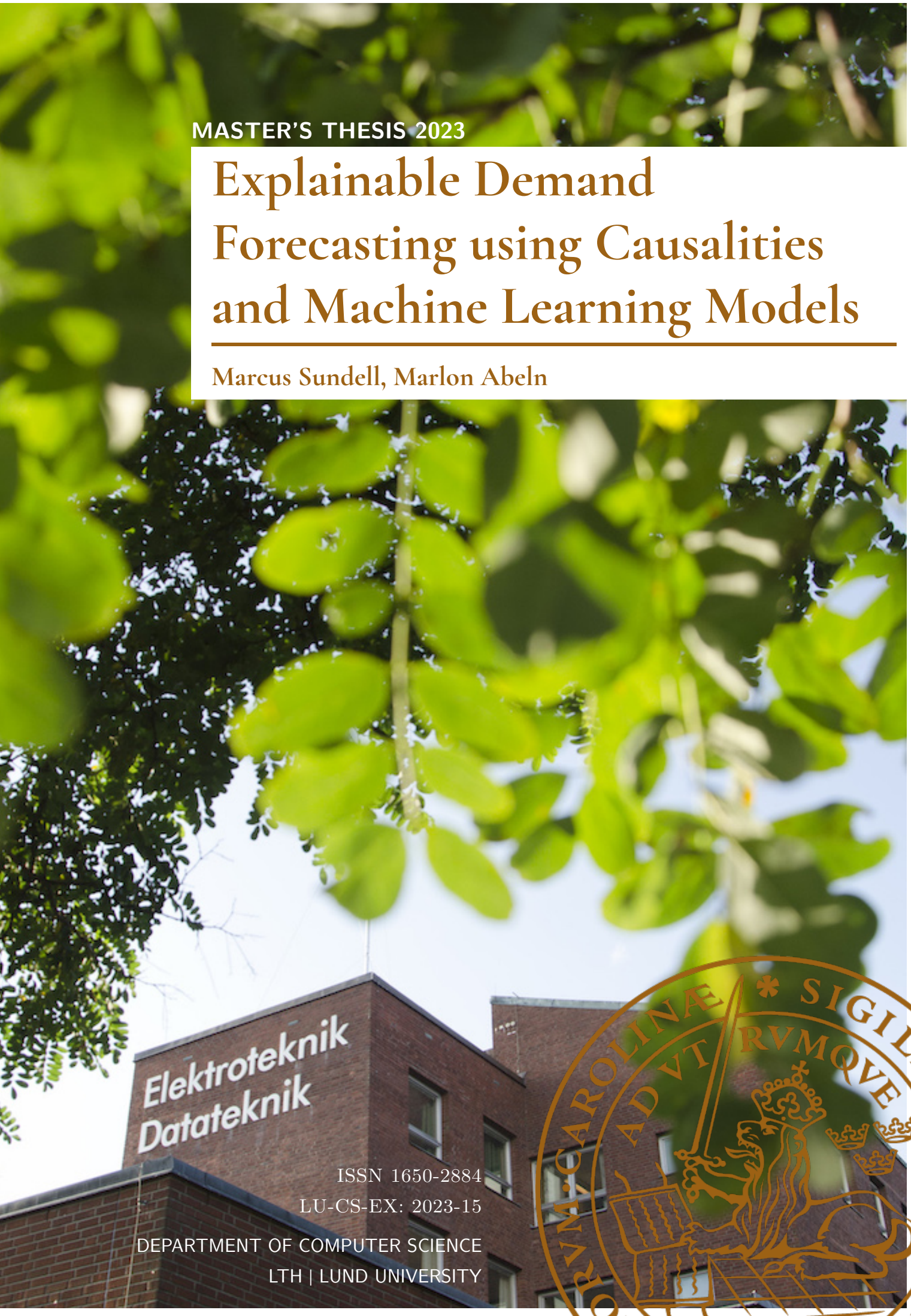


MASTER'S THESIS 2023

Explainable Demand Forecasting using Causalities and Machine Learning Models

Marcus Sundell, Marlon Abeln

ISSN 1650-2884
LU-CS-EX: 2023-15

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2023-15

Explainable Demand Forecasting using Causalities and Machine Learning Models

Användning av orsakssamband samt maskininlärning för att på ett förklarande sätt förutspå företags framtida försäljningsefterfrågan

Marcus Sundell, Marlon Abeln

Explainable Demand Forecasting using Causalities and Machine Learning Models

Marcus Sundell
ma2400su-s@student.lu.se

Marlon Abeln
ma8740ab-s@student.lu.se

June 12, 2023

Master's thesis work carried out at Microsoft Development Center Copenhagen.

