MASTER'S THESIS

Interuniversity Master in Statistics and Operations Research UPC-UB

Title: A Decomposition-Ensemble framework for oil price forecasting during Covid19 outbreak: enabling rapid response in IOCs and NOCs

Author: Àngel Ruiz Partida

Advisor: Ernest Pons

Department: Dept d'Econometria, Estadística i Economia Aplicada

University: Universitat de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

UNIVERSITAT DE BARCELONA

Abstract

The Decomposition-Ensemble forecast strategy with Adaptive Tree is a new technique aimed to perform predictions in periods with high volatility and/or structural changes.

In the present study, we apply a Decomposition-Ensemble framework to perform a 5-day ahead WTI oil spot price forecast during the first weeks after the Covid19 outbreak in Feb20 by leveraging the CEEMDAN algorithm to decompose the original series into several IMFs, apply to each an Adaptive Tree, and combine the result to get the final prediction. In all the leveraged techniques, a simple features' engineering process has been applied, i.e., using past demand pattern as the only input, simulating an autoregressive-like behavior as ARIMA.

The results have been compared with several benchmark models, i.e., ARIMA, regular XGBoost, and CEEMDAN-XGBoost, with the goal to study which is the added value of applying a Decomposition algorithm prior to performing the prediction, as well as the potentially better performance of Adaptive Trees over XGBoost in combination with CEEMDAN in a high-volatility environment like the one in 2020 after the Covid19 outbreak.

Results show that ML-based methods outperform ARIMA. In the same direction, decomposing the original series by CEEMDAN before applying XGBoost or Adaptive Tree minimizes RMSE when assessing performance in daily and weekly buckets. Unfortunately, Adaptive Tree combined with CEEMDAN has not shown a clear better performance than CEEMDAN-XGBoost except for the periods with extreme changes in the price signal.

Nevertheless, the study proposes a Roadmap and Business Case to show how an O&G company might implement the proposed Decomposition-Ensemble strategy as a first step to become a digitized company with integrated planning capabilities.

Acknowledgements

The first on the list is one of the most important enablers of the present study, my Thesis' Director, Ernest Pons. Thanks for accepting so rapidly my proposal and changes in the scope. I'm grateful for the willingness to spend time at night reviewing short-notice these pages.

Then I'd like to have a word for my wife Anna and my beloved 3-years old daughter Oona. To Anna, thanks for her patience and understanding when I was spending time developing this report instead of watching TV or Netflix at night. The same for my daughter, for the time "lost" not going to the playground or keep dancing and singing in the living room for hours.

A special word for my dear friend Jose, who without knowing it (or being actively aware), has been one of the most important psychological support during the nights I spent on this study whilst he was doing online streaming on Twitch: those nights I felt less alone thanks to you! The same applies to Tony McGuiness from A&B, his incredible deep house sets accompanied me at nights when working on these pages.

Last but not least, to my parents, who supported me when 7 years ago I decided to study this Master. Thanks to the investment they made at that point in time, I did one big step ahead in my professional career and got the fantastic job I'm still proud of today.

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, we will review which are the motivations behind this study as well as the specific triggers that drove the author to work on this topic. We will go through the main objectives of this study and the scope. Finally, we will review the evolution of the study from the initial idea we had, all the way to the final structure and content of the present report.

1.1 Motivation

The year 2020 and -so far- 2021, will be remembered as the beginning of the 2nd decade of the XXI century; but also because of the historical Australian bushfires; and Kobe Bryant's death; and as the year when for the first time, a non-English-language movie won the Oscars, *Parasite*; and the Black Live Matter protests; and the assault on the US Capitol by Trump's supporters; and Biden becoming the 46th US President; etc.

But, undoubtedly, this was the year of the SARS-CoV-2 pandemic, also known as Covid19.

Since the end of WWII, the world has never experienced such a big and global disruption. At this moment, we are not able to estimate the impact it had and will have, not only in terms of lives and physical and/or psychological aftereffects but also in the economy for each and every country and company in the globe.

One of the biggest impacts of the pandemic was on the oil markets, *the year when, for the first time in history, oil prices turned negative.* Just before the pandemic explosion, back to Jan 2020, the global oil demand-supply was balanced around 100 Mbbl/day. Nevertheless, with the start of the application of severe lockdowns in most of the countries, the economy entered a standby mode, plunging the demand for petroleum products. With the supply not able to react as fast as required, the markets entered a bearish and high-volatility period. Moreover, in March, a price war between Saudi Arabia and Russia increased the negative trend in oil prices and added more uncertainty to the markets. Despite several OPEC cuts in supply during Spring 2020, it was not

enough to avoid the historical crash in oil prices in April 2020 nor to avoid the uncertainty that was present for the rest of the year.

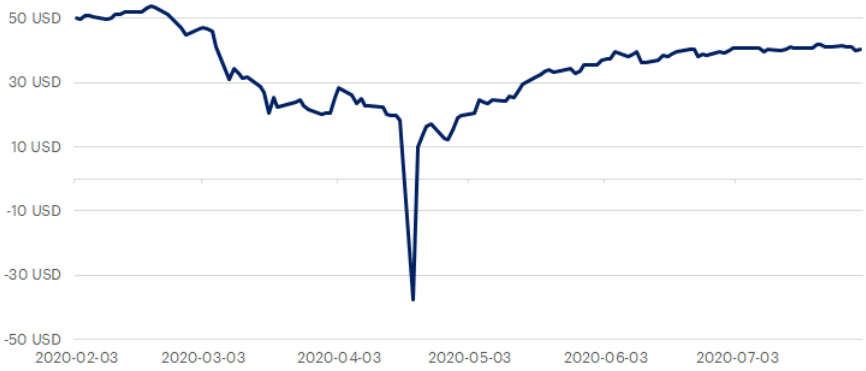

Figure 1.1: WTI NYMEX daily price in the period Feb-Jul 2020. During Apr 20th, WTI oil futures turned negative for the first time in history.

Since then, despite several attempts from the OPEC+ members and Oil Majors to cut supply, oil prices remained low and with high volatility for the rest of the year.

As IOCs and NOCs slowly start to onboard into the Digitalization journey, reacting and adapting faster to changes in their ecosystems and being able to accurately predict oil price, is a must, offering several competitive advantages:

- If a breakeven point is reached, upstream operations can be slowed down to avoid negative operational costs and flood the market with cheaper oil.
- Exploration activities can be activated or postponed.
- Refineries or Downstream companies may decide to buy large quantities of cheap crude oil to increase their margins.
- Tanker companies may adjust their transport capacity.
- Oil companies may perform large Trading operations or open/negotiate contracts.

In order to drive each of the beforementioned points, an accurate oil price forecast is required.

After several years working as a Strategy Consultant for Oil&Gas companies, I realized that they historically paid little attention to those topics as they were having big revenues and benefits. These times are over. Covid19 set the first stone to (probably), the new Energy market, where Renewables and Hydrogen will increase their share.

1.2 Objectives

As mentioned in Section 1.1, the Covid19 disruption set the first stone for a new Energy market with new players taking more and more market share from the Oil&Gas companies. For them, being able to accurately predict oil prices will be a competitive advantage, allowing them to rapidly react to market and/or environmental changes.

The Covid19 pandemic has been one of the first rapid-response stress tests for IOCs and NOCs, and they were not ready. When talking about planning and forecasting, these companies often rely on classical or basic statistical models in order to predict oil prices hindering their ability to quickly adapt to new requirements.

The objective of this paper is to study the WTI spot oil price daily forecast performance for $h+1, \cdots, h+5$ by leveraging a Decomposition-Ensemble strategy based on CEEMDAN-Adaptive Trees during the Covid19 outbreak, i.e. from the first hit in Feb 2020 till the mid of that year. In order to compare the added value of this methodology, results will be compared with ARIMA, simple XGBoost, and CEEMDAN-XGBoost. In order to make a fair comparison between ARIMA and the ML-based strategies, past daily values have been used as the only features for the algorithms, trying to mimic as much as possible ARIMA's behavior and assess the added value of these methodologies over classical approaches.

There are several reasons behind the usage of the Decomposition-Ensemble strategy:

- Commodities' prices tend to be complex, non-linear, and non-stationary time series. By applying a decomposition algorithm to the raw data, we get several simpler components to be forecasted, showing stationary and linear patterns.
- Gradient Boosting-based methods are iterative, i.e., in each step, a tree is trained on the pseudo-residuals made by the average of all previous trees. With this, difficult-to-predict observations causing large prediction errors, get more "attention" from the model. As described in the next chapter, Adaptive Trees allow the user to give more weight to recent observations. Because of that, this methodology is more suitable to deal with time series with structural changes compared to regular GBM like XGBoost.

In the end, by leveraging the decomposition-ensemble methodology, we should be able to deal with non-linearity and non-stationarity as well as periods with structural changes.

As a summary, by the end of this study we'd like to give an answer to the following questions:

- Are ML-based methods outperforming ARIMA when calculating daily forecasts for WTI spot price during periods with structural changes?

- Which is the added value of applying a decomposition algorithm to regular XGBoost in terms of forecast accuracy?
- Can we prove that CEEMNDAN-Adaptive Trees is a better forecasting strategy than the others in terms of accuracy during periods with structural changes?
- How can an Oil&Gas company implement the winning forecast strategy?

1.3 Scope

The scope of the study changed slightly from the beginning as part of the learning process. The initial idea was to use data in weekly buckets and perform 1-step ahead predictions. Using weekly data was opening the possibility of using other oil-related features as well as macroeconomic data to enrich training and prediction for XGBoost and Adaptive Trees. The main shortfalls identified were:

- From a business perspective, if the forecast is meant to be used for operational and trading purposes, it makes more sense to provide a daily than a weekly forecast. For trading purposes, it even makes more sense to get down to hourly frequency. Because of the intra-day variability of oil prices, studying it at a weekly level we are at the risk of losing too much information as prices are not stable within a week. The usage of weekly buckets was firstly picked because of computation limitation: weekly frequency means fewer data points and more stable signal hence, a reasonable amount of data and computation to be tackled by a regular personal computer.
- Machine Learning algorithms can stand out from benchmarking models when dealing with complex and large datasets. By using weekly buckets, the added value of the Decomposition-Ensemble framework either with XGBoost or Adaptive Trees is small compared to ARIMA.

After getting not-as-good-as-expected results for weekly frequency, given the 2 points below and having access to a powerful AWS EC2 instance, drove the decision to switch from 1-step ahead weekly forecast to 5-step ahead daily forecast, for the period week 09.2020 - 32.2020. Because of switching to daily buckets, other oil-related data couldn't be used and training phases for XGBoost/Adaptive Trees were purely based on the time series itself.

2

Methodology

2.1 ARIMA

ARIMA is an acronym that stands for Auto-Regressive Integrated Moving Average, a forecasting technique that was originally developed by Nortbert Wiener et al., in the '30s-'40s, and later updated to be used in business and economic data by George Box and Gwilym Jenkins, which developed a systematic method called the Box-Jenkins methodology to identify, fit, and check ARIMA models to time series data.

