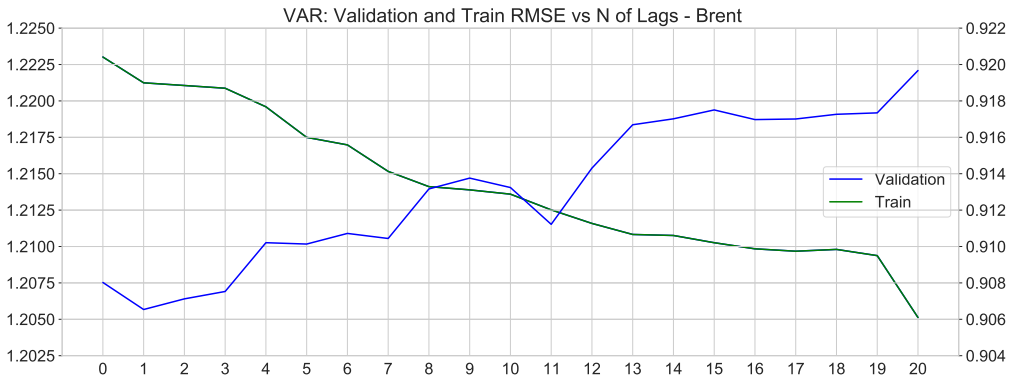

Figure 5.21: VAR over-fit: the train RMSE decreases monotonically, whereas the validation RMSE starts to increase with a higher number of lags - Brent.

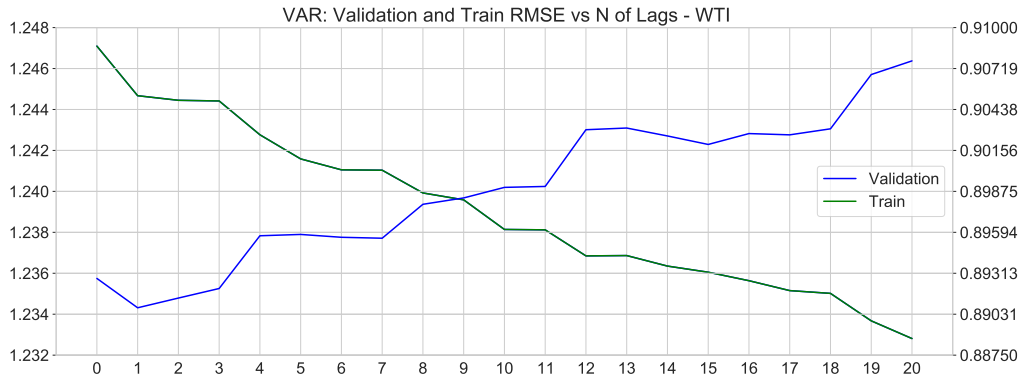

Figure 5.22: VAR over-fit: the train RMSE decreases monotonically, whereas the validation RMSE starts to increase with a higher number of lags - WTI.

occurs to both Brent and WTI. From this result, it was decided that the VAR with 1 lag was the best one.

5.3.2.2 Dynamic Nelson-Siegel with LSTM and ILSTM

Going to the LSTM and ILSTM, there are some hyperparameters to be optimized: the number of time steps, units, layers, epochs, batch size and initialization.

We first tested the number of epochs and batch size. For both Brent and WTI, a batch size of 1 generated consistently better train and validation results with multiple combinations of the other parameters. As for the number of epochs, after multiple tests, it was clear that the error stopped decreasing around the eighth to the tenth epoch, so that we decided to fix the number of epochs as 10.

As for the number of time steps, units and layers, we started from a lot of combinations. At the same time, we avoided, as much as possible, to look at every single combination, while also searching for consistency of results. A great part of the combinations generated good results, i.e., had low validation errors. To choose the final set, we took into account not only the validation error, but also the robustness to parameter variation. We have also taken into account the results from both Brent and WTI. The idea was to find the most robust set of

values for these parameters.

Taken this into account, we fixed, for all models, the number of layers as 1 and the number of units as 10. As for the number of time steps, we decided to use values between 10 and 20, depending on the model.

Finally, to take into account the random initialization, what we have done was to run the model for each set ten times, choosing the one with the smaller validation error. This way, we would be selecting a good initialization for that specific set.

5.3.2.3 Basic Model

A similar approach was used to choose the best parameters for the Basic Model, that turned out to be the same. That is, we used the number of epochs as 10, the batch size as 1, the number of layers as 1, the number of units as 10 and the number of time steps between 10 and 15.

5.3.2.4 Dynamic Nelson-Siegel with LSTM-KF

As for the LSTM-KF, there are additional hyperparameters to set. For the matrices **Q** and **H**, we found out that the most consistent results were achieved by using the ones estimated via Kalman Filter, although the other methods mentioned in section 4.4.5.2 also have good to average results. As for the number of time steps, units and layers, following what was done with the LSTM and ILSTM models, we have set respectively the values of 10, 10 and 1. Other combinations were also tested, but this was one of the most consistent ones, specially because it was already used with other models.

As for the warm up, that is, warming up the ILSTM model before applying the filter, that was discussed in section 4.4.5.2, we decided against using it, as it has not, in general, provided good results.

Finally, we also had to choose the window of days to refit the ILSTM, the number of epochs used at each window and finally the total number of epochs to use during training. A lot of combinations generated good results when looking at the validation RMSE, but the more consistent ones had longer fit window. Taking into account the validation error, and also the training error, while also trying to keep the choices the most consistent possible, we ended with our finals models. For Brent, the fit window was set to 1500, with 10 epochs in total and 5 fit epochs at each refit window. For WTI, the refit window was set to 1000, with 2 epochs in total and 10 fit epochs at each refit window.

Besides, similarly to the LSTM, to take into account the random initialization, what we have done was to run the model for each set three times, choosing the one with the smaller validation error. This way, we would be selecting a good initialization for that specific set.

5.3.3 Results

As already pointed out earlier, there will be presented the RMSE and MAE for the training data, validation data, test data, out of sample data (OOS - comprises the validation and test data) and the total data for all models and methods that were described in this work.

We also present the QQ-Plot and Kernel Distribution of the residuals for each model in the Appendix. In general, all of the residuals display heavy tails, which once more points to the fact that the error distribution is not normal, and this can be improved in future works.

We will start with the results for Brent, and then will present the results for WTI.

In Tables 5.3 and 5.4 we summarize all the results for the entire curve for Brent.

Model	Train RMSE	Validation RMSE	Test RMSE	OOS RMSE	Total RMSE
NS-RW	1.2230	0.9080	1.0942	1.0054	1.1827
NS-KF	1.2231	0.9081	1.0942	1.0054	1.1828
NS-VAR	1.2212	0.9065	1.0906	1.0028	1.1808
NS-LSTM	1.2217	0.9054	1.0909	1.0024	1.1810
NS-ILSTM	1.2231	0.9062	1.0906	1.0027	1.1822
NS-LSTM-KF	1.2039	0.9055	1.0949	1.0047	1.1668
BM	1.2256	0.9031	1.0959	1.0041	1.1846
SS-OLS	1.2374	0.9196	1.1083	1.0184	1.1968
SS-KF	1.2432	0.9131	1.1034	1.0127	1.2007

Table 5.3: Forecast RMSE - Brent

Model	Train MAE	Validation MAE	Test MAE	OOS MAE	Total MAE
NS-RW	0.8227	0.6960	0.7734	0.7347	0.8051
NS-KF	0.8227	0.6960	0.7735	0.7348	0.8051
NS-VAR	0.8201	0.6945	0.7709	0.7327	0.8026
NS-LSTM	0.8204	0.6925	0.7684	0.7305	0.8024
NS-ILSTM	0.8213	0.6917	0.7665	0.7291	0.8028
NS-LSTM-KF	0.8133	0.6927	0.7785	0.7356	0.7977
BM	0.8249	0.6896	0.7689	0.7293	0.8057
SS-OLS	0.8382	0.7078	0.7960	0.7519	0.8209
SS-KF	0.8501	0.7042	0.8024	0.7533	0.8308

Table 5.4: Forecast MAE - Brent

For better visualization, we show the results for test data, out of sample data and total data in a form of a chart in Figures 5.23 for RMSE and 5.24 for MAE. We also show them in form of a ratio against the NS-RW method, which is the baseline for all the other methods

In general, it is possible to conclude that some of the methods based on the dynamic Nelson-Siegel were slightly better than the NS-RW. It is interesting to note that the Basic Model showed similar results to the NS-RW, but slightly worse, specially compared to the NS-LSTM and NS-ILSTM, which all use an LSTM as the base for prediction. From this, we can conclude that converting the price into factors before the prediction step is in fact advantageous.

On another hand, the methods based on the Schwartz-Smith three-factor model produced predictions much worse than the ones from the dynamic Nelson-Siegel based methods. This was somewhat expected, given that the fit error was much larger for them. Actually, the results were even a positive surprise because these methods could provide a very good prediction given the difficulty of the model to fit correctly the curve.

Taking into account the RMSE, the NS-LSTM and NS-ILSTM methods were the best overall, followed closely by the NS-VAR. The NS-LSTM-KF had an impressive training RMSE and one of the best validation RMSE, but was not very good in the test data. As for the NS-KF, the results were very similar to the NS-RW, with almost no difference.

Now for the MAE, the NS-LSTM and NS-ILSTM were the best models with a good margin, specially for the test and out of sample data, with almost 1% improvement for the NS-ILSTM.

We also present the RMSE errors for each maturity. For this, we separate the models/methods in two groups: part I - containing the NS-RW, NS-VAR, NS-KF, NS-LSTM, NS-ILSTM, NS-LSTM-KF and BM - and part II - containing the NS-RW, NS-VAR, SS-OLS and SS-KF. We repeat the NS-RW and NS-VAR for reference between the groups. We first show the results

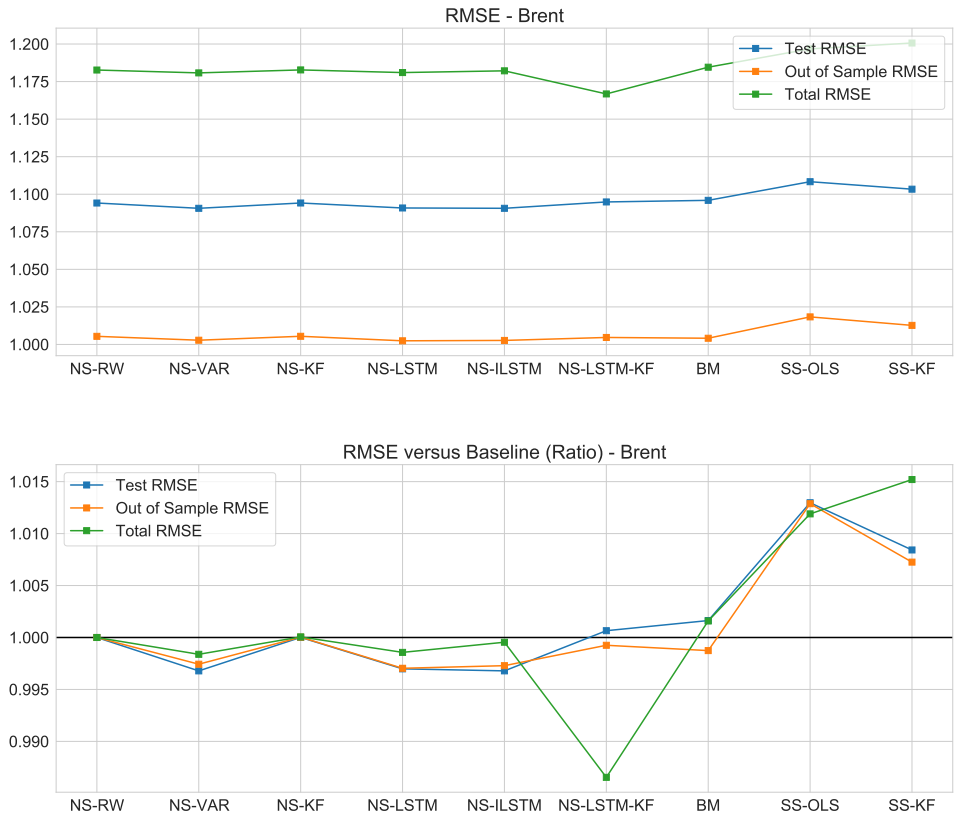

Figure 5.23: (top) RMSE - Brent. (bottom) RMSE versus Baseline (Ratio) - Brent. Test RMSE refers to the RMSE for the test data; Out of Sample RMSE refers to the RMSE for the test data plus validation data, that is, all the data that does not belong to training data; Total RMSE refers to all of the data, including training data and out of sample data.

as a ratio versus the NS-RW for better visualization, and then present the absolute values, in order to see the shape of the errors curve. These are shown in Figures 5.25, 5.26, 5.27 and 5.28 for the test data and in Figures 5.29, 5.30, 5.31 and 5.32 for the total data.

From these results, it is possible to see that the prediction error is larger for the first contracts for all models and methods analysed. The NS-ILSTM, followed by the NS-LSTM and NS-VAR were the ones that were able to best predict the first contract prices in the test data, with almost 0.8% improvement, in comparison to the NS-RW in the case of the NS-ILSTM. From these methods, the NS-VAR has shown the most consistent improvement against the Baseline across the curve. Contrarily, the BM was the method with the most difficulty in predicting the first contract for the methods in part I. For the models in part II, the SS-OLS presented a terrible test RMSE for the first contract, while the SS-KF was more stable, being worse in the second half of the curve.