Supervisors: Volker Krueger, volker.krueger@cs.lth.se
Bahram Zarrin, bahramzarrin@microsoft.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Demand forecasting is a necessary discipline to help companies predict the future demand for certain products.

This thesis aims to study the difference between traditional forecasting methods, such as ARIMA, with newer methods, such as neural network approaches, together with exploring the possibilities to combine existing forecasting results with causal signals for the different approaches.

The combination between existing forecasts and the causality signals is achieved by introducing an extra additive term to the examined decomposition models. The value of this term can be calculated with the help of libraries such as EconML. By utilizing this approach, the explainability of the models can still be retained.

The results show that, in some cases, using this approach will yield more accurate forecasts. It is also discussed why the neural network approaches did not achieve as good results as the more traditional approaches.

Keywords: Causal Demand Forecasting, Decomposition Models, Deep Learning, EconML, Machine Learning, Time-series

Acknowledgements

This project was done in cooperation with Microsoft Development Center Copenhagen for the institution of Computer Science at the Faculty of Engineering, Lund University.

We want to thank Bahram Zarrin and the employees at Microsoft for their invaluable help and feedback during the project.

We would also like to thank Volker Krueger, our supervisor at Lund University, for his help and guidance throughout this project.

Special thanks to teachers, family and friends who have supported us throughout this project by continuously giving us feedback on our progress.

Marcus Sundell "Master of Science in Engineering, Electrical Engineering"

Marlon Abeln "Master of Science in Engineering, Computer Science and Engineering"

Lund 2023

Contents

Chapter 1
Introduction

Demand forecasting is an important discipline for businesses to e.g. plan future demand of sales and/or production. This can be done in a number of ways, where one way is to use mathematical algorithms to help predict future demand. These algorithms could include statistical calculations and advanced machine learning models such as neural networks to come up with predictions using a time-series.

Rather than just focusing on one time-series of data to predict future demand, one could break up the signals into different parts and analyze the possible causalities. Understanding these underlying signals could help to generate a better forecast.

This section will introduce the reader to the background and goal of the project. It will conclude by briefly describing some important related work.

1.1 Background

This section will explain the background of the project. It will introduce- and explain terms such as *Demand Forecasting*, *Causality* and *Explainability*.

1.1.1 Demand forecasting

Demand forecasting is a discipline to predict future demand based on different inputs [1, p. 27]. Such inputs could be e.g.

- Historic sales.
- Internal signals from marketing.
- Known future sales.

- Internal demand relations.
- Demand correlations with external signals (e.g. weather).
- Qualitative estimates (from e.g. looking at results based on similar events).

The signals can be organized into different dimensions to separate or collect them in different manners. The dimensions could e.g. be:

- A subset of products for a brand (Brand-Product).
- A subset of sales for a country or region (Global-Country-Region).
- A set of sub-products, such as the number of mystery novels sold (Group-Product).

Example: A car company might have a need to predict some demand for diesel cars in a country or city. They would need this information to determine how many cars they should produce or deliver to dealers.

Statistical forecasting approaches use historical sales to decompose time-series data into *components* such as trends, seasonal periods and noise. The noise could be some effect not captured by the model, such as influence from other signals.

Modern machine learning-based forecasting approaches evolved from classic statistical concepts. These machine learning approaches are then trained on historic sales data and other signals to automatically reflect signal correlations and causalities to predict future demand [2, p. 11].

1.1.2 Causality

Causal forecasting aims to predict or forecast future demand based on variables such as e.g. different signals (price, sales) that are likely to influence future demand.

Potentially, a better forecast can be generated if a causal relationship can be identified between an action (e.g. a sale) and the demand.

It is important to note that causality is not the same as correlation. A possible observation is that if some signal X increases, another signal Y might also increase. Therefore it might be interesting to look at X when forecasting Y. Here there are two possibilities, where the first possibility is that the change in the signal X actually is causing the change in Y. The second possibility is that the change in both X and Y are instead governed by a potentially unseen signal Z. In this case, X might not contribute any information to Y and it may be said that they are conditionally independent.

$$P(X, Y|Z) = P(X|Z)\,P(Y|Z) \quad . \tag{1.1}$$

One example is that an increase in the sale of ice cream can correlate to the increase in demand for electricity. This can be misleading as ice cream tends to be sold when it is warm

outside, which also could lead to an increase in the usage of air conditioning units.