ARIMA models are composed of 2 different parts, an Auto-Regressive and a Moving Average that can be defined as:

$$X_t = \phi_1 X_{t-1} + \cdots + \phi_p X_{t-p} + c + a_t - \theta_1 a_{t-1} - \cdots - \theta_q a_{t-q} \tag{2.1}$$

where ϕ are the auto-regressive parameters, θ the moving average parameters, X is the original time series, and a is the series of unknown random errors (assumed to follow a normal distribution). By using the so-called backshift operator B defined as

$$BX_t = X_{t-1} \tag{2.2}$$

the expression in Eq 2.1 can be rewritten as

$$(1 - \phi_1 B - \cdots - \phi_p B^p) X_t = c + (1 - \theta_1 B - \cdots - \theta_q B^q) a_t \tag{2.3}$$

Eq 2.3 is known as an ARMA model of order p and q, i.e., ARMA(p,q). Most economical time series are usually not showing stationary behavior. As the Box-Jenkins method is suited for stationary processes only, differentiation should be applied to the original time series

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(1 - B)^d X_t = c + (1 - \theta_1 B - \cdots - \theta_q B^q) a_t \tag{2.4}$$

Eq 2.4 is now defining and ARIMA(p,d,q), where d is represneting the number of differentiations made into the original time series to make it stationary.

As pointed out before, Box and Jenkins provided an iterative process to fit, estimate, and check ARIMA models:

1. **Identification:** By using autocorrelation and partial autocorrelation plots the user can identify p, d, and q.
2. **Estimation:** ϕ_d and θ_q are estimated normally bu maximum likelihood.
3. **Checking:** The fitted model is checked for the remaining structure in the residuals by analyzing the residuals' autocorrelation and partial autocorrelation functions. If still large correlation values, adjusted values of p and/or q are proposed and the model re-estimated.

This is a quite manual and time-consuming process, that's why there are several automatic algorithms replacing the original Box-Jenkins method. In this study , we relied on *auto.arima()* function from the **forecast** R package developed by Hydnman and Khandakar. As described in their paper [7], the selection of p, q, P, and Q is made by minimizing AIC

$$\text{AIC} = -2\log(L) + 2(p + q + P + Q + k) \tag{2.5}$$

where $k = 1$ if $c \neq 0$ and 0 otherwise.

Regarding the estimation of d and D, for non-seasonal data d is selected based on successive KPSS unit-root tests [8]. For seasonal data, the extended Canova-Hansen test is used to select D [2], and successive KPSS tests over the seasonal-differentiated time series are run to identify d.

2.2 XGBoost

XGBoost (Extreme Gradient Boosting) was developed by Tianqi Chen and Carlos Guestrin in 2016 as a more efficient and scalable implementation of the original Gradient Boosting Trees firstly proposed by J.H. Friedman in 2001.

In general, Gradient Boosting Trees predict the output by leveraging K additive functions. For a given data set with n rows and m columns or features defined as $D = \{(\mathbf{x}_i, y_i)\}$ being $\mathbf{x}_i$ the features ($\mathbf{x}_i \in \mathbb{R}^m$) and y_i the target to be predicted ($y_i \in \mathbb{R}$), the estimated target is obtained as,

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{t} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F} \tag{2.6}$$

where $\mathcal{F}$ represents the space of regression trees, also known as weak learners. The general principle is to use weak learners leveraging an additive stretgy: fix what we have learned and add a new tree in each iteration. Being $\hat{y}_i(t)$ the prediction at time t,

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = y_i^{(t-1)} + f_t(x_i)
\end{aligned} \tag{2.7}$$

In order to train each model $f_k(x_i)$, the following regularized objective function is minimized,

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{2.8}$$
$$\text{where} \quad \Omega(f) = \gamma + \frac{1}{2}\lambda\|w\|^2$$

The loss function l measures the difference between the prediction $\hat{y}_i$ and the target y_i. The second term Ω is penalizing the complexity of the model, acting as a regularization component to avoid overfitting.

As the model in Eq.2.8 has functions as parameters, it can not be optimized using regular techniques, hence, the model is trained in an additive manner. A detailed explanation of how the optimization is performed can be found in Chen et al.[3].

XGBoost's authors proposed 2 more functionalities to avoid overfitting: shrinkage and subsampling.

Shrinkage scales newly added weights by a factor η known as learning rate to reduce the influence of each individual tree and leave space for future trees to improve the prediction task [4]. As η gets smaller, less weight is given to each boosted tree, decreasing the chance of incurring overfitting but increasing the number of iterations needed to achieve good prediction results ($\eta \in [0, 1]$).

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{t} \eta f_k(\mathbf{x}_i) \tag{2.9}$$

By subsampling rows and columns of the original dataset, we add a stochastic element in each boosted tree, decreasing also the likelihood of overfitting. This technique was already applied successfully in the first implementations of the Random Forest algorithm [1, 5]. Given the nature of the data used in this study (a time series), row subsampling it's not applied, as we'd break the correlation structure of the raw data.

2.3 Adaptive Trees

Adaptive Trees introduce a new paradigm called adaptive boosting to deal with structural changes, a phenomenon known as concept drift in Machine Learning literature. This issue arises when the distribution of the target variable Y, the features used for prediction X, and the joint distribution, change over time. Some studies propose to use a small training window from the near-term past as this will collate more information about the present structural change. Nevertheless, by following this approach we are at risk of generating a short-sighted prediction, not being able to replicate patterns from the mid/long term past that may re-occur in the future.

The Adaptive Trees methodology developed by Woloszki in [10] is an extension of Gradient Boosting Trees and it is based on the XGBoost algorithm

as the estimator of the model and predictions. This methodology is built on the fact that GBT gives more importance to hard-to-predict observations as described in 2.7. Based on that, we get an adaptive behavior by the usage of increasing ex-ante observation weights during the Gradient Boosted Trees' training phase. By doing this, Adaptive Trees adjust to structural change, making the impacted observations harder to predict compared to the remote past, hence, the algorithm will put more "effort" into predicting them accurately.

Let $w(t)$ be the weight applied to the obvservation at time t with $t \in [1, T]$,

$$w(t) = e^{-\theta(1-\frac{t}{T})} \tag{2.10}$$

The θ parameter defines the steepness of the weights curve and it normally has a value around 15 [10]. When structural change arises, the more recent observations, which are more heavily weighted, will get increasing weights along with the boosting iterations as they will be inaccurately predicted. As a result, Adaptive Trees will give more importance to the short term whilst capturing patterns from the mid-long term past.

2.4 CEEMDAN decomposition algorithm

The main goal of this algorithm is to decompose a non-linear, non-stationary time-series into several Intrinsic Mode Functions (IMFs) and a residual.

$$x(t) = \sum_{n} IMF_n(t) + r(t) \tag{2.11}$$

IMFs should obey two properties [17]:

1. An IMF has only one extremum between two sub-sequent zero crossings, i.e., the difference between the number of local minima and maxima is at most one.
2. An IMF has a mean value of zero.

In order to understand how the *Complete Ensemble Empirical Mode Decomposition with Additive Noise* (CEEMDAN) works, one must first explore its predecessors: the EMD and the EEMD algorithms.

The EMD (Empirical Mode Decomposition) algorithm was first defined by Huang et al. (1998) with the aim of decomposing a sequence, i.e. a time-series, into several IMFs [6].

Step 0: Set $n = 1$, $r_0(t) = x(t)$; being $x(t)$ the original time series.

Step 1: Extract the n-th IMF as follows:

(a) Set $h_0(t) = r_{n-1}(t)$ and $k = 1$.

(b) Identify all local maxima and minima of $h_{k-1}(t)$.

(c) Construct the upper ($U_{k-1}(t)$) and lower ($L_{k-1}(t)$) envelopes linking all maxima and minima by cubic splines.

(d) Calculate the mean envelope $m_{k-1}(t) = \frac{1}{2}(U_{k-1}(t) - L_{k-1}(t))$.

(e) Build the k-th component $h_k(t) = h_{k-1}(t) - m_{k-1}(t)$. If $h_k(t)$ satisfies the IMF criteria, then set $IMF_n(t) = h_k(t)$, $r_n(t) = r_{n-1}(t) - IMF_n(t)$.

Step 2: If early stop is met, stop the process, if not $r_n(t)$ is the input signal for $n = n + 1$, going back to Step 1.

Being EMD a heuristic method, it suffers from some shortcomings, being the most critical the appearance of the mode mixing effect. This effect can be defined as the situation that similar pieces of oscillations exist at the same corresponding position in different IMFs [17].

To overcome this issue, Wu and Wang extended the original EMD to a new algorithm, EEMD (Ensemble Empirical Mode Decomposition), which adds white noise to the original time series and performs EMD several times [15].

$$x_i(t) = x(t) + w_i(t) \tag{2.12}$$

where $x(t)$ is the original time series, and $w^i(t)$ the i-th white noise with $i = 1, 2, \cdots, N$, being N the number of times one runs the EMD algorithm.

EEMD algorithm decomposes every $x^i(t)$ into the associated $IMF_k^i(t)$, to finally get IMF_k by averaging each $IMF_k^i(t)$.

$$\overline{IMF}_k = \frac{1}{N}\sum_{i=1}^{N} IMF_k^i(t) \tag{2.13}$$

Theoretically, as the mean of the white noise is zero, the impact of this should be eliminated by calculating 2.13. Nevertheless, Torres et al. found that a large number of $x^i(t)$ are needed to completely remove the white noise impact. Because of that, Torres et al. developed an updated version called CEEMDAN [14].

CEEMDAN algorithm overcomes this issue as well as the model-mixing. The algorithm is described as follows [16]

Let $E_k(.)$ be the operator that produces the k-th mode obtained by EMD, and let $w^i(t)$ be a realization of a zero-mean unit variance white noise.

Step 0: For every $i = 1, 2, \cdots, N$ decompose each $x_i(t) = x(t) + \epsilon_0 w_i$ by EMD until the first CEEMDAN IMF is obtained as,

$$\overline{IMF}_1 = \frac{1}{N}\sum_{i=1}^{N} IMF_{i1} \tag{2.14}$$

Step 1: Calculate the residue for the first stage ($k = 1$).

$$r_1(t) = x(t) - \overline{IMF}_1 \tag{2.15}$$

Step 2: For every $i = 1, 2, \cdots, N$, decompose each $r_1(t) + \epsilon_1 E_1(w_i(t))$ by EMD, until its first mode, and define the second CEEMDAN IMF as,

$$\overline{IMF}_2 = \frac{1}{N} \sum_{i=1}^{N} E_1(r_1(t) + \epsilon_1 E_1(w_i(t))) \tag{2.16}$$

Step 3: For $k = 2, 3, \cdots, K$ calculate the k-th residue,

$$r_k(t) = r_{k-1}(t) - \overline{IMF}_k \tag{2.17}$$

Step 4: For every $i = 1, 2, \cdots, N$ decompose each $r_k(t) + \epsilon_k E_k(w_i(t))$ by EMD, until its first mode, and define the $(j+1)$-th CEEMDAN IMF as,

$$\overline{IMF}_{j+1} = \frac{1}{N} \sum_{i=1}^{N} E_1(r_k(t) + \epsilon_k E_k(w_i(t))) \tag{2.18}$$

Step 5: Go to step 4 for the next k.