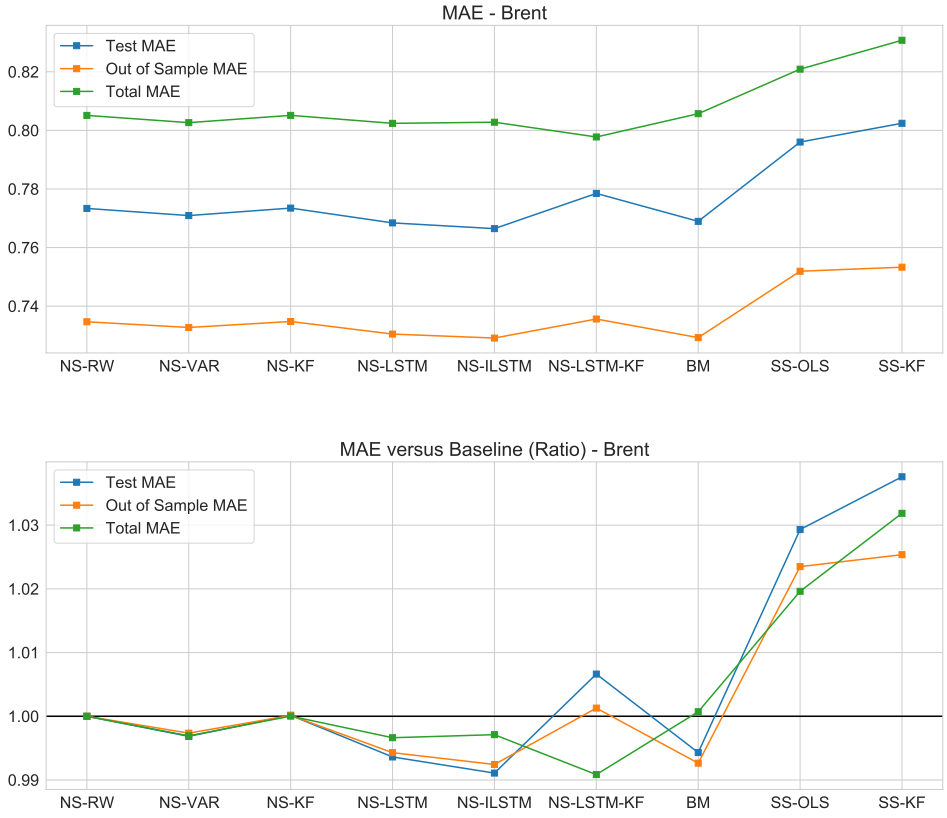

Figure 5.24: (top) MAE - Brent. (bottom) MAE versus Baseline (Ratio) - Brent. Test MAE refers to the MAE for the test data; Out of Sample MAE refers to the MAE for the test data plus validation data, that is, all the data that does not belong to training data; Total MAE refers to all of the data, including training data and out of sample data.

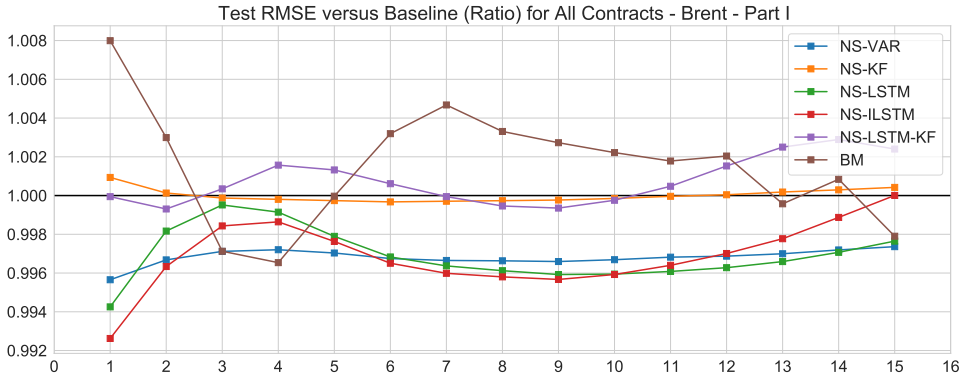

Figure 5.25: Test RMSE per Rolling Contract versus Baseline (NS-RW) - Part I - Brent.

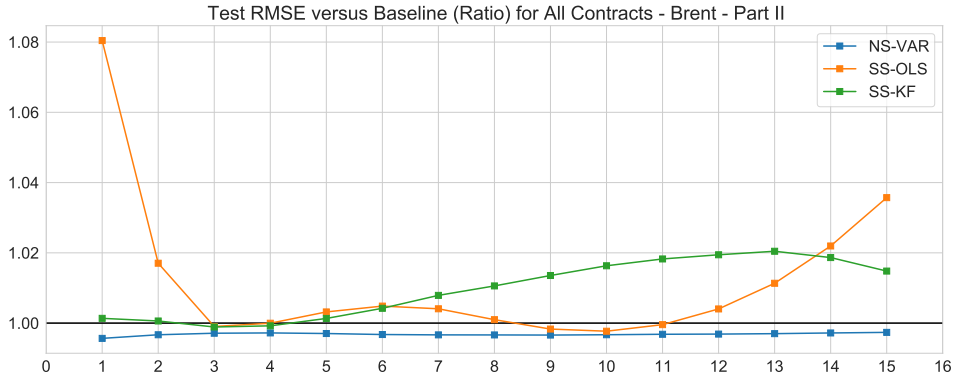

Figure 5.26: Test RMSE per Rolling Contract versus Baseline (NS-RW) - Part II - Brent.

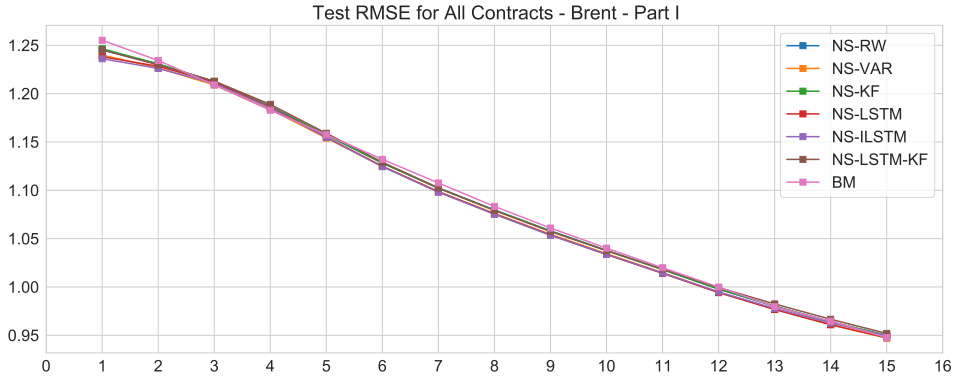

Figure 5.27: Test RMSE per Rolling Contract - Part I - Brent.

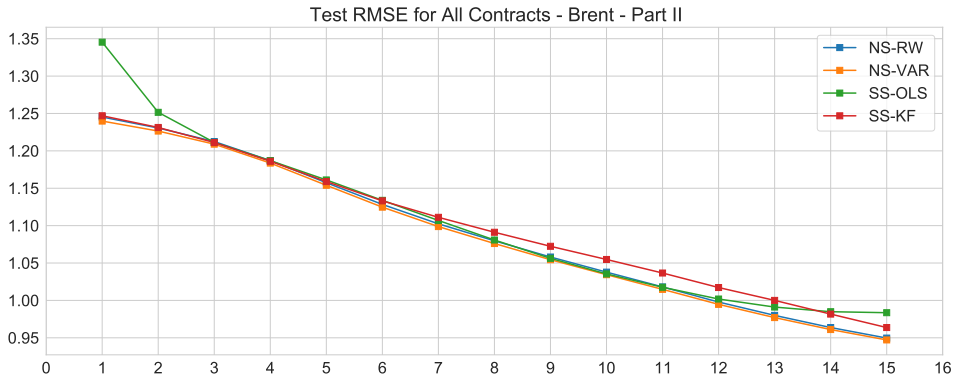

Figure 5.28: Test RMSE per Rolling Contract - Part II - Brent.

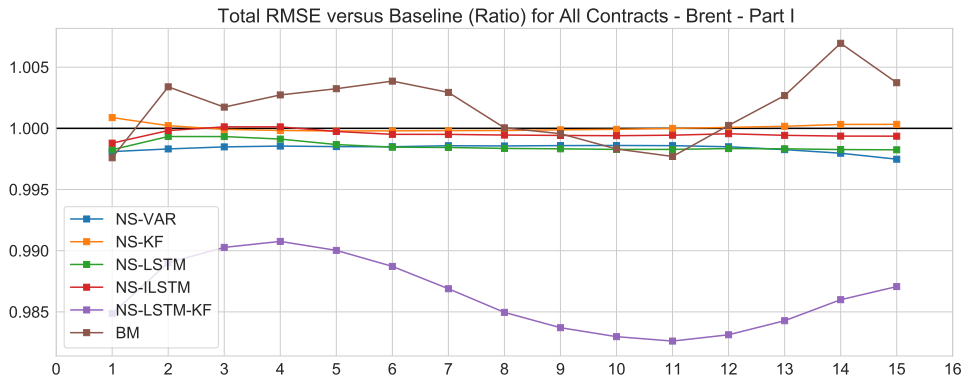

Figure 5.29: Total RMSE per Rolling Contract versus Baseline (NS-RW) - Part I - Brent.

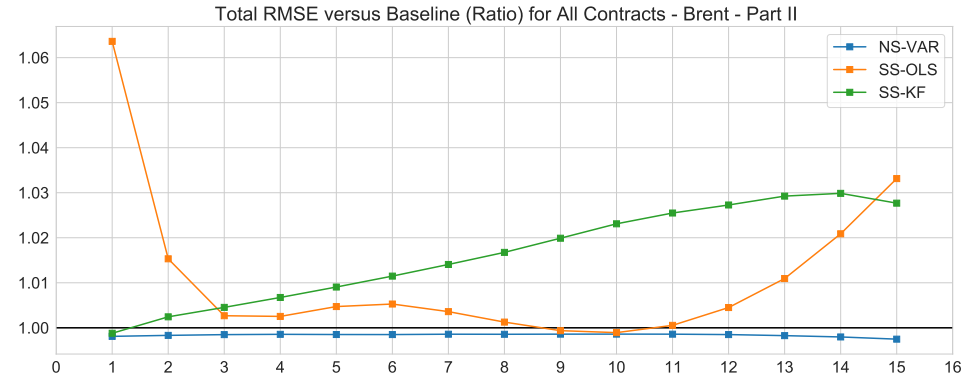

Figure 5.30: Total RMSE per Rolling Contract versus Baseline (NS-RW) - Part II - Brent.

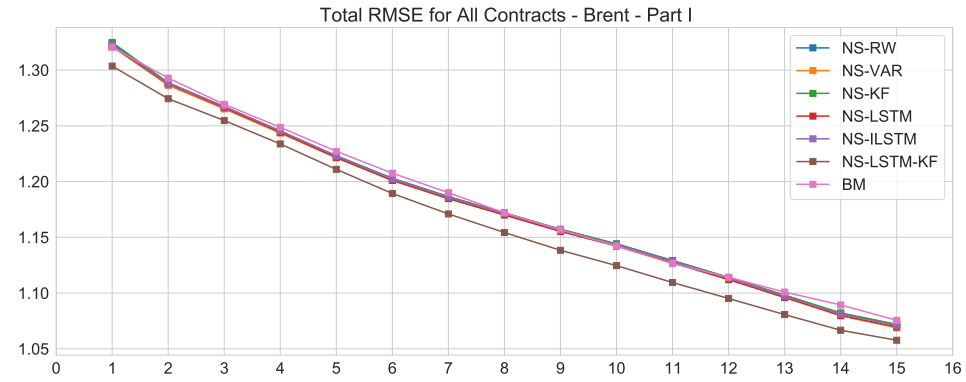

Figure 5.31: Total RMSE per Rolling Contract - Part I - Brent.

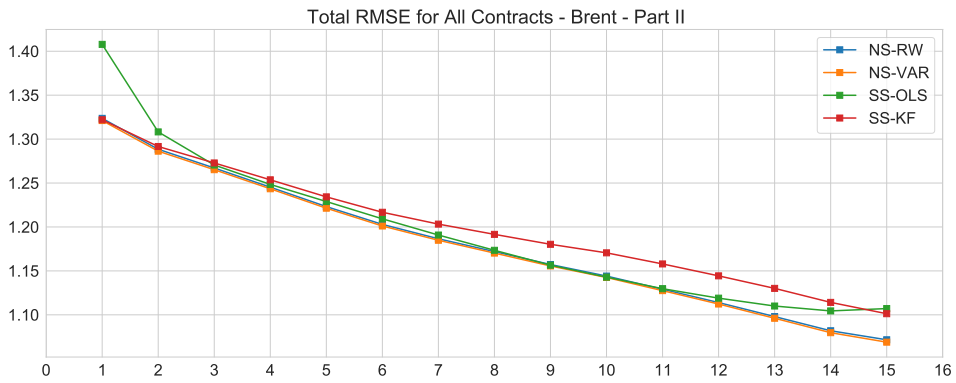

Figure 5.32: Total RMSE per Rolling Contract - Part II - Brent.