Therefore, an informed approach is to base the forecast on the outside temperature, which can be argued to govern the sale of ice cream and the use of air-conditioners.

1.1.3 Explainability

In order to understand the term *explainability*, several things could be considered. One important aspect is that a trained model can explain how the decision for a value or classification is determined. It may also mean how a forecasting system that creates a prediction on a subset of data selects its data and why.

There are two main interpretations of explainability in this project.

1. The results from a forecast should be interpretable. In other words, the user should be able to interpret how the model came to its conclusion.

2. Given that a selection of different models are used, the user should be able to examine the results from all used models. This is because different models can yield vastly different results depending on the provided data.

By examining the explainability of a method, it is also possible to avoid a black-box nature. This essentially means that instead of a process being a black box that takes some input and returns some output, it is more transparent.

1.2 Problem and motivation

Traditional demand forecasting techniques require advanced algorithmic expertise and frequent manual adjustments of the specific model parameters by analytic experts.

This thesis aims to examine models such as EconML (see 2.3.1) to estimate the causal effects of different signals. These causality models should then be combined with existing forecasting models, such as Prophet and NeuralProphet, in order to study if incorporating causal effects in a forecast can increase forecasting accuracy.

1.2.1 Research questions

1. To what extent is it possible to improve the prediction of future demand using statistical models that take components such as e.g. seasonalities and trends into account?

    1.1. Which statistical models can be used?

2. To what extent is it possible to predict future demand using machine learning methods such as e.g. neural networks?

    2.1. Which machine learning methods are preferable?

3. To what extent is it possible to improve the prediction of future demand using models that take causality signals into account?

    3.1. Which methods regarding model building are preferable (statistical, neural networks, etc)?

    3.2. To what extent is the forecasting performance impacted when looking at a signal that has a large causal effect compared to a signal that has a smaller causal effect?

1.3 Related work

There exist many projects in which the goal is to improve the performance of forecasting demand. Some of these projects create more advanced algorithms based on earlier work by including e.g. seasonalities and/or neural networks. Other projects allow causal signals to be analyzed by measuring the causal inference between the original- and causal signals.

1.3.1 Prophet

Prophet is an open-source software released by Facebook [3]. It uses an additive model in order to forecast time-series data. An additive model consists of a set of components that each model an aspect of the time-series. The individual forecasts of these components are then added together in order to produce the final forecast. With this model, non-linear trends can e.g. be fit with different types of seasonalities.

Prophet aims to improve on more traditional algorithms used in forecasting. Its paper compares Prophet to traditional algorithms such as ARIMA and TBATS [4, p. 5].

The Prophet paper contains an example, where the task is to forecast the number of events on the Facebook platform between approximately 2013-2017. In this example, the authors argue why Prophet is preferable to e.g. the ARIMA, exponential smoothing and TBATS algorithms. The ARIMA forecast is argued to be prone to large trend errors when a change of trend occurs near a cutoff period. ARIMA also does not capture any seasonality. While the exponential smoothing algorithms capture a weak seasonality, they have a hard time capturing any long-term seasonality.

While Prophet might improve on more traditional algorithms in forecasting, it is still an additive decompositional algorithm similar to many of its predecessors. Prophet does e.g. not use any neural networks in its forecast predictions. It is also not possible to include different signals to consider causalities in Prophet.

1.3.2 NeuralProphet

NeuralProphet is a successor to Prophet that aims to improve on Prophet by modeling some of its components using neural networks. It is a decomposition model that breaks a time-series into a series of components [2, p. 4]. Despite the black-box nature common in neural networks, the authors state that NeuralProphet provides an explainable model accessible to

forecasting beginners [2, p. 35].

One interesting component is the holiday component, which is similar to the holiday component in the Prophet model [2, p. 5]. The aim of this component is to examine the effect that a holiday, such as Christmas, may have on the future values of a time-series.

NeuralProphet also has components to take other, external, signals into account for its forecast. These components are built using an autoregressive model called AR-Net. This is however not the same as estimating a causal effect. This means that NeuralProphet merely looks for correlations between the provided variables.