Repeat steps 4 to 6 until the obtained residue can no longer be further decomposed. The final residue is,

$$r_K(t) = x(t) - \sum_{k=1}^{K} \overline{IMF}_k \tag{2.19}$$

Therefore, the original time series can be expressed as,

$$x(t) = \sum_{i=1}^{K} \overline{IMF}_k + r_K(t) \tag{2.20}$$

The parameter ϵ allows the selection of the SNR at each stage.

2.5 Training, evaluation, and test

As in any regular Supervised Learning problem, in order to perform a prediction/classification task, the data set should be split into 3 sets:

- **Training set:** Sample of data used for training purposes.
- **Evaluation set:** Subset of the original dataset used for evaluating the fit of the model on the training set and fine-tune the hyperparameters.
- **Test set:** Sample of data used to evaluate in an unbiased way the final model performance.

One of the most used methodologies to split the original dataset into the 3 subsets mentioned above, is the k-fold Cross-Validation. Given the nature of the dataset used here, i.e., a time series, the beforementioned method is not applicable, as one of its main characteristics is to randomly split the data into k folds, breaking any time-dependency existing in the observations. Because of that, we'd take an iterative approach defining a fixed training, validation, and test set in the 1st step, and increasing and moving that 1 period in each iteration. This technique is known as the *walk-forward approach*.

2.5.1 Walk-forward approach

This methodology was firstly introduced by R. Stein in 2007 for validating credit default models [13]. This approach relies on using out-of-sample and out-of-time testing to evaluate the final performance of a model, which have been proved as a more reliable strategy compared to the typical *hold-out sample* technique.

The walk-forward procedure works as follows:

1. Select a fixed training set.
2. Define the validation set with the next d contiguous days to the set defined in step 1.
3. Fit the model using the defined training set in step 1. Perform the validation of the model with the settings defined in step 2.
4. Once the model is validated, create a prediction using as input the training set plus d. Save the prediction as part of the result set, used later on to evaluate the out-of-sample model performance.
5. Increase the training set in step 1 by 1 period and repeat steps 2 to 4 to retrain the model with the last data available and add the new predictions to the result set in every iteration.

By leveraging this methodology we are simulating the out-of-sample and out-of-time performance, re-training, and re-validating the model with each new piece of data that we are getting.

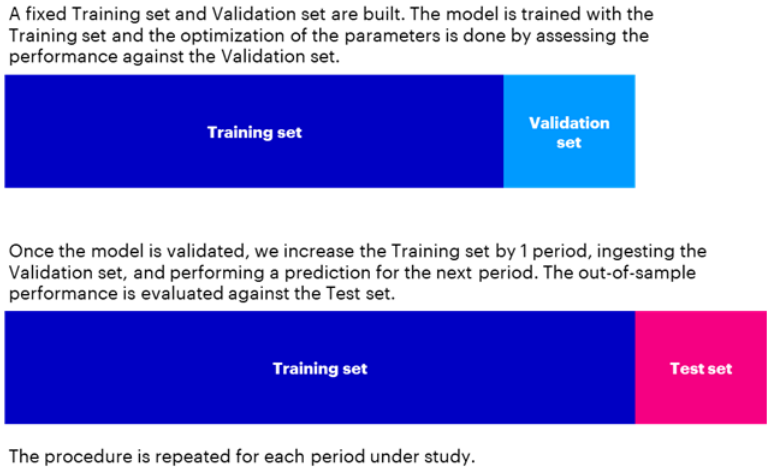

Figure 2.1: Walk-forward iterative process used in this study as the backtesting technique.

2.5.2 Performance metrics

Along with the study, there are 2 types of performance metrics used to assess performance at 2 different levels:

- **Global metrics:** Used to assess the performance of the forecasting strategies and compare among them.
- **Training and validation metrics:** Used during the training and validation phases to minimize the objective function and evaluate the performance of the model with the different hyperparameters.

Global Metrics

In order to evaluate the overall Forecasting Strategy performance and do comparisons between them, 2 different metrics have been used.

- **RMSE:** Square root of the mean squared error (MSE).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}} \tag{2.21}$$

- **FCA:** Performance metric used by the Industry to assess in a percentual way, how accurate the predictions are.

$$FCA = min\{0; 1 - \frac{\sum_{i=1}^{N}|\hat{y}_i - \mathbf{y}_i|}{\sum_{i=1}^{N} y_i}\} \tag{2.22}$$

Training and validation metrics

Depending on the model used, the training and validation metrics are different.

- **ARIMA:** The **auto.arima** function from the *forecast* R package, estimates several ARIMA models with optimized parameters to the time series. The metric driving model selection is AIC as described in Eq 2.5.
- **XGBoost and Adaptive Trees:** Given the learning task to be performed by the algorithm, i.e. a regression task, the RMSE is the objective function to be minimized and the metric used to assess the in-sample performance during training, the validation, and out-of-sample performance. Refer to Eq. 2.22 for further details.

2.6 Data used

The oil price index to be forecasted is the daily WTI Crude Oil Spot Price. WTI is a US blend of several streams of domestic light sweet crude oil. The centralized point of delivery for WTI is located in Cushion, Oklahoma which has a storage capacity of around 91 Mbbl. Although the pricing and delivery of these contracts are done in the US, the WTI index is used as a benchmark for energy markets worldwide.

The time series used is reported on daily buckets (Monday to Friday) and so will be the target feature to be forecasted. The time horizon used is between Jan 1986 and the 3rd week of 2021.

As XGBoost and Adaptive Trees rely on features in order to train the models and calculate predictions, the original Time Series is used to artificially generate lagged values of it, i.e., shift the original data set backward from 1 to 200 periods, so in order to predict $y_d = f(x_{d-1}, x_{d-2}, \cdots, x_{d-100})$.

For both ARIMA and Decomposition-Ensemble methods, from the complete data set, periods going from Jan 1986 till week 9 in 2020 have been used as the initial training set. Then, using the Walk-forward approach described in Section 2.5.1, the training set is increased by 5 days until week 2020.32. In each iteration, a 5-day forecast is calculated and compared with the actual WTI price.

2.7 Data cleansing and transformations

Missing values' extrapolation has been the only data cleansing technique applied. From the original Time Series of dimension 9115, there are 310 missing values, all of them related to US bank holidays. The following logic has been used:

- For isolated days in a week with a missing value, the average between $d-1$ and $d+1$ has been used.
- For cases where more than 1 consecutive day is missing, the average between $d-2$ and $d-1$ has been used.

In terms of data transformations, as tree-based algorithms are robust to outliers, we did not apply any. This might lead to a loss in performance for ARIMA, but in order to do a fair comparison between all the applied techniques, the author decided to not apply any transformation.

2.8 Forecasting strategies

After reviewing the main techniques and data that support this report, in this section, we'll go through the 4 forecasting strategies that will be leveraged to provide a prediction over the WTI oil prices.

Table 2.1: Forecasting strategies used during the study and compared to assess the best performer when generating 5-setp ahead daily forecast for the WTI price.

	Name	Description
1	CEEMDAN-Adaptive Trees	The original WTI time series is decomposed in n IMFs and Adaptive Trees are used to calculate the forecast for each component. The final forecast is an ensemble of the one calculated for each IMF.
2	CEEMDAN-XGBoost	The original WTI time series is decomposed in n IMFs and XGBoost is used to calculate the forecast for each component. The final forecast is an ensemble of the one calculated for each IMF.
3	XGBoost	Regular XGBoost is applied directly into the WTI time series to calculate the 5-step ahead daily forecast.
4	ARIMA	The ARIMA model is applied directly into the WTI time series to calculate the 5-step ahead daily forecast. ARIMA parameters are estimated automatically by leveraging the *auto.arim()* function from the **forecast** package.

2.8.1 Strategy 1 and 2

The first two forecasting strategies are based on the Decomposition-Ensemble framework which leverages the CEEMDAN algorithm to decompose the original Time Series into several IMFs plus a residual (as described in section 2.4), in combination with a Machine-Learning-based forecasting technique applied for each IMF/residual. Finally, an ensembling of each forecast is performed to get a final prediction for the target variable.

In this study, the ML-based algorithms used to perform the forecast for each IMF are XGBoost and Adaptive Trees, being the latter, in combination with CEEMDAN, a new technique never used before for prediction over commodity' prices during periods with structural changes (like the one in 2020 because of the Covid19). A graphical representation of the Decomposition-Ensemble framework can be found in Figures 2.2 and 2.3.

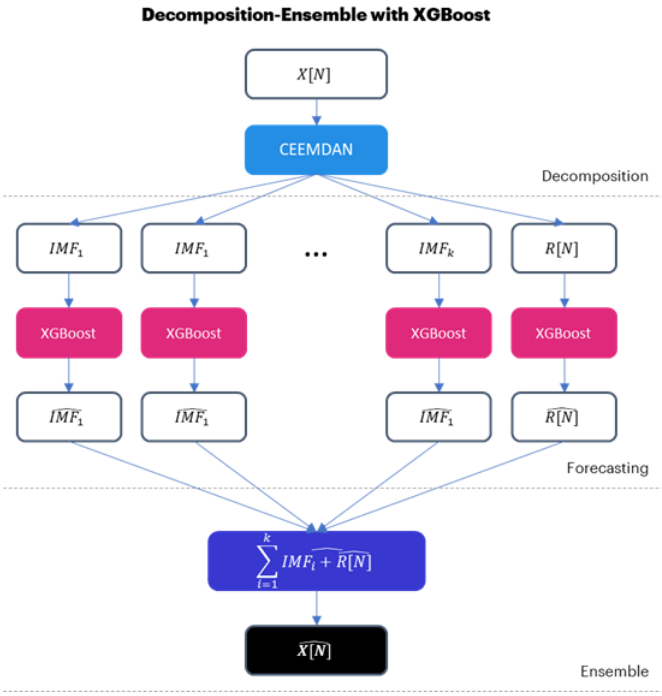

Figure 2.2: High-level Decomposition-Ensemble framework to calculate predictions based on XGBoost.

Backtesting

The technique used to simulate the past performance of the Decomposition-Ensemble strategy has been the Walk-forward approach as described in section 2.5.1. The iterations performed are the same as the ARIMA in section 2.8.3.