We proceed to show the results for WTI.

In Tables 5.5 and 5.6 we summarize all the results for the entire curve for WTI.

Metric	Train RMSE	Validation RMSE	Test RMSE	OOS RMSE	Total RMSE
NS-RW	1.2471	0.8928	1.0413	0.9699	1.1968
NS-KF	1.2468	0.8930	1.0419	0.9703	1.1966
NS-VAR	1.2447	0.8908	1.0377	0.9670	1.1943
NS-LSTM	1.2483	0.8904	1.0385	0.9673	1.1972
NS-ILSTM	1.2483	0.8903	1.0398	0.9679	1.1973
NS-LSTM-KF	1.2495	0.8895	1.0376	0.9664	1.1983
BM	1.2490	0.8907	1.0421	0.9694	1.1981
SS-OLS	1.2808	0.9075	1.0622	0.9879	1.2278
SS-KF	1.2819	0.9100	1.0615	0.9887	1.2289

Table 5.5: Forecast RMSE - WTI

Metric	Train MAE	Validation MAE	Test MAE	OOS MAE	Total MAE
NS-RW	0.8403	0.6875	0.7400	0.7137	0.8150
NS-KF	0.8401	0.6877	0.7402	0.7139	0.8148
NS-VAR	0.8384	0.6870	0.7365	0.7117	0.8131
NS-LSTM	0.8402	0.6844	0.7322	0.7083	0.8137
NS-ILSTM	0.8404	0.6841	0.7338	0.7089	0.8140
NS-LSTM-KF	0.8437	0.6858	0.7436	0.7147	0.8179
BM	0.8403	0.6821	0.7353	0.7087	0.8139
SS-OLS	0.8629	0.6945	0.7506	0.7226	0.8348
SS-KF	0.8641	0.6977	0.7519	0.7248	0.8363

Table 5.6: Forecast MAE - WTI

As it was done with Brent, for better visualization, we show the results for test data, out of sample data and total data in a form of a chart in Figures 5.33 for RMSE and 5.34 for MAE. We also show them in form of a ratio against the NS-RW method, which is the Baseline for all the other methods.

The results are similar overall to the ones obtained with Brent, both for the relative and absolute error values.

Again, it is possible to conclude that there is no much difference between the NS-RW and the other methods also based on the dynamic Nelson-Siegel, although these methods were slightly better overall. The Basic Model also showed similar results to the NS-RW. This supports even further the claim that converting the price into factors before the prediction step is advantageous.

Once more, the methods based on the Schwartz-Smith three-factor model produced predictions much worse than the ones from the dynamic Nelson-Siegel based methods, which was also expected.

Taking into account the RMSE, the NS-LSTM-KF had the best validation, test and out of sample RMSE, followed by the NS-VAR, NS-LSTM and NS-ILSTM. The NS-LSTM-KF, however, had a bad training RMSE. This was exactly the opposite from what occured with Brent. The NS-VAR, NS-LSTM and NS-ILSTM, on the other hand, were more consistent. For the NS-KF, the results were again very similar to the NS-RW, with almost no difference.

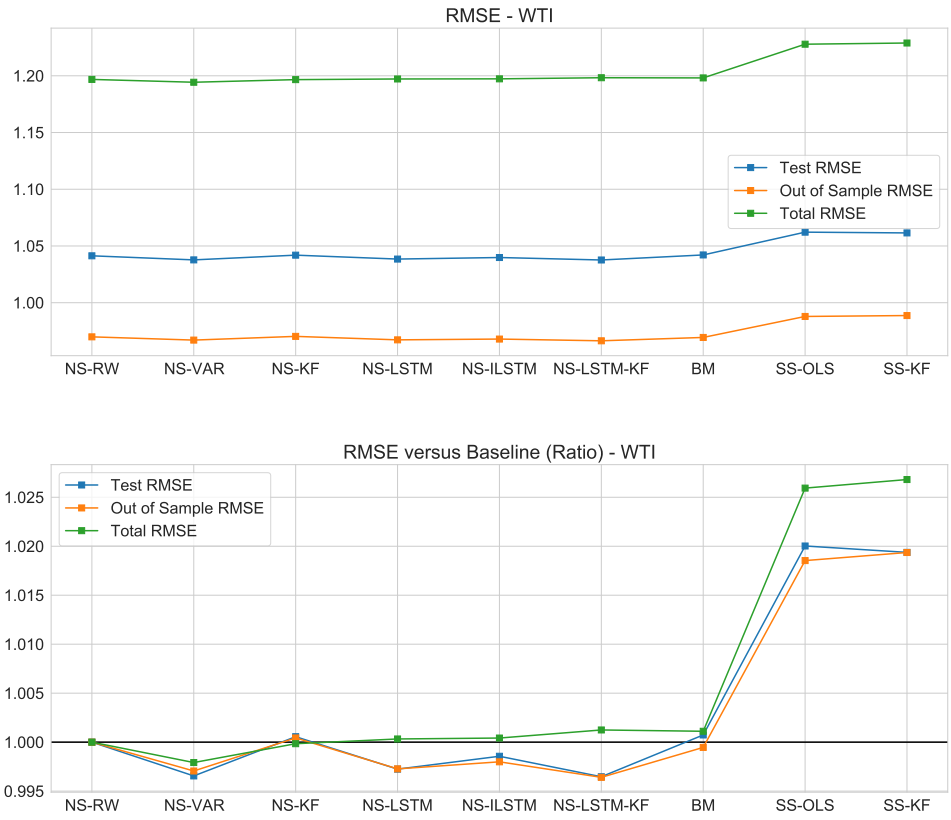

Figure 5.33: (top) RMSE - WTI. (bottom) RMSE versus Baseline (Ratio) - WTI. Test RMSE refers to the RMSE for the test data; Out of Sample RMSE refers to the RMSE for the test data plus validation data, that is, all the data that does not belong to training data; Total RMSE refers to all of the data, including training data and out of sample data.

As for the MAE results, just as what happened with Brent, the NS-LSTM and NS-ILSTM were the best models with a good margin, specially for the test and out of sample data, with more than 1% of improvement for the NS-LSTM in the test data. The BM showed interesting MAE results. Distinctively, the NS-LSTM-KF method has not shown good MAE results, diversely from what occured with the RMSE.

We also present the RMSE errors for each contract. For this, we separate the models/methods in the same two groups used for Brent. These are shown in Figures 5.35, 5.36, 5.37 and 5.38 for the test data and in Figures 5.39, 5.40, 5.41 and 5.42 for the total data.

The results were, in general, very similar to the ones from Brent. The first contracts were again the ones with larger errors. The NS-LSTM, NS-ILSTM and NS-ILSTM-KF produced the best predictions for the first contract. The BM was very inconsistent, with very good predictions for some contracts and very bad predictions for others. The NS-VAR was the one with the most consistent gain versus the NS-RW across the entire curve, just like with Brent. As for the SS-KF and SS-OLS, both presented terrible prediction RMSE for the first two contracts, in comparison to the NS-RW.

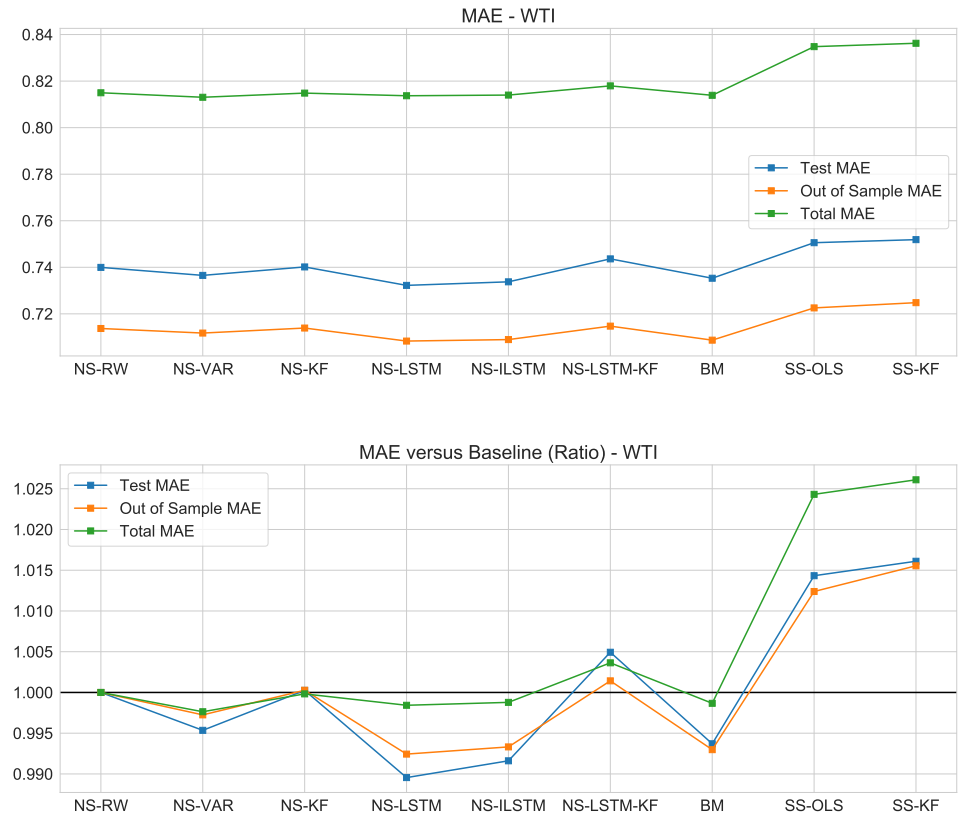

Figure 5.34: (top) MAE - WTI. (bottom) MAE versus Baseline (Ratio) - WTI. Test MAE refers to the MAE for the test data; Out of Sample MAE refers to the MAE for the test data plus validation data, that is, all the data that does not belong to training data; Total MAE refers to all of the data, including training data and out of sample data.

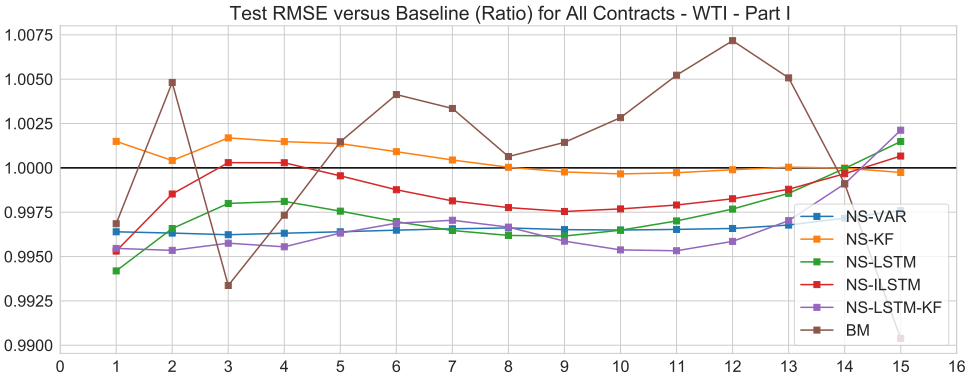

Figure 5.35: Test RMSE per Rolling Contract versus Baseline (NS-RW) - Part I - WTI.

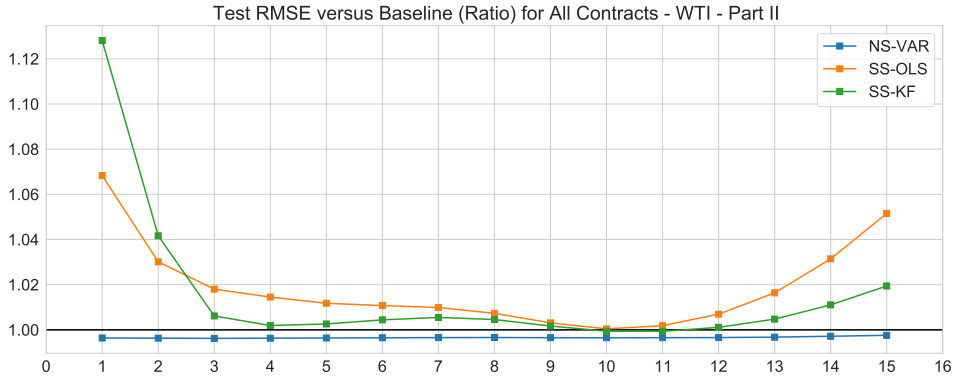

Figure 5.36: Test RMSE per Rolling Contract versus Baseline (NS-RW) - Part II - WTI.