1.3.3 EconML

EconML is a Python library, used to estimate causal inference, developed and maintained by Microsoft. EconML is one of the methods that DoWhy (see Section 1.3.5) can use to analyze the causal inference between two signals [5, p. 1-2].

EconML contains nine methods to use for the estimation of causal effects and several methods to allow for interpretability, model selection and cross-validation [6]. It is supposed to be a robust library that can be used even by forecasters new to causalities.

One of the use cases that the developers argue for is its potential use for forecasting [7]. Using EconML as the core of DoWhy means that it becomes a tool to estimate the causal effect of one statistical variable on another.

EconML is an analytical tool to estimate the causal effect between statistical variables and to analyze causal relationships. It does not perform any forecasting.

1.3.4 CausalML

CausalML is a Python library, used to estimate causal inference, developed and maintained by Uber [8]. The creators of CausalML state that its suite of algorithms has several potential use cases. One of these is determining the causal impact of some intervention for a subject with some observed features.

In total, there are eight algorithms implemented in the CausalML library [9, p. 1]. This could e.g. be used to determine the impact some promotional event could have on the potential demand.

CausalML also aims to provide a way to explain the provided treatment models (see Section 2.3.1) [10]. How these works are model specific. One example is that in an estimator which uses features, the importance of each feature can be used to describe the fitted model.

CausalML is an analytical tool to estimate the causal effect between statistical variables and to analyze causal relationships. As was the case with EconML, CausalML does not perform any forecasting.

1.3.5 DoWhy

DoWhy is an open-source Python library used to estimate causal inference [5, p. 1-2]. It can make use of both EconML and CausalML to estimate causal effects. The intention of DoWhy is to be able to make robust graphical models of causal relationships and analyze the strength of these relationships.

Compared to other libraries for causal inference, DoWhy does not only focus on estimation. The model follows three steps: It provides a way to model the given problem, it provides several estimation methods used in causal inference and it tests the validity of the causal assumptions.

While using DoWhy allows a forecaster to estimate causal inference, it does not itself perform any forecasting. This means that it would need to be combined with some other method to achieve a forecast.

Chapter 2
Theory

An analyst working for a company is assigned the task to forecast the future sales of a product. The analyst decides to start by looking at historical sales. An example of historical sales data for the product can be seen in Figure 2.1.

The data presented in Figure 2.1 is part of the *Store Item Demand Forecasting Challenge*. This dataset is described in further detail in section 3.2.1.

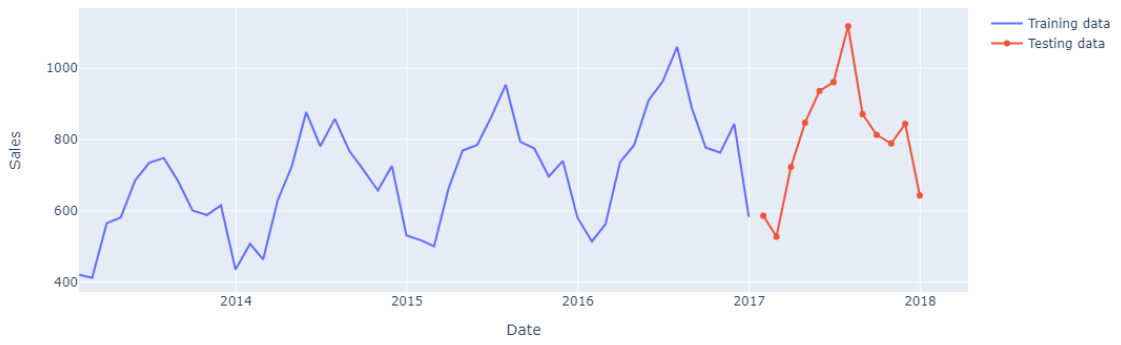


Figure 2.1: An example of historical sales data

Figure 2.1 shows temporal, monthly sampled data. That the data is temporal means that it consists of values measured at different time points. It is vital to realize that the data consists of values sampled at discrete time points. This means all time steps are in discrete time

The analyst would need to decide how far into the future they want to forecast. This is the forecast horizon (denoted h throughout the report). If e.g., the data is sampled monthly, and

the analyst would like to forecast one year into the future, the forecast horizon would be 12. It is intuitively the number of time steps (denoted t) into the future which should be forecasted.