As a result, we will have a weekly forecast in daily buckets calculated every week starting from week 2020-09 till week 2020.32. This set of predictions

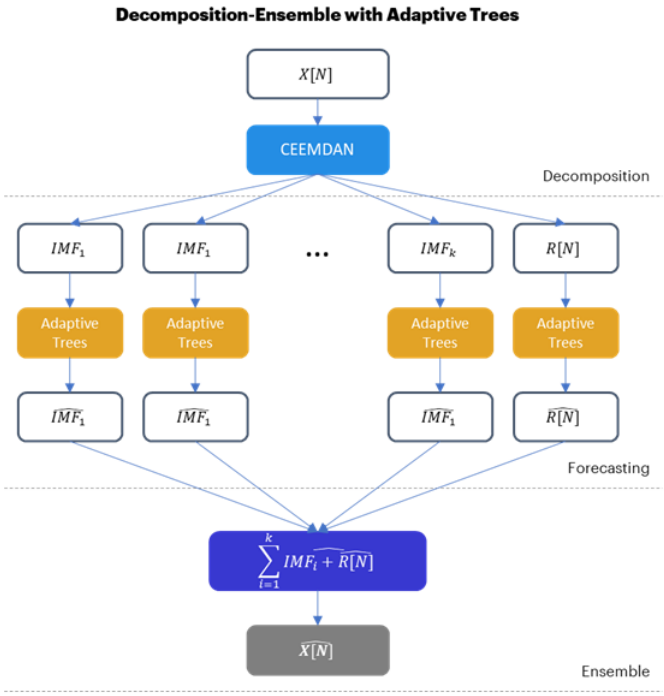

Figure 2.3: High-level Decomposition-Ensemble framework to calculate predictions based on Adaptive Trees.

are the ones used to assess the forecast performance and compare it with the benchmark model.

Model training and fine-tuning

The Model training phase is the most critical step in any ML Supervised Learning problem. During the Model Training, we provide the algorithm with examples and the associated labels with the aim to find the best way to accurately predict the labels while minimizing the algorithm's objective loss function.

During this phase, the user should also define the values of the model's hyperparameters, as these cannot be estimated from data. There are several techniques to get the optimal values of the hyperparameters, in this study we are using Grid Search.

Grid Search is a brute-force technique, which simply makes a complete search over a given subset of the hyperparameters' space. One of the main shortfalls of this method is that for high-dimensional spaces, the computation time can be big. The lightest version of Grid Search is Random Search, which just visits a random subset of the complete hyperparameters space. Random Search can outperform a regular Grid Search, especially if only a small number of hyperparameters affect the performance of the machine learning algorithm [9]. Because of the limitations in computing power, the number of hyperparameters

we will fine-tune is limited as well as its ranges. Therefore, the hyperparameter's space is smaller, and we can apply Grid Search.

The following hyperparameters have been optimized in XGBoost and Adaptive Trees:

Table 2.2: XGBoost hyperparameters to be optimized.

Hyperparameter	**Description**	**Grid Search range**
η	Controls the learning rate, i.e., scale the contribution of each tree by a factor of $0 < \eta < 1$ when it is added to the current approximation. Used to prevent overfitting by making the boosting process more conservative.	[0.01, 0.25]
Boosting iterations	Maximum number of boosting iterations.	{200}
ex-ante θ	Value given to the θ parameter in Eq. 2.10.	{10,12,15}

Setting the Adaptive Tree algorithm

As the Adaptive Tree is a quite new prediction technique, as of now, there is not a standard function in R nor Python to be leveraged. Nevertheless, by tweaking the *xgboost()* function from the **xgboost** R package, the Adaptive Tree described in [10] can be set up.

During the XGBoost training phase with the *xgboost()* function, the **weight** argument should be specified with a numeric vector including the ex-ante weights for each observation in the training set. The ex-ante weights are calculated as described in Eq 2.10.

Iteration and Computation

In order to calculate the 5-step ahead daily forecast for the weeks in scope, perform the hyperparameter's optimization, for each IMF, to finally combine the results into an ensemble, several iterations have been performed. A high-level algorithm can be found in Algorithm 2.

Algorithm 1 Strategy 1 and 2 algorithm

```
Decompose X(t) into IMFs by CEEMDAN
Define hyperparameters matrix
p:= backtesting periods
h:= {1,2,3,4,5}
for each IMF do
    for each element in p do
        for each element in h do
            define training set
            define validation set
            for each combination of hyperparameters do
                train XGBoost / Adaptive Tree
                prediction using the model
            end for
            assess RMSE of the predictions with the validation set
            opt.par:= hyperparameters' set that minimizes RMSE
            move the training set one period forward
            define test set as the validation set moved 1 period forward
            train XGBoost / Adaptive Tree with opt.par
            perform 1-step ahead prediction
        end for
        accumulate predictions for each h
    end for
    accumulate results for each p
end for
accumulate results for each IMF
aggregate predictions by addind up results for each IMF to get the Ensemble
```

Given the complexity of the Algorithm 2, with 4 loops, in each of them, training several XGBoost and Adaptive Trees, it was impossible to run it on a regular computer. Just as a reference, do a test for 1 IMF, and 1 backtesting period took 12h in a Lenovo i5 vPRO with 8 cores and 8Gb RAM.

In order to accelerate the calculation, the whole algorithm was parallelized using the future_map function from the **furrr** package, and run into an EC2 AWS instance. More specifically, the m5a.12xlarge instance was used with 48 vCPU and 192Gb RAM on top of a Windows Server 2019 AMI. With those 2 actions, the full Algorithm 2 for XGBoost and Adaptive Trees took around 15h.

2.8.2 Strategy 3

In this strategy, a regular XGBoost is applied to the training set to calculate the daily forecast. The backtesting, model training, and fine-tuning phases are the same as the ones used for Strategies 1 and 2 described in Section 2.8.1.

Algorithm 2 Strategy 3 algorithm

```
Define hyperparameters matrix
p:= backtesting periods
h:= {1,2,3,4,5}
for each element in p do
    for each element in h do
        define training set
        define validation set
        for each combination of hyperparameters do
            train XGBoost
            prediction using the model
        end for
        assess RMSE of the predictions with the validation set
        opt.par:= hyperparameters' set that minimizes RMSE
        move the training set one period forward
        define test set as the validation set moved 1 period forward
        train XGBoost with opt.par
        perform 1-step ahead prediction
    end for
    accumulate predictions for each h
end for
accumulate results for each p
```

2.8.3 Strategy 4

As described in 2.1, we relied on the *auto.arima()* function from the **forecast** R package.

In order to mimic the backtesting approach by the Decomposition-Ensemble strategy described in Section 2.5.1 and 2.6, we have leverageg the following algorithm:

Algorithm 3 ARIMA algorithm

```
p:= backtesting periods
for each element in p do
    define training set
    define validation set
    estimate ARIMA model with training set
    training set := training set + validation test
    set test set as the period contiguous to the training set
    calculate 5-step ahead forecast
end for
accumulate results for each p
```

3

Results

3.1 Descriptive statistics

The WTI Spot Price data set used has 9,015 data points, starting in 20/01/1986 up to 07/08/2020. A graphical representation of the time series can be found in Figure 3.1.

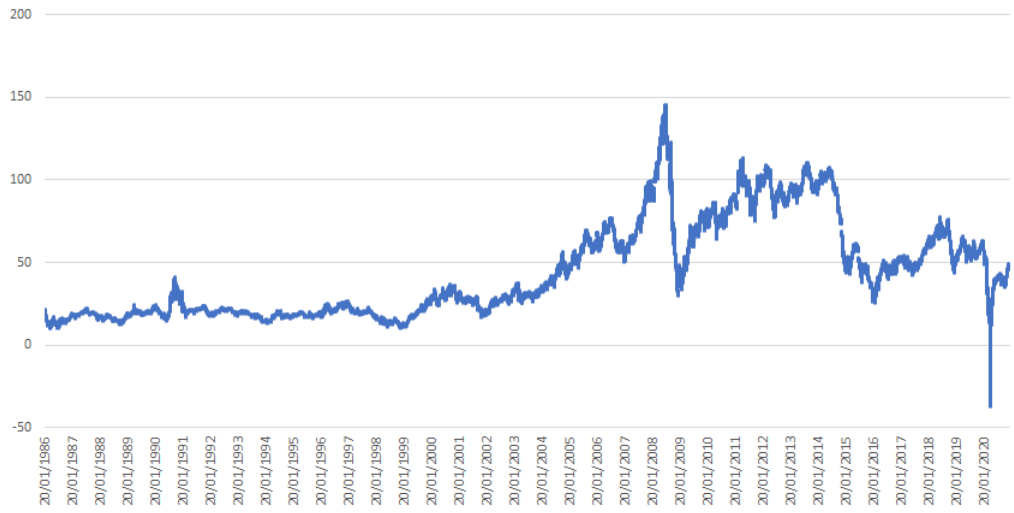

Figure 3.1: Daily evolution of the WTI Spot price from Jan 1986 till Aug 2020.

As ilustrated in Figure 3.2, there has been 5 major disruptions in WTI prices during the last 15 years:

- The big peak (yellow): All oil-reference index had a maximum price by that year, with a record of 145.85 USD on North Sea Brent.
- The 2008 crash (light blue): After the maximum in July 2008, a sell-off burst began after President Bush announced the lift on the ban for offshore oil drilling in the US coasts [11]. By the end of that week, prices went down 11% and kept that trend for the rest of the year, more aggressively after the Lehman Brothers bankruptcy. By Dec 2020, it reached 32 USD.

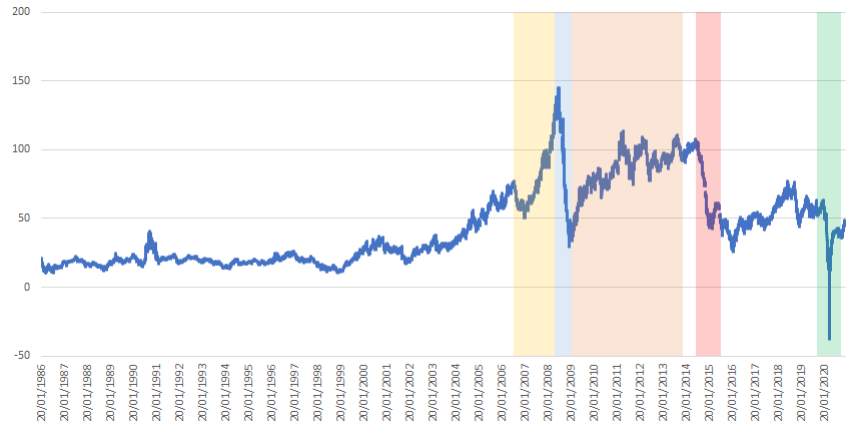

Figure 3.2: Daily evolution of the WTI Spot price from Jan 1986 till Aug 2020 with a focus on 5 major disruptions in prices during the last 15 years.

- Hydraulic fracturing era (orange): US economy was growing after the 2008 crisis, and hydraulic fracturing wells were drilled over the country.
- Oversupply (red): The gains in fracking's efficiency enabled oil suppliers to flood the market with cheap oil.
- The Covid19 big crash (blue): The big reduction in demand worldwide due to lockdowns, the lack of response from suppliers to cut production ended up in a dramatic reduction in stock capacity in Cushion, which triggered a panic to investors that start selling the M+1 WTI Future contracts. The sale momentum was so big that translated into negative oil prices.