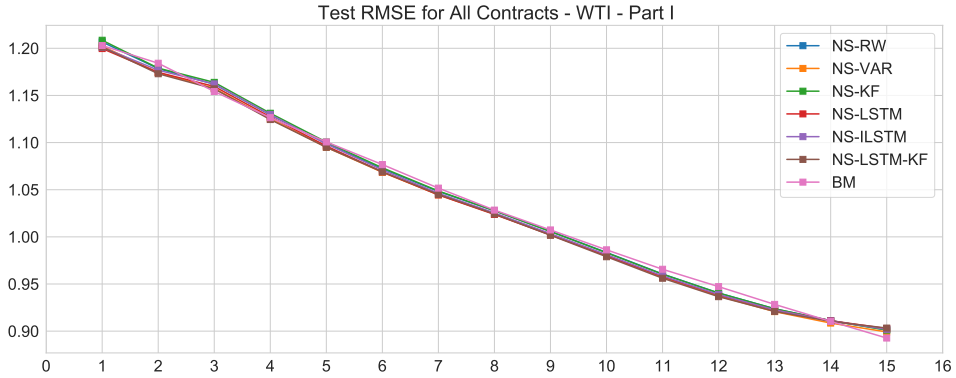

Figure 5.37: Test RMSE per Rolling Contract - Part I - WTI.

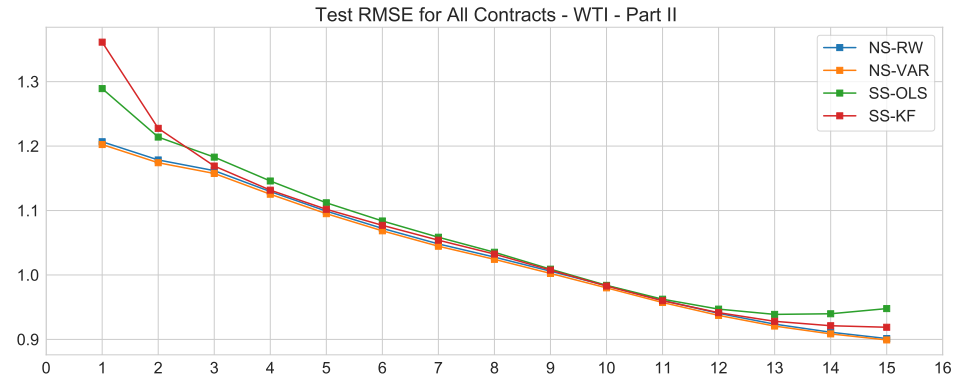

Figure 5.38: Test RMSE per Rolling Contract - Part II - WTI.

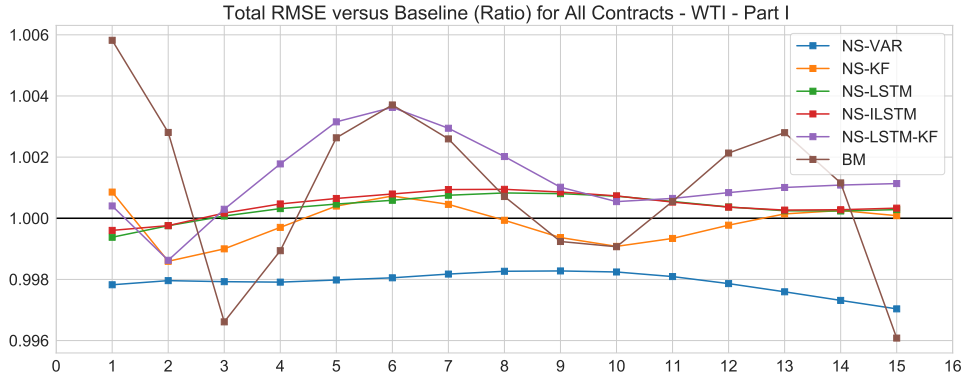

Figure 5.39: Total RMSE per Rolling Contract versus Baseline (NS-RW) - Part I - WTI.

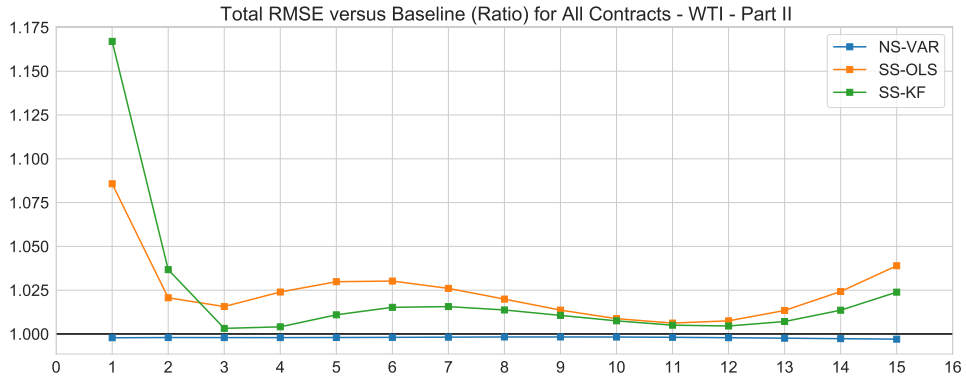

Figure 5.40: Total RMSE per Rolling Contract versus Baseline (NS-RW) - Part II - WTI.

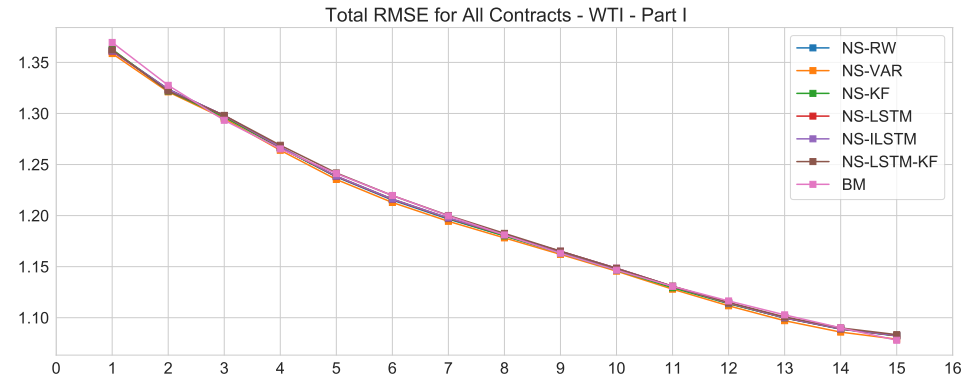

Figure 5.41: Total RMSE per Rolling Contract - Part I - WTI.

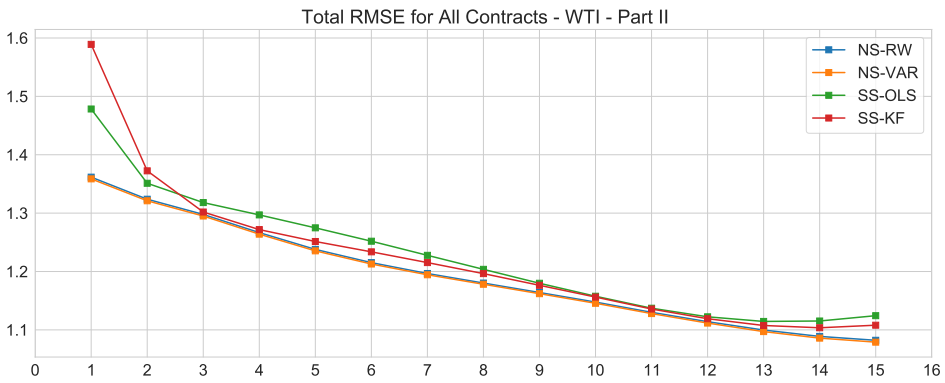

Figure 5.42: Total RMSE per Rolling Contract - Part II - WTI.

5.3.4 Rolling Validation Results

Using the best models in terms of error from last section, we have also run a simple rolling validation. Skipping the first two years, for each year in the data, we used the preceding two years as test data to fit the model, and then applied the model for that year. The treatment given has two main limitations: two years represents a very limited amount of data, so the models may not be properly fitted; for the same reason, we have not separated the data into validation data, hence decreasing our capability of choosing the best models (which is specially a limitation for the models involving LSTM, because we cannot select a good initialization). As we have not used validation data, we set the hyperparameters of the models based on the ones calculated in the last Section.

Nevertheless, although we may not obtain the best results possible with each model, this rolling validation will let us gain more insight on the robustness of these methods.

Below, we show the results for the total RMSE for each year. We include the best models from last section: dynamic Nelson-Siegel with Random Walk, VAR, LSTM, ILSTM and the Basic Model.

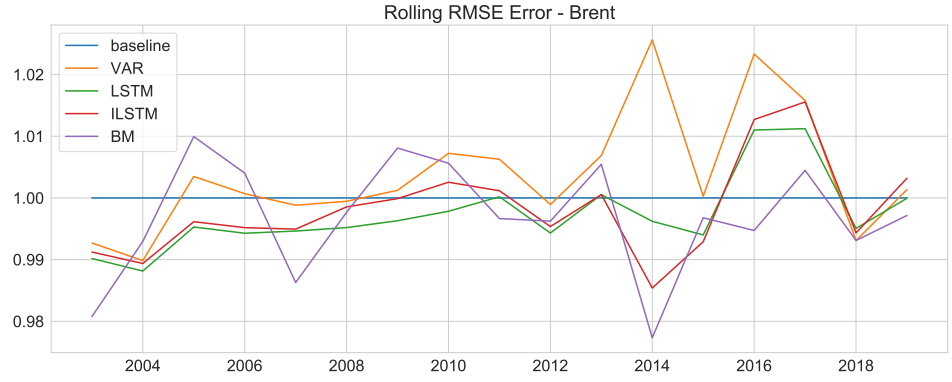

Figure 5.43: Rolling RMSE per year - Brent.

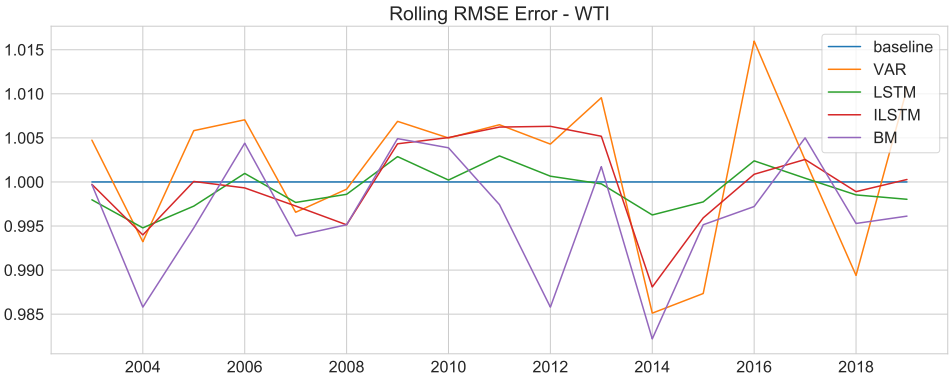

Figure 5.44: Rolling RMSE per year - WTI.

From these results, we can see that only the LSTM, ILSTM and Basic Model produced better results than the Random Walk. Hence, even limited by the amount of data and not

using validation to choose the best models, we still obtained robust results. On the other hand, the VAR was actually worse than the Random Walk, despite not being so limited.

We can conclude that the dynamic Nelson-Siegel with LSTM and ILSTM and the Basic Model both are robust models, and may produce even better results in a more appropriate setting.

Chapter 6

Conclusions

In this work, we have analysed Brent and WTI future prices curves using the dynamic Nelson-Siegel and Schwartz-Smith three-factor models. We compared how well these two models fit the data, and have also explored multiple ways of forecasting the curve price one day ahead.

Brent and WTI future prices have very similar behaviours, so we expected the results to be very similar, which in fact they were, what is a positive thing for the robustness of the models studied.

In regards to the term structure models, we have found out that the dynamic Nelson-Siegel was better at both fitting and forecasting the price curve for the two commodities. Besides, this model was simpler to fit, as it has less parameters, and also offered more flexibility in forecasting, making in possible to test neural networks for instance.

It was made clear, however, that it is difficult to gain much advantage over the Random Walk in forecasting the curve prices, but it definitely exists. Among the methods we have tested, the LSTM and ILSTM were the overall best, when considering RMSE and MAE for Brent and WTI, but also the robustness to parameter variation. These models could provide an improvement in the prediction up to 0.5% to 1.0% against the Random Walk, specially for the first contract. The VAR(1) model has also displayed good results, specially when taking into account the simplicity of the model and the speed of calculation.

These results, in general, fall in line with those found in the literature for estimation of commodities future prices.

Taking into account the results for the rolling window models, the advantage over the Random Walk was even smaller. In this case, the LSTM, ILSTM and Basic Model were the best models. Considering the simplicity of this rolling validation, it was made apparent that these models are indeed robust. On the other hand, the VAR(1) model was actually worse than the Random Walk, which is certainly is concerning for this model robustness. There is a lot of margin to improvement in the way the rolling models are fitted, but that can be the theme for a future work.

We can conclude that the best estimation models were the dynamic Nelson-Siegel with LSTM and ILSTM.

As for the LSTM-KF, it has shown decent results and is certainly promising and may be used in forecasting financial time series. The greater drawback is its greater complexity, universe of parameters and time to run. There is a lot of room for improvement for this method, and similar methods, that is, neural networks in a filter framework.

Furthermore, we have also shown that converting the price curve into factors before the prediction step is advantageous, as it was clear by the Basic Model results. For the rolling window results, however, the Basic Model was among the best models. Thus we can conclude

that is possible to make the prediction directly from the curve prices, without much accuracy loss, even though, in this case, we need to take into account the time to maturity inside the prediction step.