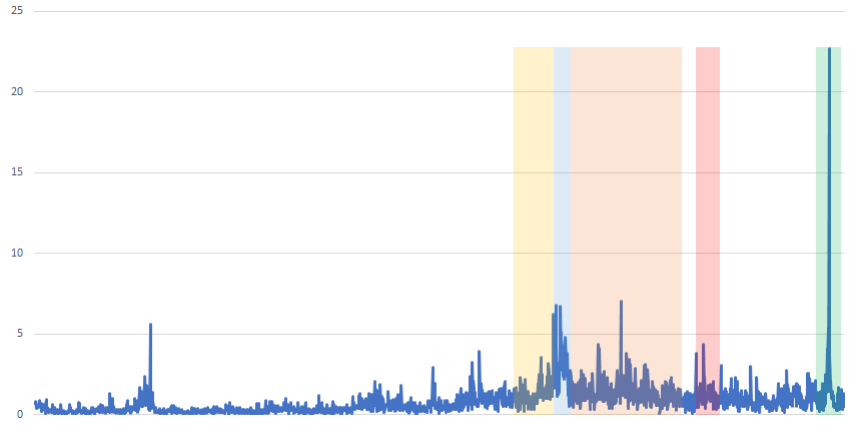

Figure 3.3: Within-week standard deviation evolution.

Figure 3.3 shows in a more analytical way the periods with high volatility, i.e. with structural changes by showing the evolution of the within-week standard deviation. The Covid19-related break is the biggest one by far.

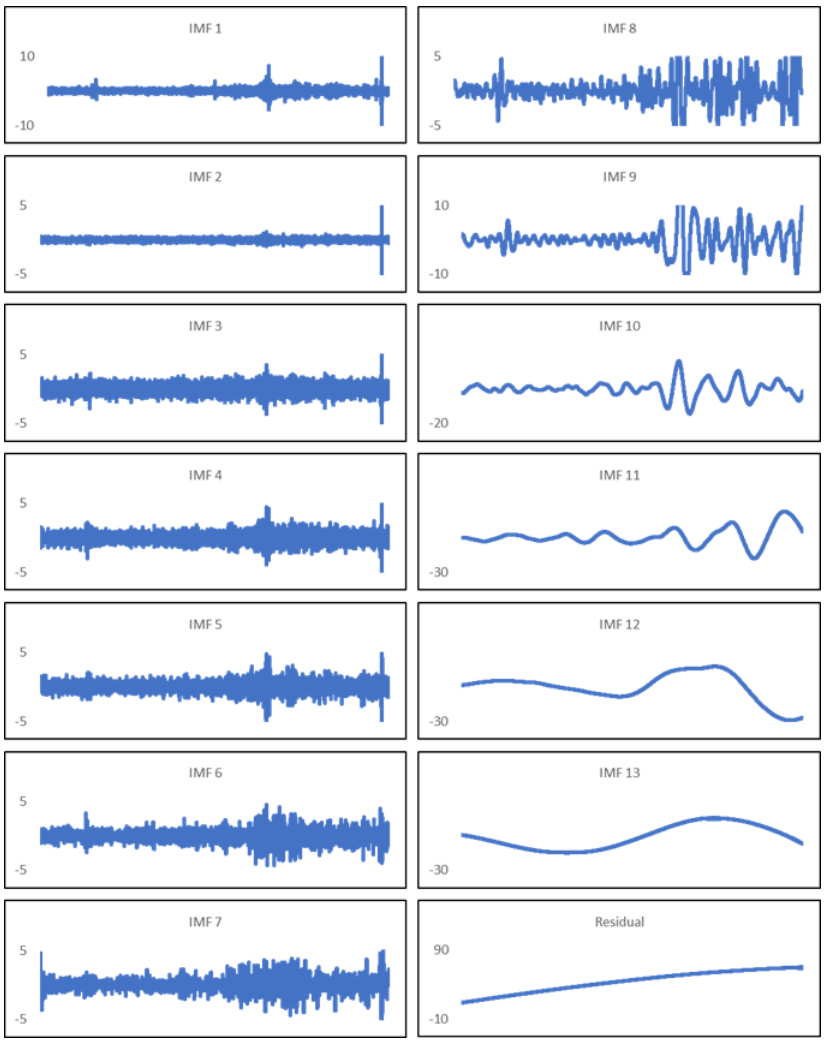

Figure 3.4: CEEMDAN of the WTI price into 13 IMFs and a residual.

3.2 Number of IMFs

As proposed in the original EMD study by Huang et al. (1998) [15], the residual can be seen as the long-term trend, because of that, and following the same approach as other papers leveraging CEEMDAN, we have been increasing the number of IMFs starting with 5 until we got a residual showing a linear trend behavior.

As shown in Figure 3.4, we achieve the beforementioned condition, i.e., getting a residual with a clear linear trend, after 13 IMFs plus 1 residual, the first 8 showing a high-frequency pattern whilst the remaining 5, a low-frequency one. As a result, for all the forecast strategies used in this study relying on CEEMDAN, the WTI time series has been decomposed into 13 IMFs and 1 residual.

Table 3.1: Descriptive statistics for the WTI price IMFs and residual.

	Mean	Variance	Variance percentage	Perc of 0-crossing points
IMF 1	< -0.001	0.419	0.070	65.109
IMF 2	-0.001	0.052	0.009	43.976
IMF 3	0.002	0.392	0.066	46.544
IMF 4	-0.001	0.473	0.079	25.368
IMF 5	0.001	0.659	0.110	13.573
IMF 6	-0.001	1.025	0.172	7.000
IMF 7	-0.004	1.644	0.275	3.401
IMF 8	0.044	6.426	1.075	1.635
IMF 9	0.008	18.755	3.138	0.790
IMF 10	-0.156	19.216	3.215	0.329
IMF 11	1,179	57.233	9.577	0.121
IMF 12	-0.140	150.245	25.141	0.033
IMF 13	-0.832	105.578	17.667	0.208
Residual	43.979	235.495	39.406	0.000

Table 3.1 supports the number of IMFs chosen by visual assessment:

- Residual is not showing a wave pattern as mean is > 0.
- As we get closer to the residual, the variance increases heavily, especially from IMF12 onwards.
- In the residual there is not any 0-crossing, confirming the non-waived behavior.

3.3 Weekly forecast performance

In this section, we will explore the weekly forecast performance of the 4 forecast strategies analyzed. In order to analyze results at this aggregation level, we are accumulating the daily errors within a week for both the RMSE and FCA.

Although the focus of this study is the daily forecast performance, by analyzing the weekly, we can get an overview and identify those periods with the biggest gap between the strategies. A complete view over the weekly performance can be found in Table 3.2.

A few facts to summarize the weekly results:

- In 19 out of the 24 weeks analyzed (79%), the Machine Learning approaches, are outperforming ARIMA.
- In 11 out of the 24 weeks analyzed (46%), regular XGBoost shows the best performance than ARIMA.

Table 3.2: Weekly forecast performance for the 4 forecast strategies analysed in this study.Empty cells in week 2020.16 because with negative values FCA cannot be calculated.

		2020.09	2020.10	2020.11	2020.12	2020.13	2020.14	2020.15	2020.16	2020.17	2020.18	2020.19	2020.20
RMSE	ARIMA	2.38	8.84	8.37	2.99	11.31	4.98	1.68	24.96	3.74	5.34	2.92	5.11
	XGBoost	3.03	7.74	7.98	6.89	7.36	3.64	1.47	24.84	4.5	3.13	3.18	2.13
	CEEMDAN-XGBoost	2.01	2.55	5.09	4.28	3.17	1.99	1.61	20.42	3.21	3.66	1.86	1.71
	CEEMDAN-Adaptive Trees	2.70	3.24	5.14	5.37	4.04	1.68	2.31	19.35	1.54	4.88	1.91	0.84
FCA	ARIMA	95.12	72.96	68.89	87.02	56.38	80.05	92.44		80.15	78.20	91.26	84.78
	XGBoost	93.88	76.55	71.85	71.00	70.88	86.48	93.96		74.66	87.94	90.82	94.41
	CEEMDAN-XGBoost	95.98	93.49	81.77	81.88	87.45	92.15	93.40		82.02	86.08	94.83	95.03
	CEEMDAN-Adaptive Trees	94.90	90.89	81.98	77.33	82.99	94.31	89.54		91.38	79.97	94.61	97.86

		2020.21	2020.22	2020.23	2020.24	2020.25	2020.26	2020.27	2020.28	2020.29	2020.30	2020.31	2020.32
RMSE	ARIMA	1.14	2.93	1.44	1.72	1.08	1.43	0.43	0.43	0.81	0.83	1.42	0.7
	XGBoost	1.50	1.43	2.07	1.82	0.78	1.74	1.53	0.69	0.91	2.42	1.23	2.58
	CEEMDAN-XGBoost	1.21	2.44	1.48	1.41	1.16	0.90	0.55	0.46	0.40	2.82	2.52	1.74
	CEEMDAN-Adaptive Trees	1.08	1.84	0.46	0.92	1.06	1.23	1.18	1.45	0.57	2.42	1.75	1.15
FCA	ARIMA	97.23	92.92	96.76	96.12	97.44	96.63	99.08	99.13	98.31	98.40	96.80	98.56
	XGBoost	96.17	96.71	95.35	95.32	98.18	96.00	96.77	98.41	98.23	96.09	97.19	94.33
	CEEMDAN-XGBoost	96.95	95.79	96.66	97.10	97.44	98.13	98.82	99.12	99.15	93.78	94.27	96.36
	CEEMDAN-Adaptive Trees	97.33	96.05	98.92	97.84	97.48	97.30	97.86	97.26	98.75	95.61	96.03	97.77

- In 18 weeks out of the 24 (75%), the CEEMDAN-XGBoost outperforms XGBoost, i.e., the decomposition-ensemble strategy is clearly adding value in terms of forecast performance.
- In 12 out of 24 weeks (50%), CEEMDAN-Adaptive Tree is performing better than CEEMDAN-XGBoost, especially in high-volatility periods.

From the points above we can clearly state that ML-based learning shows a better performance than time series-based ARIMA when analyzing daily forecasts aggregated to weekly buckets. We can conclude the same when comparing the regular XGBoost vs Decomposition-Ensemble approach.

3.4 Daily forecast performance

In this section, we are analyzing the performance of the 5-step ahead forecast calculated every first day of the week.

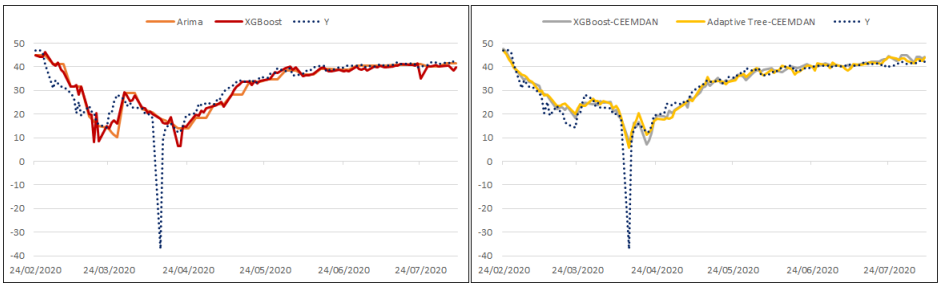

Figure 3.5: Daily forecast evolution for the 4 strategies in scope. UOM: US Dollars.