Finally, there are multiple improvements and expansions that can be made from this work, and can inspire future works:

- Try using the Svensson parametrization instead of the Nelson-Siegel;
- Consider λ to be dynamic as well;
- Consider a different distribution for the errors, with heavier tails - this can be specially good for the methods involving filters;
- Consider the volatility to be stochastic;
- Consider H and Q from the LSTM-KF method to also be the output of a LSTM, as was made in [Coskun et al., 2017] and [Krishnan et al., 2015];
- Apply this framework to a larger universe of commodities, including the ones with seasonality patterns;
- Consider a different number of contracts and maturity ranges;
- Try using this framework in other financial contexts, such as volatility surface prediction;
- Expand further on the neural network methods used to predict financial time series, including the LSTM-KF;
- Include external series to the prediction, such as macroeconomic time series.

Appendix A

Appendix

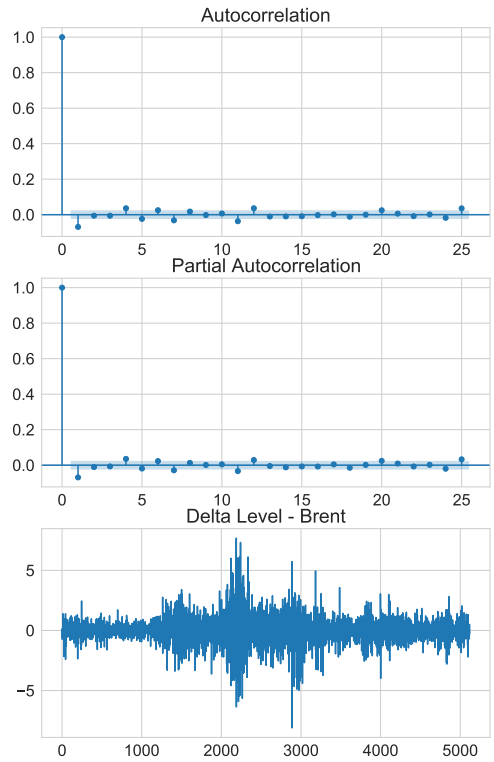

Figure A.1: Delta Level ACF and PACF

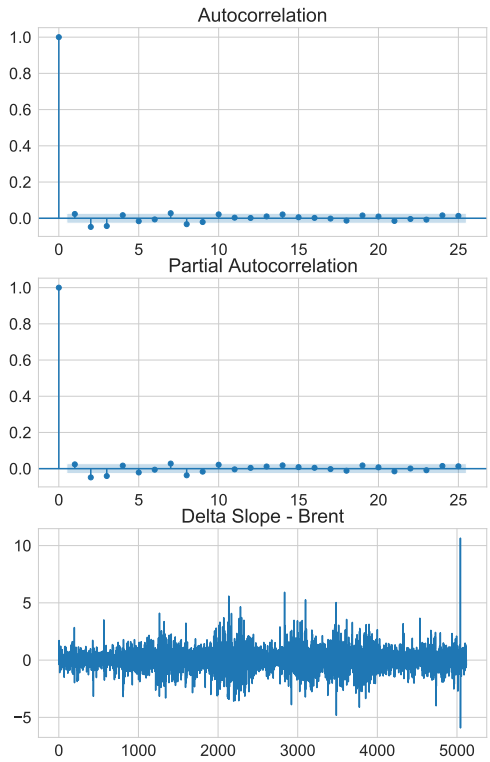

Figure A.2: Delta Slope ACF and PACF

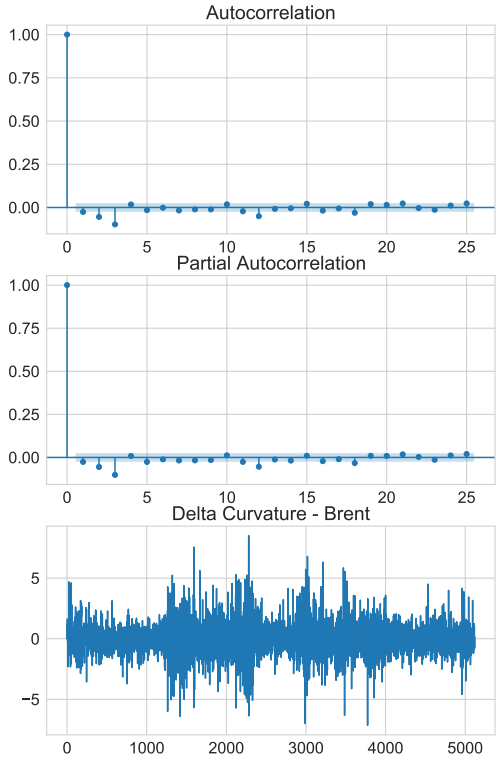

Figure A.3: Delta Curvature ACF and PACF

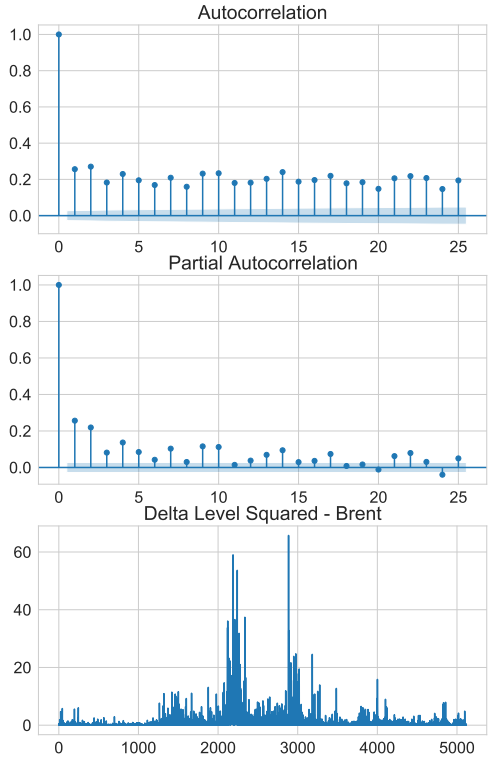

Figure A.4: Delta Level Squared ACF and PACF

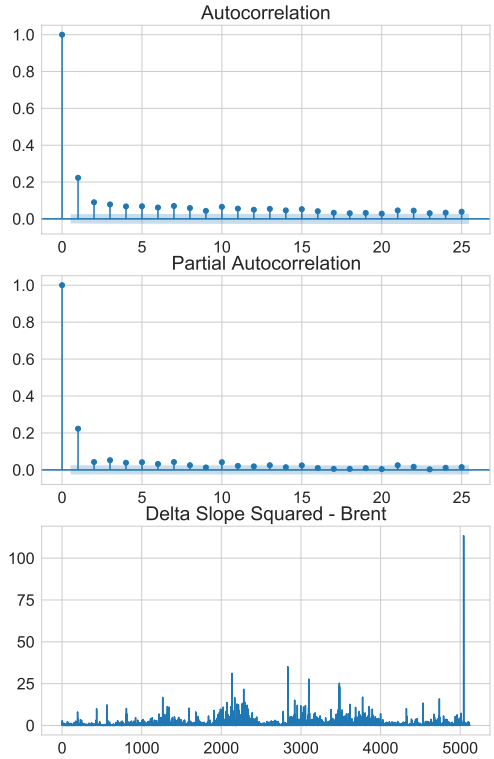

Figure A.5: Delta Slope Squared ACF and PACF

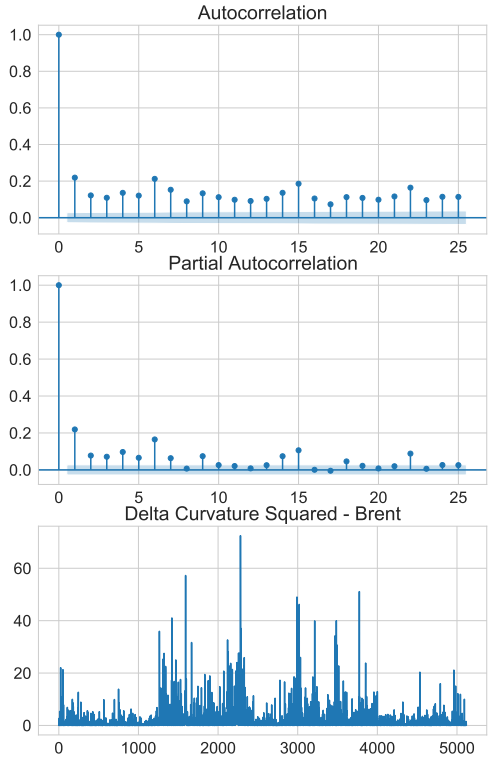

Figure A.6: Delta Curvature Squared ACF and PACF

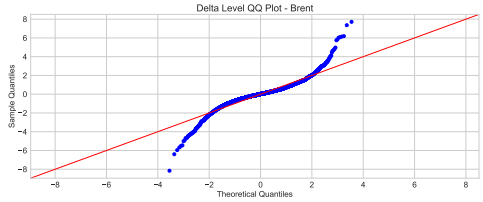

Figure A.7: Delta Level QQ Plot

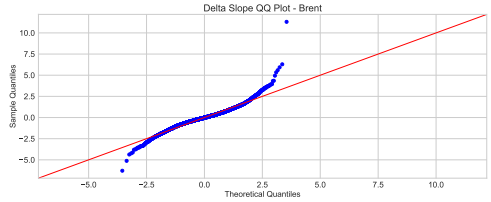

Figure A.8: Delta Slope QQ Plot

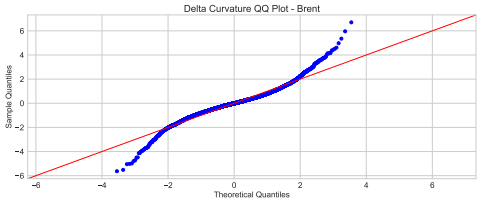

Figure A.9: Delta Curvature QQ Plot

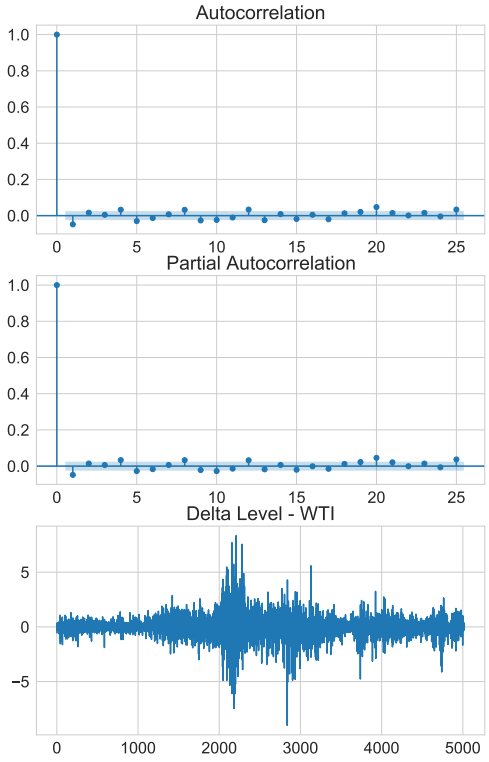

Figure A.10: Delta Level ACF and PACF

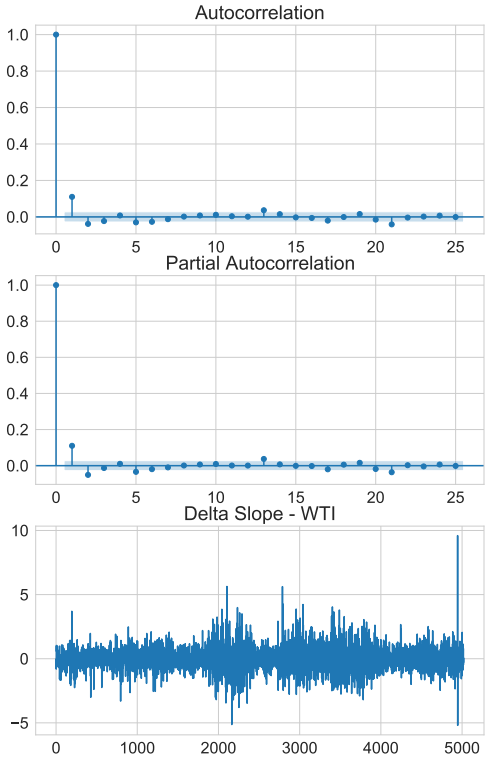

Figure A.11: Delta Slope ACF and PACF

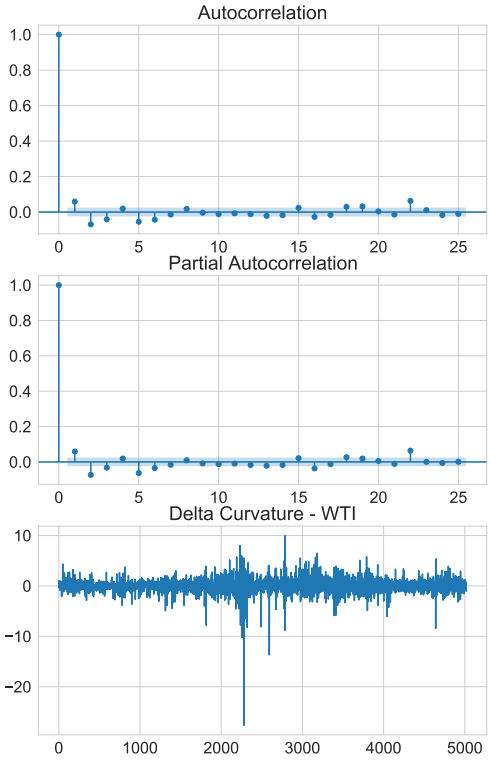

Figure A.12: Delta Curvature ACF and PACF

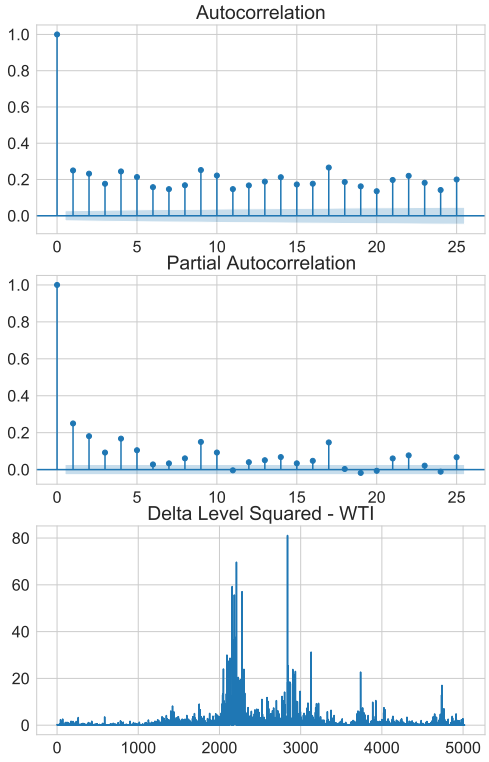

Figure A.13: Delta Level Squared ACF and PACF

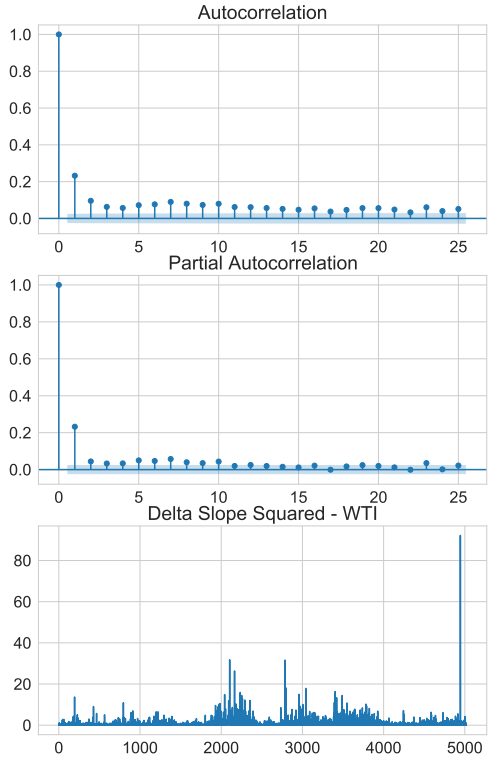

Figure A.14: Delta Slope Squared ACF and PACF

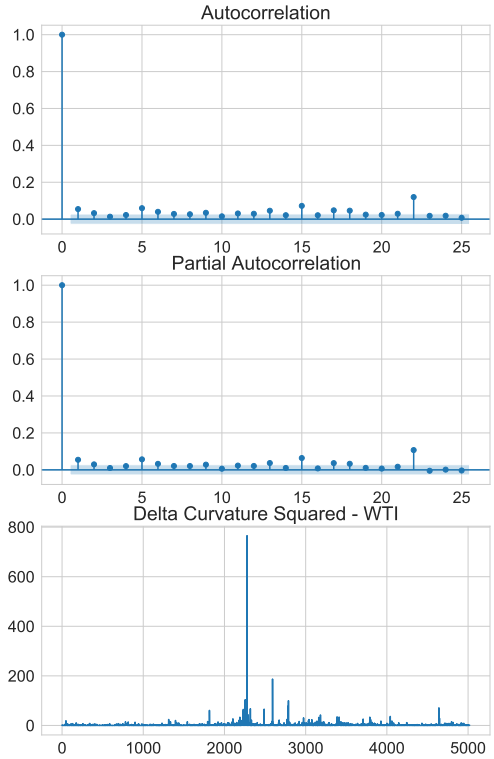

Figure A.15: Delta Curvature Squared ACF and PACF

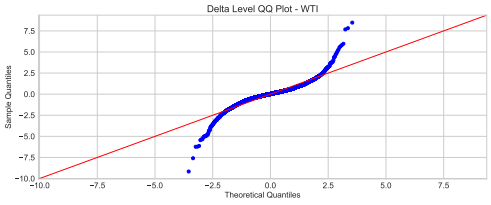

Figure A.16: Delta Level QQ Plot

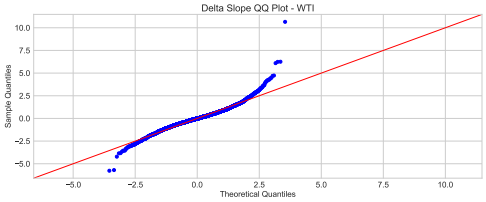

Figure A.17: Delta Slope QQ Plot

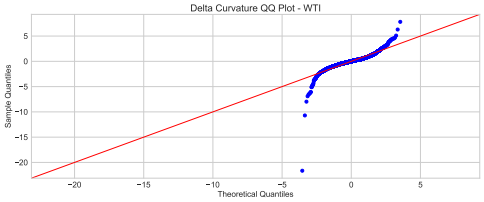

Figure A.18: Delta Curvature QQ Plot

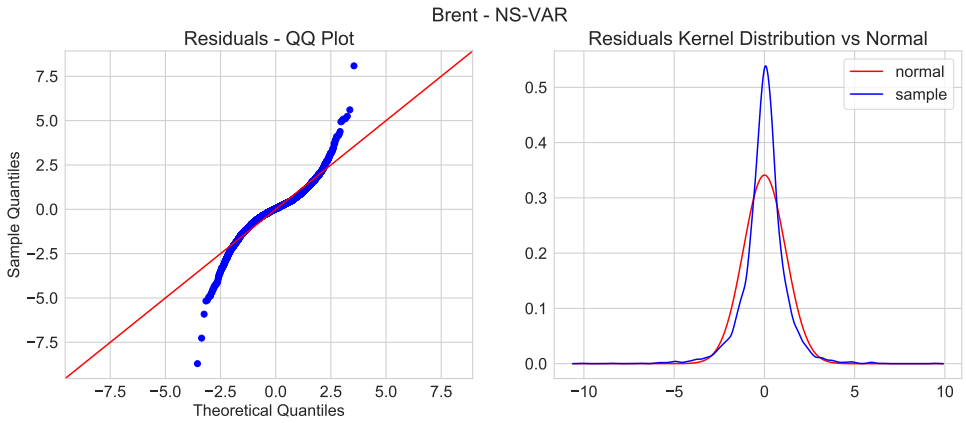

Figure A.19: dynamic Nelson-Siegel: VAR Residuals QQ Plot and Distribution

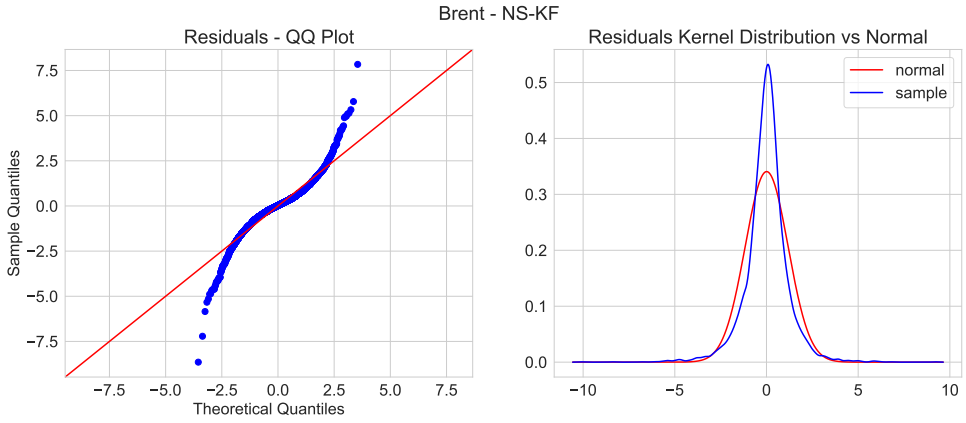

Figure A.20: dynamic Nelson-Siegel: Kalman Filter Residuals QQ Plot and Distribution

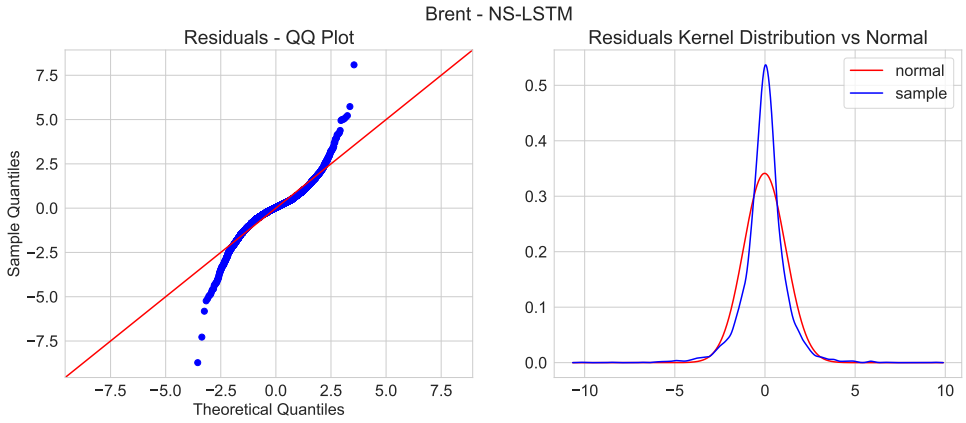

Figure A.21: dynamic Nelson-Siegel: LSTM Residuals QQ Plot and Distribution

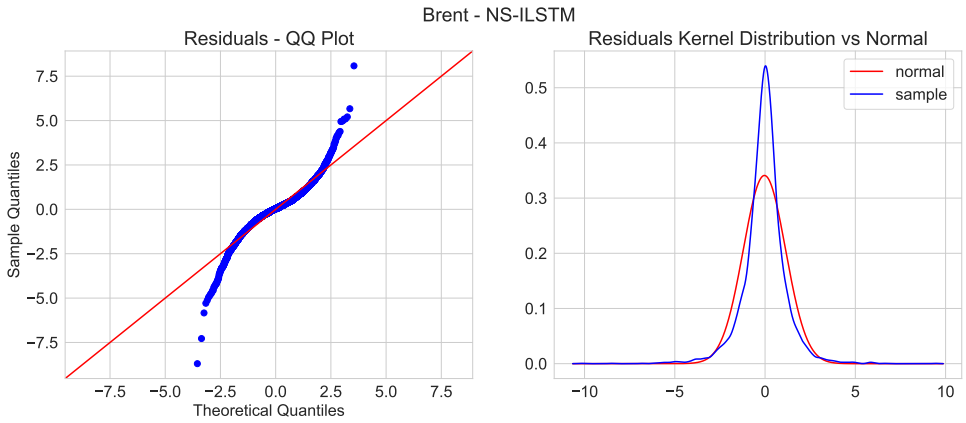

Figure A.22: dynamic Nelson-Siegel: ILSTM Residuals QQ Plot and Distribution

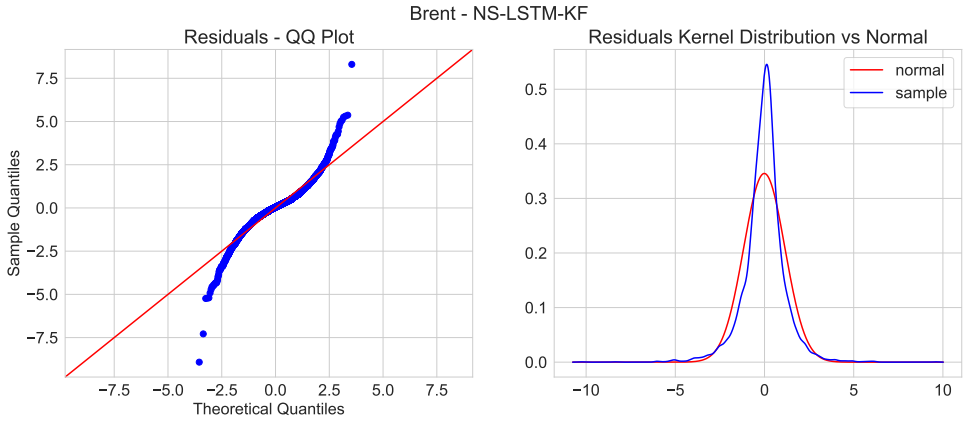

Figure A.23: dynamic Nelson-Siegel: LSTM-KF Residuals QQ Plot and Distribution

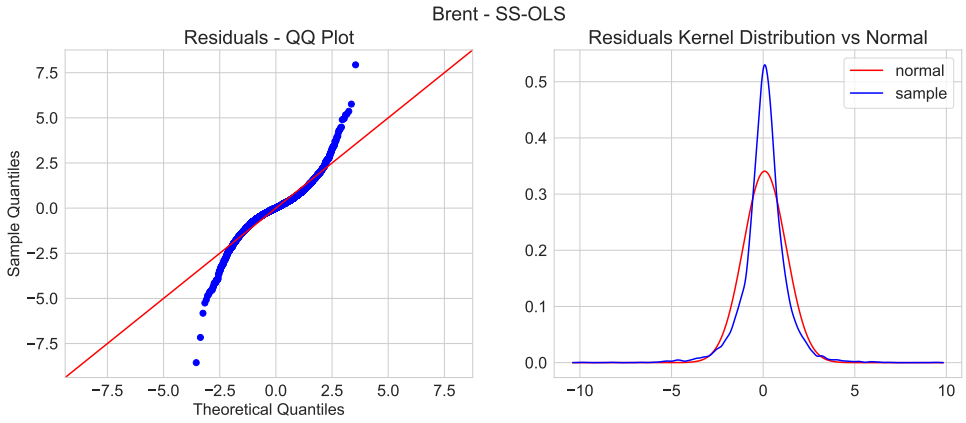

Figure A.24: Schwartz-Smith three-factor model: OLS Residuals QQ Plot and Distribution