In Figure 3.5, the daily evolution of the forecast is shown for the 4 forecast strategies. In order to better analyze the daily results, we split the 24 weeks in scope into 3 periods:

3.4.1 First Covid19 hit

(24/02/2020 - 07/04/2020). During these 4 weeks, the first hit of Covid19 impacted the WTI price due to the several lockdowns in China, Europe, and the US causing a plunge in oil and petroleum products' demand.

As shown in Figure 3.6, at a high level, the two Decomposition-Ensemble strategies remain closer to the actual values, in a smoother way, without reacting too much to ups and downs. When comparing both, there is not a significant difference between them, nor during more stable days, nor periods with some volatility.

On the other hand, the regular XGBoost is clearly showing more erratic behavior, overreacting to drops and peaks and sometimes showing a non-expected

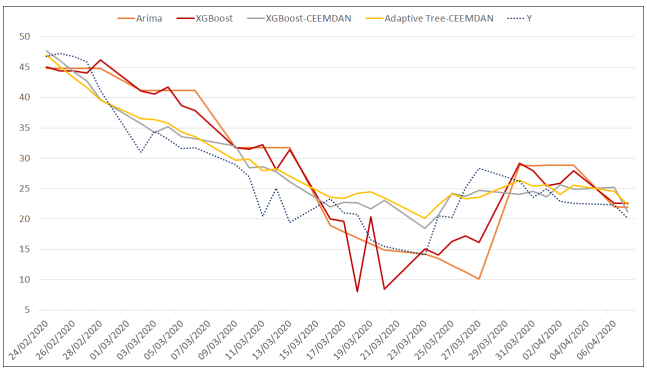

Figure 3.6: Daily WTI price evolution during the first analised period. UOM: USD.

patterns like the one during 18/03/2020 - 21/03/2020.

ARIMA reacts sluggishly and keeps a very conservative estimation within the studied period. It is also noticeable the flat pattern reproduced for entire weeks, which gives the calculation made by ARIMA a stepped way of looking.

Table 3.3: Aggregated FCA and RMSE results for the first analysed period, i.e., 24/02/2020 - 07/04/2020

	ARIMA	XGBoost	CEEMDAN-XGBoost	CEEMDAN-Adaptive Tree
FCA	79.7%	81.3%	90.2%	88.7%
RMSE	7.02	6.23	3.32	3.83

At an overall level, Table 3.3 confirms that for this first period, Decomposition-Ensemble strategies outperform ARIMA and the regular XGBoost by far. Nevertheless, CEEMDAN-Adaptive Trees shows a slightly worst behavior than CEEMDAN-XGBoost.

3.4.2 Historical drop

(08/04/2020 - 24/04/2020). This period comprises the days before the big crash and the ones just after the plunge on April 13th.

Figure 3.7 shows a similar picture as Figure 3.6: both Decomposition-Ensemble strategies fits better to the actuals than ARIMA and XGBoost.

Focusing on the big drop, CEEMDAN-Adaptive Trees performs marginally better than CEEMDAN-XGBoost, by reacting more aggressively to the plunge.

XGBoost shows again an overreacting behavior but just after the drop. ARIMA reproduces a quasi-linear forecast without capturing any movement in the WTI price.

In Table 3.4 one can assess the overall performance over period 2, which confirms what was seen for period 1: Decomposition-Ensemble outruns ARIMA and XGBoost -also during high-volatility periods. The difference between CEEMDAN-XGBoost and CEEMDAN-Adaptive Trees is negligible.

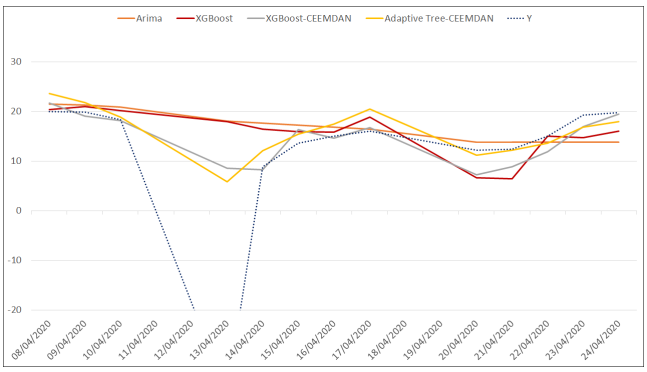

Figure 3.7: Daily WTI price evolution during the second analised period. UOM: USD.

Table 3.4: Aggregated FCA and RMSE results for the second analysed period, i.e., 08/04/2020 - 24/04/2020

	ARIMA	XGBoost	CEEMDAN-XGBoost	CEEMDAN-Adaptive Trees
FCA	41.0%	40.1%	55.8%	56.0%
RMSE	15.68	15.68	12.80	12.08

3.4.3 Recovery and stabilization period

(27/04/2020 - 07/08/2020). These weeks include the recovery from the big drop and the short-term stabilization around 40USD/bbl (prices went higher from onwards and are still showing a positive trend as of Sep 2021).

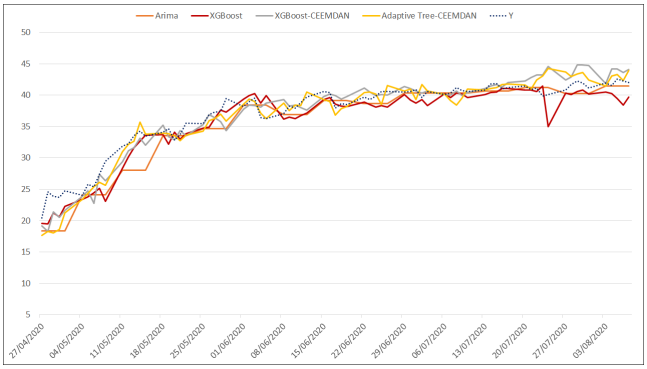

Figure 3.8: Daily WTI price evolution during the third analised period. UOM: USD.

Results for period 3 (in Figure 3.8) do not add new insights not already captured before:

- Decomposition-Ensemble framework fits better to the actuals than ARIMA and XGBoost.
- There is not a noticeable difference performance-wise between CEEMDAN-XGBoost and CEEMDAN-Adaptive Trees.
- XGBoost shows a weird behavior, not reproducing any familiar actual pattern.

- ARIMA proposes a smoother forecast with flat lines in consecutive days within a week.

Table 3.5: Aggregated FCA and RMSE results for the third analysed period, i.e, 27/04/2020 - 07/08/2020

	ARIMA	XGBoost	CEEMDAN-XGBoost	CEEMDAN-Adaptive Trees
FCA	95.5%	95.8%	96.3%	96.6%
RMSE	2.39	1.96	1.86	1.83

Aggregated results for period 3 in Table 3.5 confirms the conclusions made in the 3 points above.

3.5 Summary

After analyzing the forecast performance results for the 4 strategies in scope on weekly and daily buckets, and in different periods (period 1: negative trend with medium volatility; period 2: big drop with high-volatility; period 3: positive trend with low volatility), we can conclude:

- The Decomposition-Ensemble strategy (whether combined with XGBoost or Adaptive Trees), clearly outperforms ARIMA and XGboost, more specifically, the former it's adding +8-9pp in FCA for period 1, +15pp in period 2, and 1-2 pp in period 3 compared to the latter.
- When comparing XGBoost and Adaptive Trees combined with CEEMDAN, there is not a significant difference performance-wise; the 2nd shows slightly better results in periods 2 and 3.
- Regular XGBoost is reproducing non-expected patterns and overreacting to peaks and drops. We can conclude that decomposing the original time series by CEEMDAN is adding around +8pp in terms of FCA.
- ARIMA is the forecasting strategy showing the lowest results in the 3 analyzed periods. The difference is not so big in periods with a stable pattern and low volatility, as ARIMA tends to produce a flat and smooth forecast.

4

Discussion

In this last chapter of the study we will provide an answer to the questions we asked ourselves in Section 1.2 and build a high-level roadmap and Business Case for one of the Oil Majors companies I have been working on over the last years. As the last step, we will briefly expose which are the improvements and next steps identified during this study.

4.1 Did we achieve our Objectives?

During the Introduction chapter in this report, we listed down the specific questions we wanted to get an answer for after analyzing the results (see Section 1.2). Let's review them one by one.

- *Are ML-based methods outperforming ARIMA when calculating daily forecasts for WTI spot price during periods with structural changes?*

  Clearly yes, at both weekly and daily buckets as shown in Section 3.3 and 3.4. At the weekly level, we saw that in $\approx 80\%$ of the 24 weeks analyzed, the ML-based methods outperformed ARIMA in terms of RMSE and FCA. In the same direction, at a daily level, and for the 3 periods analyzed (with different volatility patterns), the Decomposition-Ensemble strategy shows a higher performance than ARIMA, adding up to +15pp in FCA during the Historical drop.

- *Which is the added value of applying a decomposition algorithm to regular XGBoost in terms of forecast accuracy?*

  Adding the Decomposition algorithm to XGBoost is clearly increasing the forecast performance. When analyzing the results in weekly buckets, in $\approx 75\%$ of the 24 weeks analyzed, the CEEMDAN-XGBoost gets higher accuracy than the regular XGBoost. At the daily level, the CEEMDAN algorithm is adding up to +15pp in terms of FCA. Just as a reference, in a similar study [18], the authors found that adding

CEEMDAN to XGBoost improved the RMSE from an average of 1.26 to 0.41 for 1-step ahead daily forecast[1].

- *Can we prove that CEEMNDAN-Adaptive Trees is a better forecasting strategy than the others in terms of accuracy during periods with structural changes?*

  Unfortunately with the features used and the non-exhaustive training phase performed, we cannot conclude that CEEMDAN-Adaptive Tree is the best of the strategies. Analyzing the weekly and daily results we can see that there are no significant differences when compared with CEEMDAN-XGBoost. Just during periods with high volatility (like the historical drop), Adaptive Trees is outperforming the XGBoost one. Woloszko (2020) [10] conducted a study to forecast GDP for a bunch of European countries in monthly buckets with Adaptive Trees (without decomposition). For most of the countries analyzed, they were able to reduce RMSE by 50% by using Adaptive Trees when compared to Gradient Boosted Trees (similar to XGBoost). In our case, we really think that the performance of the Decomposition-Ensemble strategy has been hindered by the lack of computation power and usage of simple features during the training phase. A more extended discussion on this topic can be found in Section 4.3.

- *How can an Oil & Gas company implement the winning forecast strategy?*

  In Section 4.2.1 we provide a detailed answer to this question: how to implement an enhanced version of the Decomposition-Ensemble strategy and how this can be the first step towards a Digitalization journey towards a real Integrated Planning capability. We are also including the associated Business Case with benefits and implementation and running costs.