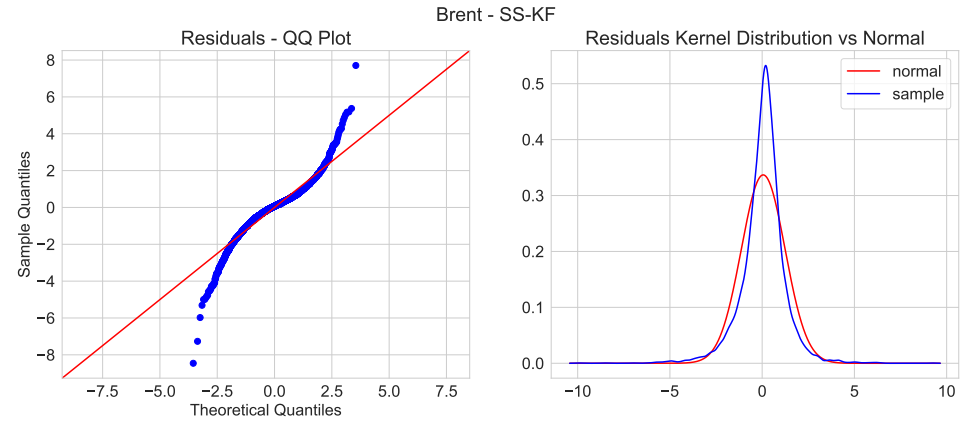

Figure A.25: Schwartz-Smith three-factor model: Kalman Filter Residuals QQ Plot and Distribution

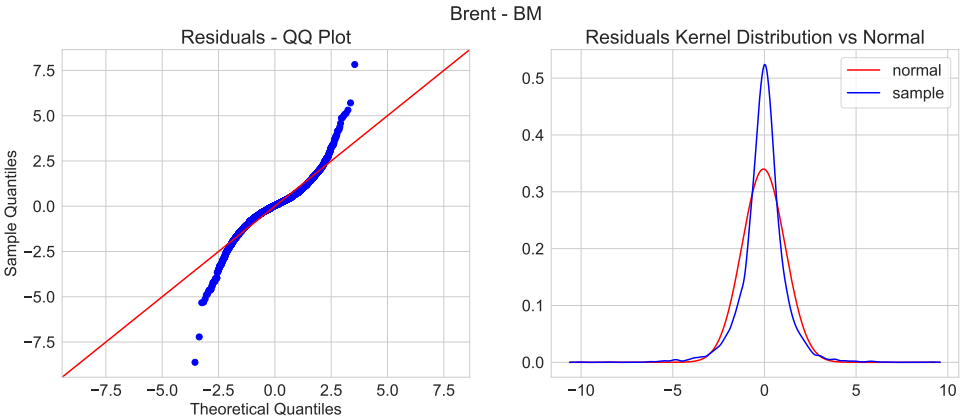

Figure A.26: Basic Model: Residuals QQ Plot and Distribution

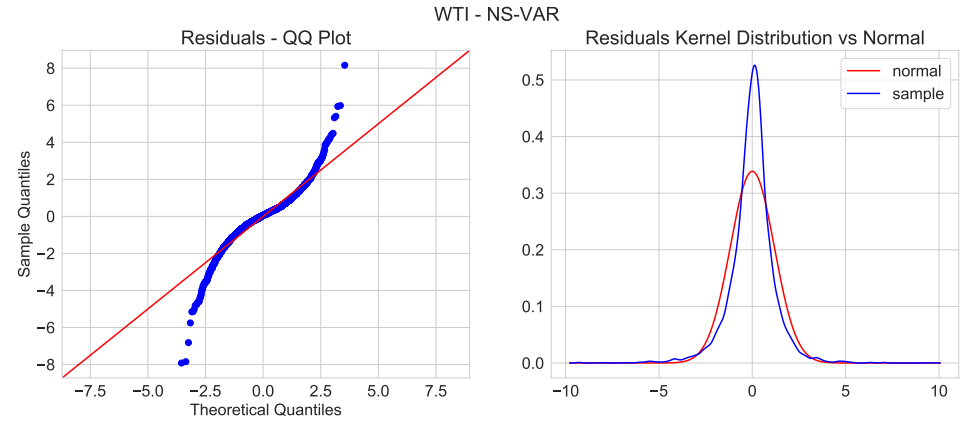

Figure A.27: dynamic Nelson-Siegel: VAR Residuals QQ Plot and Distribution

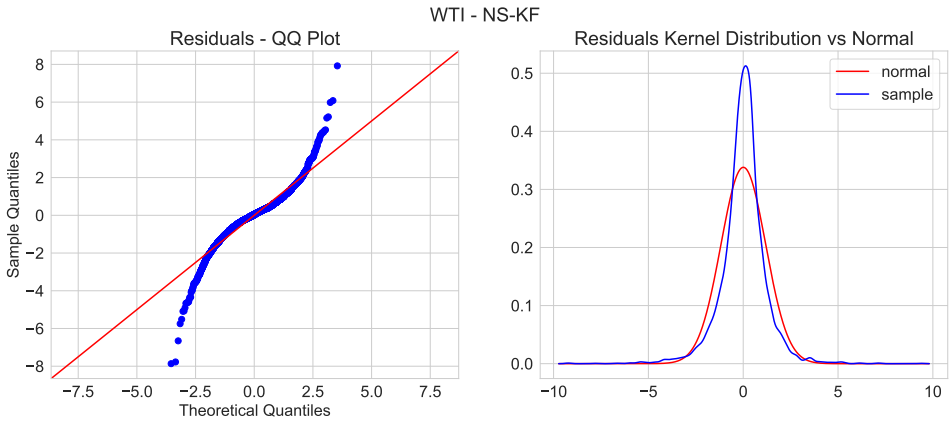

Figure A.28: dynamic Nelson-Siegel: Kalman Filter Residuals QQ Plot and Distribution

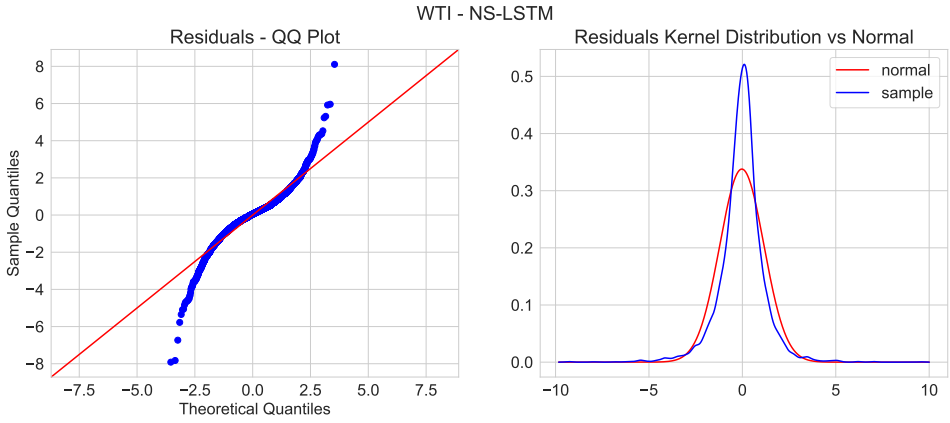

Figure A.29: dynamic Nelson-Siegel: LSTM Residuals QQ Plot and Distribution

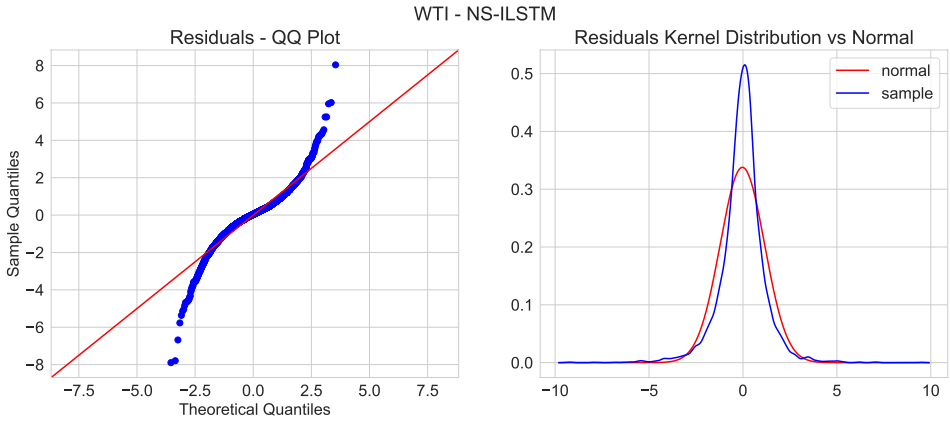

Figure A.30: dynamic Nelson-Siegel: ILSTM Residuals QQ Plot and Distribution

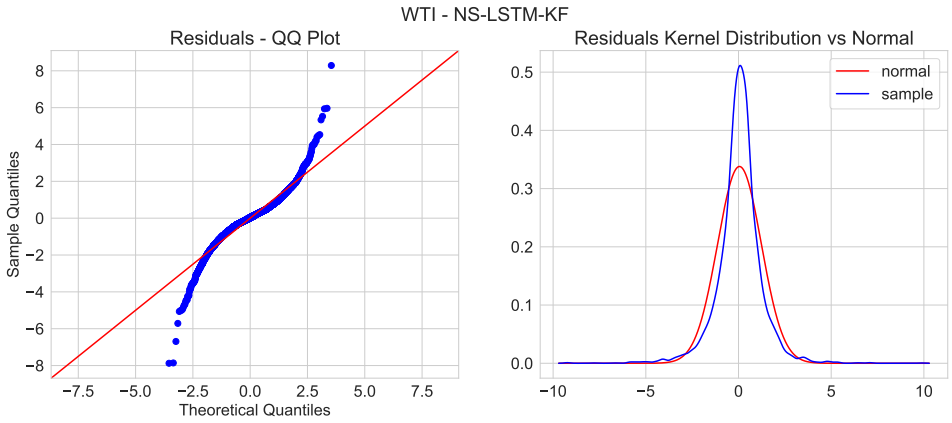

Figure A.31: dynamic Nelson-Siegel: LSTM-KF Residuals QQ Plot and Distribution

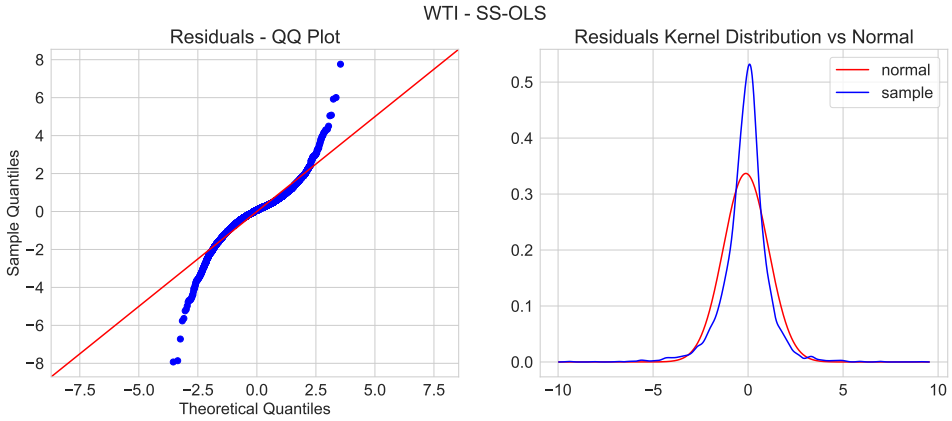

Figure A.32: Schwartz-Smith three-factor model: OLS Residuals QQ Plot and Distribution

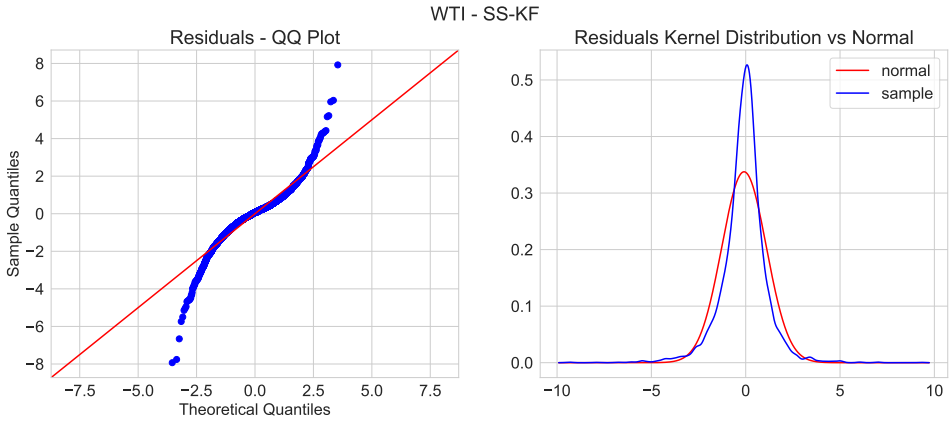

Figure A.33: Schwartz-Smith three-factor model: Kalman Filter Residuals QQ Plot and Distribution

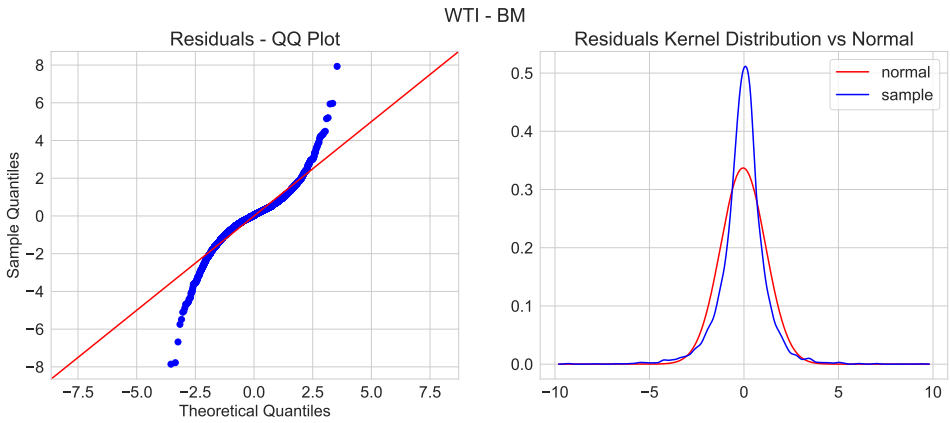

Figure A.34: Basic Model: Residuals QQ Plot and Distribution

Hyperparameter	Value
Number of Lags	1

Table A.1: Hyperparameters values for dynamic Nelson-Siegel with VAR for Brent

Hyperparameter	Value
Number of Lags	1

Table A.2: Hyperparameters values for dynamic Nelson-Siegel with VAR for WTI

Hyperparameter	Value
Number of Time Steps	10
Number of Layers	1
Number of Units	10
Number of Epochs	10
Batch Size	1

Table A.3: Hyperparameters values for dynamic Nelson-Siegel with LSTM for Brent

Hyperparameter	Value
Number of Time Steps	20
Number of Layers	1
Number of Units	10
Number of Epochs	10
Batch Size	1

Table A.4: Hyperparameters values for dynamic Nelson-Siegel with LSTM for WTI

Hyperparameter	Value
Number of Time Steps	15
Number of Layers	1
Number of Units	10
Number of Epochs	10
Batch Size	1

Table A.5: Hyperparameters values for dynamic Nelson-Siegel with ILSTM for Brent

Hyperparameter	Value
Number of Time Steps	15
Number of Layers	1
Number of Units	10
Number of Epochs	10
Batch Size	1

Table A.6: Hyperparameters values for dynamic Nelson-Siegel with ILSTM for WTI

Hyperparameter	Value
Number of Time Steps	10
Number of Layers	1
Number of Units	10
Number of Total Epochs	10
Fit Window	1500
Fit Epochs	5
Batch Size	1

Table A.7: Hyperparameters values for dynamic Nelson-Siegel with LSTM-KF for Brent

Hyperparameter	Value
Number of Time Steps	10
Number of Layers	1
Number of Units	10
Number of Total Epochs	2
Fit Window	1000
Fit Epochs	10
Batch Size	1

Table A.8: Hyperparameters values for dynamic Nelson-Siegel with LSTM-KF for WTI

Hyperparameter	Value
Number of Time Steps	10
Number of Layers	1
Number of Units	10
Number of Epochs	10
Batch Size	1

Table A.9: Hyperparameters values for Basic Model for Brent

Hyperparameter	Value
Number of Time Steps	15
Number of Layers	1
Number of Units	10
Number of Epochs	10
Batch Size	1

Table A.10: Hyperparameters values for Basic Model for WTI

Bibliography

[Aiube, 2013] Aiube, F. A. L. (2013). *Modelos quantitativos em finanças com enfoque em commodities*. Bookman Editora.

[Aiube et al., 2008] Aiube, F. A. L., Baidya, T. K. N., and Tito, E. A. H. (2008). Analysis of commodity prices with the particle filter. *Energy Economics*, 30(2):597–605.

[Aiube and Samanez, 2014] Aiube, F. A. L. and Samanez, C. P. (2014). On the comparison of schwartz and smith's two- and three-factor models on commodity prices. *Applied Economics*, 46(30):3736–3749.

[Baruník and Malinská, 2016] Baruník, J. and Malinská, B. (2016). Forecasting the term structure of crude oil futures prices with neural networks. *Applied Energy*, 164:366–379.

[Bhar and Lee, 2011] Bhar, R. and Lee, D. (2011). Time-varying market price of risk in the crude oil futures market. *Journal of Futures Markets*, 31(8):779–807.

[Brennan and Schwartz, 1985] Brennan, M. J. and Schwartz, E. S. (1985). Evaluating natural resource investments. *Journal of business*, 58(2):135–157.

[Chen et al., 2017] Chen, Y., He, K., and Tso, G. K. (2017). Forecasting crude oil prices: a deep learning based model. *Procedia Computer Science*, 122:300–307. 5th International Conference on Information Technology and Quantitative Management, ITQM 2017.

[Christensen et al., 2011] Christensen, J. H., Diebold, F. X., and Rudebusch, G. D. (2011). The affine arbitrage-free class of nelson–siegel term structure models. *Journal of Econometrics*, 164(1):4–20. Annals Issue on Forecasting.

[Cortazar and Naranjo, 2006] Cortazar, G. and Naranjo, L. (2006). An n-factor gaussian model of oil futures prices. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 26(3):243–268.

[Cortazar and Schwartz, 2003] Cortazar, G. and Schwartz, E. S. (2003). Implementing a stochastic model for oil futures prices. *Energy Economics*, 25(3):215–238.

[Coskun et al., 2017] Coskun, H., Achilles, F., DiPietro, R., Navab, N., and Tombari, F. (2017). Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[Dash et al., 2016] Dash, S. K., Bisoi, R., and Dash, P. (2016). A hybrid functional link dynamic neural network and evolutionary unscented kalman filter for short-term electricity price forecasting. *Neural Computing and Applications*, 27(7):2123–2140.

[Diebold and Li, 2006] Diebold, F. X. and Li, C. (2006). Forecasting the term structure of government bond yields. *Journal of Econometrics*, 130(2):337–364.

[Diebold et al., 2008] Diebold, F. X., Li, C., and Yue, V. Z. (2008). Global yield curve dynamics and interactions: A dynamic nelson–siegel approach. *Journal of Econometrics*, 146(2):351–363. Honoring the research contributions of Charles R. Nelson.

[Ding, 2018] Ding, Y. (2018). A novel decompose-ensemble methodology with aic-ann approach for crude oil forecasting. *Energy*, 154:328–336.

[Durbin and Koopman, 2012] Durbin, J. and Koopman, S. J. (2012). *Time series analysis by state space methods*. Oxford university press.

[Freitas et al., 2000] Freitas, J. F. G. d., Niranjan, M., Gee, A. H., and Doucet, A. (2000). Sequential Monte Carlo Methods to Train Neural Network Models. *Neural Computation*, 12(4):955–993.

[Ghoddusi et al., 2019] Ghoddusi, H., Creamer, G. G., and Rafizadeh, N. (2019). Machine learning in energy economics and finance: A review. *Energy Economics*, 81:709–727.

[Gibson and Schwartz, 1990] Gibson, R. and Schwartz, E. S. (1990). Stochastic convenience yield and the pricing of oil contingent claims. *The Journal of Finance*, 45(3):959–976.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

[Grønborg and Lunde, 2016] Grønborg, N. S. and Lunde, A. (2016). Analyzing oil futures with a dynamic nelson-siegel model. *Journal of Futures Markets*, 36(2):153–173.

[Heaton, 2020] Heaton, J. (2020). Programming LSTM with Keras and TensorFlow (10.2). https://www.youtube.com/watch?v=wY0dyFgNCgY. Last Accessed: 2021-05-08.

[Hikspoors and Jaimungal, 2007] Hikspoors, S. and Jaimungal, S. (2007). Energy spot price models and spread options pricing. *International Journal of Theoretical and Applied Finance*, 10(07):1111–1135.

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

[Jammazi and Aloui, 2012] Jammazi, R. and Aloui, C. (2012). Crude oil price forecasting: Experimental evidence from wavelet decomposition and neural network modeling. *Energy Economics*, 34(3):828–841.

[Julier and Uhlmann, 1997] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In Kadar, I., editor, *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182 – 193. International Society for Optics and Photonics, SPIE.

[Karstanje et al., 2017] Karstanje, D., van der Wel, M., and van Dijk, D. J. (2017). Common factors in commodity futures curves. *Available at SSRN 2558014*.

[Keynia, 2012] Keynia, F. (2012). A new feature selection algorithm and composite neural network for electricity price forecasting. *Engineering Applications of Artificial Intelligence*, 25(8):1687–1697.

[Kleppe et al., 2021] Kleppe, T. S., Liesenfeld, R., Moura, G. V., and Oglend, A. (2021). Analyzing commodity futures using factor state-space models with wishart stochastic volatility. *Econometrics and Statistics*.

[Krishnan et al., 2015] Krishnan, R. G., Shalit, U., and Sontag, D. (2015). Deep kalman filters.

[Lim and Zohren, 2021] Lim, B. and Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194):20200209.

[Lucia and Schwartz, 2002] Lucia, J. J. and Schwartz, E. S. (2002). Electricity prices and power derivatives: Evidence from the nordic power exchange. *Review of derivatives research*, 5(1):5–50.

[Lütkepohl, 2005] Lütkepohl, H. (2005). *New introduction to multiple time series analysis*. Springer Science & Business Media.

[Masini et al., 2021] Masini, R. P., Medeiros, M. C., and Mendes, E. F. (2021). Machine learning advances for time series forecasting.

[Nelson and Siegel, 1987] Nelson, C. R. and Siegel, A. F. (1987). Parsimonious modeling of yield curves. *Journal of business*, 60(4):473–489.

[Olah, 2015] Olah, C. (2015). Understanding LSTM Networks. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Last Accessed: 2021-05-08.

[Panapakidis and Dagoumas, 2016] Panapakidis, I. P. and Dagoumas, A. S. (2016). Day-ahead electricity price forecasting via the application of artificial neural network based models. *Applied Energy*, 172:132–151.

[Peng et al., 2018] Peng, L., Liu, S., Liu, R., and Wang, L. (2018). Effective long short-term memory with differential evolution algorithm for electricity price prediction. *Energy*, 162:1301–1314.

[Phi, 2018] Phi, M. (2018). Illustrated Guide to LSTM's and GRU's: A step by step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21. Last Accessed: 2021-05-08.

[Schwartz and Smith, 2000] Schwartz, E. and Smith, J. E. (2000). Short-term variations and long-term dynamics in commodity prices. *Management Science*, 46(7):893–911.

[Schwartz, 1997] Schwartz, E. S. (1997). The stochastic behavior of commodity prices: Implications for valuation and hedging. *The Journal of finance*, 52(3):923–973.

[Sørensen, 2002] Sørensen, C. (2002). Modeling seasonality in agricultural commodity futures. *Journal of Futures Markets: Futures, Options, and Other Derivative Products*, 22(5):393–426.

[Svensson, 1994] Svensson, L. E. (1994). Estimating and interpreting forward interest rates: Sweden 1992-1994.

[Villaplana, 2003] Villaplana, P. (2003). Pricing power derivatives: A two-factor jump-diffusion approach. *Available at SSRN 493943*.

[Wan and Nelson, 1997] Wan, E. and Nelson, A. (1997). Neural dual extended kalman filtering: applications in speech enhancement and monaural blind signal separation. In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 466–475.

[Wan and Van Der Merwe, 2000] Wan, E. and Van Der Merwe, R. (2000). The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158.

[Wan et al., 1999] Wan, E. A., van der Merwe, R., and Nelson, A. T. (1999). Dual estimation and the unscented transformation. In *NIPS*, pages 666–672.

[West, 2012] West, J. (2012). Long-dated agricultural futures price estimates using the seasonal nelson-siegel model. *International Journal of Business and Management*, 7(3):78–93.

[Zhang and Luh, 2005] Zhang, L. and Luh, P. (2005). Neural network-based market clearing price prediction and confidence interval estimation with an improved extended kalman filter method. *IEEE Transactions on Power Systems*, 20(1):59–66.

[Zhou et al., 2019] Zhou, Y., Li, T., Shi, J., and Qian, Z. (2019). A ceemdan and xgboost-based approach to forecast crude oil prices. *Complexity*, 2019.