4.2 Roadmap and Business Case

Most of the Oil Majors are in a similar maturity level when it comes to Digitalization. These companies have historically been the ones with more revenues and benefits. Just as an example Royal Dutch Shell plc. reported annual revenues of 344,877 mUSD in 2019 and an EBITDA of 29,566 mUSD with an EBITDA profitability of 8.5% [12]. On the same page, Saudi Aramco was in 2019 the most profitable company in the world by large, with a net income of 111.1 bUSD vs. the 59.5 bUSD reported by Apple, the 2nd most profitable (this picture is different after the pandemic). Diving in these big profitability numbers, Oil Majors became big and complex organizations, not focusing too much on getting into the Digitalization journey.

This translated into a large amount of manual workload to execute processes, sometimes not relying on any system; decisions driven more by *gut feelings* rather than on data. In terms of forecasting capabilities, there is not any or

[1] This study was conducted in 2019 in a pre-Covid19 scenario with no structural changes.

really basic: Excel spreadsheets with some historical data and predictions done by moving averages with different time windows. In some of the analyses these companies perform, they just take an average crude oil price, making the consequent conclusions not robust at all.

4.2.1 A Roadmap towards Digitalization

The baseline (or starting point) is really set the basics, hence an implementation of the enhanced methodology we explored and the ingestion of it into their planning and trading processes may take some time. In fact, implementing the enhanced Decomposition-Ensemble methodology will just be one of the first steps into the Digitalization journey. As shown in Figure 4.1, we propose to achieve an *Advanced and Integrated Planning* in 4 years with a list of initiatives to reach that North Star divided into 4 groups: Planning capabilities, Analytics, Skills / Organization, and Technology.

Planning capabilities

- **ERP Integration:** Crude Oil forecast in daily buckets is shared with the ERP system so all the organization is leveraging the same information when steering operational and trading activities. Ad-hoc analysis and reporting are based on the same underlying information.
- **Enhance Operations Planning:** Crude Oil forecast is incorporated into the existing planning tools. This information is used by such tools to enhance planning processes: adjust production throughput, manufacturing capacity, inventory levels, procurement, and transport planning.
- **Scenario Planning:** Advanced planning tools are implemented with Scenario Planning capabilities (e.g. Kinaxis, E2Open). Users can simulate the impact of different parameters (e.g. oil price) on operational performance and costs. The generation of different scenarios allows the users to make better decisions based on data.
- **Digital Twin:** Taking Scenario Planning to the next level. Digital Twin allows the user to completely mimic a physical Supply Chain in a virtual space and test in real-time different configurations and assess the impact. The configuration made on the virtual environment is executed in the physical Supply Chain.

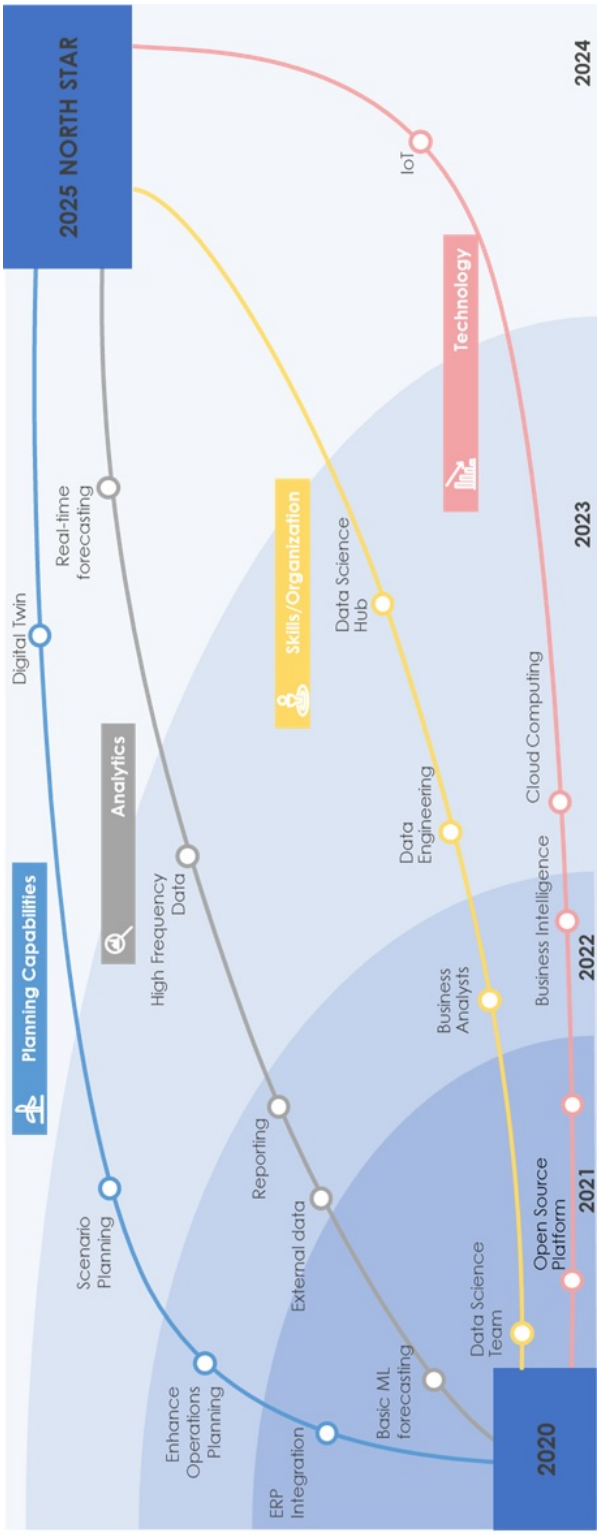

Figure 4.1: List of initiatives and enablers proposed to reach an Advanced and Integrated Planning Company in 4 years.

Analytics

- **Decomposition-Ensemble:** An enhanced version of the methodology presented in this report is implemented. It can be upgraded with the enhanced version presented in Section **??**.
- **External data:** The beforementioned methodology can be improved by adding external data into the modeling, e.g., macroeconomic data for the US and Europe, prices of other crude oil grades and commodities, crude oil stock capacity in the US, crude export/imports, refinery capacities, etc.
- **Reporting:** In order to assess the forecast performance a user-friendly and robust reporting layer is implemented. On top of it, a Continuous Improvement process can be arranged to identify corrective actions and hence, improve the forecast performance.
- **High-frequency data:** Hourly or real-time price data is leveraged into the Ensemble-Decomposition framework. As a result, the hourly or real-time forecast is calculated, fostering rapid response in the Supply Chain.

Skills / Organization

- **Data Science Team:** Most of the O&G companies are still lacking a dedicated Data Science team and are simply relying on in-house employees with analytical acumen. In order to successfully implement all the initiatives, a dedicated Data Science team should be built.
- **Business Analyst:** Although all companies have some spare FTEs under the Business Analyst or Business Intelligence profiles, a dedicated team will allow the organization to effectively communicate the results and outcomes of the Data Science team across the company. This team can also be steering the Continuous Improvement function.
- **Data Engineering:** As the models and data used to become more complex, dedicated Data Engineers are required in order to effectively store and manage the large amounts of data required by the Data Science Team.
- **Data Science Hub:** As the organization becomes more mature, the usages of ML-based forecasting techniques can be expanded, and the Data Science team can become a sort of Data Science as a Service team. In such cases creating a central dedicated Hub will increase the efficiency when offering these services across the company.

Technology

- **Open Source Platform:** In the early stages of digitalization, the whole organization can rely on open source platforms for all analytics-related topics: R or Python for ML, MySQL for data storage, and Shiny for data visualization.

- **Business Intelligence:** In order to support Advanced Data Visualization functionalities, a dedicated tool should be implemented such as Tableau or Qlikview. These tools are way flexible than Shiny when it comes to scaling and connection to different and diverse data sources.
- **Cloud Computing:** When executing complex ML models traditional servers become obsolete. To sustain the big computing power required, we recommend the usage of cloud computing solutions such as Microsoft Azure or Amazon AWS.
- **IoT:** As the infrastructure supporting becomes more advanced, including advanced data storage solutions and cloud computing, the organization can do a big step forward by installing sensors in their facilities in order to capture real-time operational data. This information can be leveraged by the Data Science team to enhance their models and tackle new problems, i.e., Predictive Asset Maintenance, Spare Parts Inventory Optimization, etc.

4.2.2 High-level Business Case

One of the main outcomes of building a roadmap of initiatives is the Business Case. In the Business Case, we estimate the potential impact in Revenues, Cost, and CAPEX of the different initiatives to reach the future statement, and these are confronted with the cost of implementing the solution.

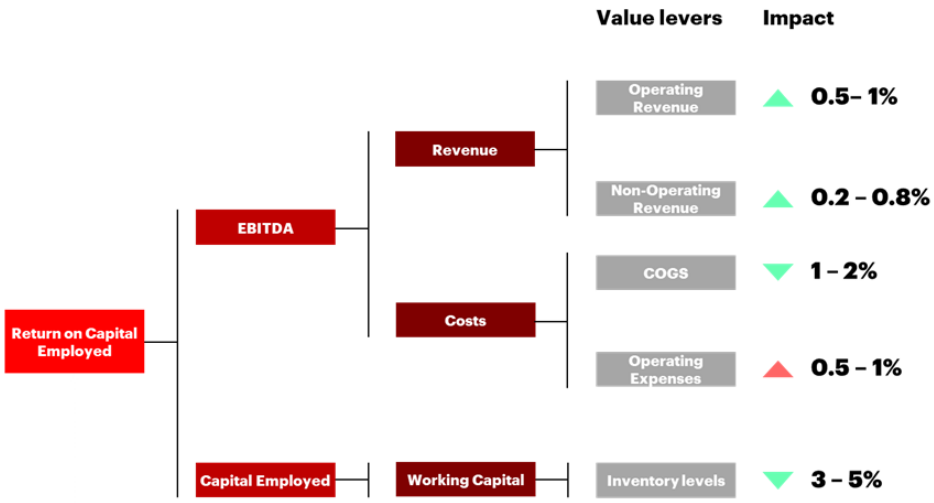

Figure 4.2: Value Tree with the value levers taht are mostly impacted by the initiatives in Section 4.2

The Impact shown in Figure 4.2 represent the increase or decrease in percentage over the associated Value Lever if the company is able to successfully

implement the Initiatives stated in the Roadmap.

In order to get a glimpse of the impact this may have on an Oil Major, let's apply the Business Case into Shell's 2019 financial figures as per the 2019 Annual Report (the last one before Covid19 outbreak).

Table 4.1: 2019 values reported in the Shell plc 2019 annual report and the associated Business Case. Numbers in brackets indicate negative numbers. Values out of the 2019 Shell Annual Report [12]. UOM: mUSD.

Value lever	Adressable P&L	Impact	Business Case
Operating Revenue	58,709	0.5 -1%	294 - 587
Non-Operating Revenue	7,396	0.2 - 0.8%	15 - 59
COGS	-47,599	1 - 2%	(476) - (952)
Operating Expenses	8,040	0.5 - 1%	40 - 80
Inventory Levels	-3,859	3 - 5%	(116) - (193)

By adding the 2 first rows of Table 4.1, we get a total increase in revenue of 309 - 646 mUSD. In the opposite direction, by aggregating figures in the 3rd and 4th row, we get the reduction in total expenditure of (436) - (872) mUSD. The reduction in crude oil Inventory Levels is translated into a reduction of (116) - (193) in the Balance Sheet.

Assuming the rest of the P&L items remain the same, the above figures translate into an increase in profitability[2] of 3 - 6% (2019 EBITDA 25,485 mUSD; updated EBITDA 26,230 - 27,003 mUSD).

All these numbers, however, do have an associated implementation and running cost.

Table 4.2: Evolution of the implementation and running costs of the Roadmap. Assumptions in Appendix 1. UOM: mUSD

Cost item	2021	2022	2023	2024
Headcount cost	0.60	0.45	0.60	2.03
Consultancy Cost	18.75	15.63	9.38	3.13
Technology Cost	4.00	8.00	10.00	15.00
Total Cost	23.35	24.08	19.98	20.15
Benefits	1,286	1,108	1,112	1,112
Business Case	1,263	1,108	1,112	1,112
Cumulative Business Case	**1,263**	**2,371**	**3,843**	**4,594**

Table 4.2 shows how the implementation and running costs are evolving through the 4 years of Roadmap's deployment. As stated in the last row of the table, even with the high costs for the first 2 years, the benefit of deploying the Roadmap clearly overcomes the cost.

[2]Measured as EBITDA.

4.3 Improvements and next steps

In this section, we will review which have been the main improvement points identified through the study and which are the potential next steps to further improve and start the implementation.

Regarding the identified improvement points, the most relevant has been the computation power constraint. Although an AWS EC2 instance was used, the high associated cost limited the number of hours the server was used, limiting the ability to test different parameters and features. When implementing this in a company, a bigger budget will be available and a more powerful EC2 instance (or similar) might be available on demand.

Because of the same reason, during the hyperparameter's optimization phase, the number of parameters and ranges to look for an optimal combination were low, limiting the potential forecast performance of the Decomposition-Ensemble methods. If more powerful computing resources are available, the number of hyperparameters to optimize and the reviewed optimization space will be bigger, increasing the forecast performance. This has been especially a pain point for Adaptive Tree, where just 3 values of the θ parameter have been reviewed during the training phase, limiting the potential of this methodology in terms of forecast performance.

With bigger computation power, a longer forecast horizon can be calculated. Once the methodology is implemented and in production, for planning and trading purposes, a daily forecast for the next 3 months should be available (at least).

In order to make all the forecasting strategies comparable among them and understand the added value of each, the feature engineering remained basic, i.e., just using past values, simulating a kind of auto-regressive model like ARIMA. This approach reduced the required computation power and made the comparisons between methodologies fairer but, at the same time, hindered the performance of the most advanced techniques, mainly the Decomposition-Ensemble. When implemented, we strongly recommend extracting more features from the time series, e.g., day of the week, week indicator, month indicator, average/median values, min and max values within a week, moving averages, etc. External data can be also leveraged: macroeconomic metrics or O&G specific.

Related to the last point, with a broader set of features to train the models, a deeper study around features' importance can be conducted. One of the most user-friendly capabilities of the XGBoost implementation in R and Python is the possibility to estimate which is the importance of each feature used to train the model. In the present report, all features have been used. By removing features with low importance or those highly correlated, a reduction in complexity and risk of overfitting can be achieved, translating into more accurate results.

Regarding next steps, or how to start Roadmap's implementation:

- Before going any step further, the stakeholders may request a detailed

version of the Business Case. This is crucial in order to get buy-in from the organization. To do that we need to tailor down the cost items that might be impacted by the initiatives and use up-to-date financial information from the company. In the same way, a more detailed picture of the implementation and running costs should be provided. With these 2 pieces of information, the Business Case is more detailed and shows a more realistic view of the impact and associated costs.

- For each of the initiatives in the Roadmap, a plan should be built including tasks to be performed with timelines, owner of each of the tasks, benefit, and cost of implementing, initiative's sponsor, objective, and interdependencies with other *on-the-fly* initiatives.
- For the Decomposition-Ensemble initiative, the one of more interest in this study, all the improvement points stated above should be applied before going into implementation mode. Afterward, a pilot for 1 specific country or Business Unit should be performed in order to test the methodology, fine-tune if required, and proof value.
- A PMO function should be set up to ensure the Roadmap's realization and track status over all the initiatives being implemented.
- A few weeks before kicking off a new initiative, the required team should be mobilized, ensuring everyone in the team has a good understanding of the activities to be performed.
- Once an initiative is in implementation mode, perform regular meetings with all the task force in order to check the status and identify potential risks and roadblocks. Ensure a proper escalation path is established in order to raise any issue the team might face.
- During the implementation of each initiative and when closing, it is important to revisit the initiative's Business Case and assess the level of adherence to it and the initial plan. This might help in refining the plan and Business Case for the following initiatives to be implemented.

5

Glossary

- **ARIMA,** Autoregressive Integrated Moving Average
- **AWS,** Amazon Web Services
- **bbl,** one barrel of crude oil
- **bUSD,** billion USD
- **CAPEX,** Capital Expenditure
- **CEEMDAN,** Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
- **COGS,** Cost of Goods Sold
- **EBITDA,** Earnings Before Interest, Taxes, Depreciation, and Amortization
- **EEMD,** Ensemble Empirical Mode Decomposition
- **EMD,** Empirical Mode Decomposition
- **FCA,** Forecast Accuracy
- **FTE,** Full Time Employee
- **GBM,** Gradient Boosting Machine
- **GBT,** Gradient Boosting Tree
- **IMF,** Instrinsic Mode Function
- **IOC,** International Oil Company
- **IoT,** Internet of Things
- **Mbbl,** One million of crude oil barrels
- **ML,** Machine Learning

- **mUSD,** million USD
- **NOC,** National Oil Company
- **O&G,** Oil&Gas
- **OPEC,** Organization of the Petroleum Exporting Countries
- **OPEC+,** Extended Organization of the Petroleum Exporting Countries (including Russia)
- **P&L,** Profit&Loss
- **PMO,** Project Management Office
- **RMSE,** Root-Mean Square Error
- **SNR,** Signal-to-Noise Ratio
- **UOM,** Unit of Measure
- **WTI,** West Texas Intermediate
- **XGBoost,** Extreme Gradient Boosting

Appendices

A

Business Case assumptions

The aim of this appendix is to list down all the assumptions made when calculating the Business Case and the associated implementation and running costs.

- **Addressable P&L:** Portion of the Revenue and Cost stated in the 2019 Shell Annual Report which can be impacted by the initiatives. It was estimated that the main impacted area within the company is the Downstream Manufacturing Business Unit, representing $\simeq 17\%$.
- **Impact:** Values on the 3rd column in Table 4.1 come from market benchmarks. Data source: confidential.
- **COGS:** Calculated as the sum of *Purchases* and *Production and manufacturing expenses* P&L items from the Consolidated Statement of Income in the 2019 Shell Annual Report.
- **Operating Expenses:** Calculated as the sum of *Selling, distribution, and administrative expenses*, *Research and development*, *Exploration*, *Depreciation, depletion, and amortiation*, and *Interest expense* P&L items from the Consolidated Statement of Income in the 2019 Shell Annual Report.
- **Headcount cost:** Cost of new employees. Average annual cost: Data Scientist 60,000 USD, Business Analyst 45,000 USD, Data Engineer 60,000 USD. The number of new hires per year:
  - 2021: 10 Data Scientists.
  - 2022: 10 Business Analysts.
  - 2023: 10 Data Engineers.
  - 2024: 20 Data Scientists, 10 Business Analysts, and 5 Data Engineers.
- **Consultancy Cost:** Cost associated with the support of an external partner, i.e., a consulting firm. Assuming an Average Daily Rate per consultant of 2,500 USD/day. The number of working days of the consulting team per year: 250 days. Size of consulting team per year: 2021, 30 consultants; 2022, 25 consultants; 2023, 15 consultants; 2024, 5 consultants.

- **Technology Cost:** Implementation and running costs (including licenses) of the tools required to deploy the Roadmap, i.e., Cloud Computing services, Data Visualization tools, Advanced Planning tools, etc.

Bibliography

[1] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[2] Fabio Canova and Bruce E. Hansen. Are seasonal patterns constant over time? a test for seasonal stability. *Journal of Business and Economic Statistics*, 1(3):237–252, 1995.

[3] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 08 2016.

[4] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 28(4):367–378, 2002.

[5] Jerome Friedman and Bogdan Popescu. Importance sampled learning ensembles. 10 2003.

[6] Norden Huang, Zheng Shen, Steven Long, M.L.C. Wu, Hsing Shih, Quanan Zheng, Nai-Chyuan Yen, Chi-Chao Tung, and Henry Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454:903–995, 03 1998.

[7] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software, Articles*, 27(3), 2008.

[8] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, 1992.

[9] Petro Liashchynskyi and Pavlo Liashchynskyi. Grid search, random search, genetic algorithm: A big comparison for NAS. *CoRR*, abs/1912.06059, 2019.

[10] Woloszko N. Adaptive trees: a new approach to economic forecasting. *OECD Economics Department Working Papers*, (1593), 01 2020.

[11] The Politico. Bush lifts executive ban on offshore drilling. *CBS News*.

[12] Royal Dutch Shell plc. Royal dutch shell plc, 2020 annual report. https://reports.shell.com/annual-report/2020/, January 2021.

[13] Roger Stein. Benchmarking default prediction models: Pitfalls and remedies in model validation. *Journal of Risk Model Validation*, 1:77–113, 05 2007.

[14] M. E. Torres, M. A. Colominas, G. Schlotthauer, and P. Flandrin. A complete ensemble empirical mode decomposition with adaptive noise. *Proceedings of the 36th IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1:4144–4147, 05 2011.

[15] Z. Wu and N. E. Huang. Ensemble empirical mode decomposition: a noise-assisted data analysis method. *Advances in Adaptive Data Analysis*, 1:1–41, 2009.

[16] Yang Xu, Mingzhang Luo, Tao Li, and Gangbing Song. Ecg signal denoising and baseline wander correction based on ceemdan and wavelet threshold. *Sensors*, 17(12), 11 2017.

[17] Angela Zeiler, Rupert Faltermeier, Ingo R. Keck, Maria Tomé, Carlos Garcia Puntonet, and Elmar Wolfgang Lang. Empirical mode decomposition - an introduction. *International Joint Conference on Neural Networks, IJCNN, Barcelona*, pages 1–8, 2010.

[18] Yingrui Zhou, Taiyong Li, Jiayi Shi, and Zijie Qian. A ceemdan and xgboost-based approach to forecast crude oil prices. *Complexity in Forecasting and Predictive Modelss*, 2019, 2019.