This process is described as

$$\boldsymbol{X} = \begin{bmatrix} x_0 & x_1 & \cdots & x_{t-2} & x_{t-1} \end{bmatrix} \rightarrow \hat{\boldsymbol{X}} = \begin{bmatrix} x_t & x_{t+1} & \cdots & x_{t+h-2} & x_{t+h-1} \end{bmatrix}, \tag{2.1}$$

where $\boldsymbol{X}$ includes the sales data at the previous t time-steps and $\hat{\boldsymbol{X}}$ is the predicted sales data for the h future time steps.

Note: In order for $\boldsymbol{X}$ to contain t values, its indices will range from $0 \rightarrow t - 1$. This means that the first predicted value stored in $\hat{\boldsymbol{X}}$ will start at index t (where $x_t = \hat{x}_0$) and contain a total of h indices.

Studying Figure 2.1, it can be seen that there is a periodic- or *seasonal* effect. In the given example, the period of said season is 1 year. The period for a season is denoted m throughout the report.

In order to achieve the forecast described in Eq. (2.1), the analyst needs to determine a function F that is used to calculate the forecasting results. A graphical representation of the process is presented in Figure 2.2.

Figure 2.2: A graphical representation of forecasting

2.1 Forecasting

There are several ways the function F in Figure 2.2 can be implemented. The analyst should examine different approaches and compare their results to one another. The approaches that are studied in this thesis are presented in further detail in the coming sections.

2.1.1 AR, MA, ARMA and ARIMA

AR- and MA models are classic stationary process models [11, p. 165]. A stationary process is a stochastic process where the mean-, variance and other statistical properties do not change over time [11, p. 19].

These models can be used in time-series analysis of both stationary- and non-stationary processes e.g. in model identification and parameter estimation [11, p. 165].

Auto-regressive (AR) models

AR models use a linear combination of past values in order to estimate a future value [12, ch. 8.3]. This can be expressed as an $AR(p)$ model, where p is the order of the model. Intuitively, the order represents the number of historical values to include in the model.

$$x_t = \sum_{i=1}^{p} a_i x_{t-i} + \epsilon_t \quad . \tag{2.2}$$

In Eq. (2.2), $a_i \in \mathbb{R}$ are the AR-parameters that can be calculated using the *Yule-Walker*-equations, and ϵ_t is white noise [11, p. 167-180].

Moving average (MA) models

MA models instead use past forecast error signals in order to model the data [12, ch. 8.4]. This can be expressed as an $MA(q)$ model, where q is the order of the model.

$$x_t = \mu + \sum_{i=1}^{q} c_i \epsilon_{t-i} + \epsilon_t \quad . \tag{2.3}$$

Here, μ is the mean of the series and $c_i \in \mathbb{R}$ are the MA parameters. As for the AR model, ϵ_t is white noise.

Auto-regressive moving average (ARMA)

ARMA-models are a combination of the $MA(q)$ and $AR(p)$ models [11, p. 175]. One of the most important things for ARMA models is to be able to predict future values in a time-series [11, p. 182]. An $ARMA(p, q)$-process is described as

$$x_t = \epsilon_t + \sum_{i=1}^{p} a_i x_{t-i} + \sum_{i=1}^{q} c_i \epsilon_{t-i} \quad . \tag{2.4}$$

Note that ARMA also is a stationary model [13]. If the modeled data is not stationary, the ARMA model needs to be tweaked to achieve stationarity. This can be done by taking a series of differences.

Auto-regressive integrated moving average (ARIMA)

ARIMA models combine the MA and AR approaches in order to model the time-series and make a forecast for the time-series in the future [12, ch. 8.5].

This can be done by using *differencing*, taking the differences between consecutive observations [14]. The goal is to stabilize the mean of a time-series by taking e.g. seasonality- or trend into account [12, ch. 8.1].

This can be expressed as an $ARIMA(p, d, q)$ model, where d is the order of the differencing [15, p. 3]. A good example of how the equation can be written is presented in Eq (2.5) [14].

$$\left(1 - \sum_{i=1}^{p} a_i L^i\right)(1 - L)^d x_t = \left(1 + \sum_{i=1}^{q} c_i L^i\right)\epsilon_t \quad . \tag{2.5}$$

Here, L is a lag operator. This means that a variable at time t which is multiplied by L will yield the same variable at the previous time-step $t - 1$.

Example: $Lx_t = x_{t-1}$ and $L^2 x_t = x_{t-2}$.

A good example of the differencing using $d = 1$ and $d = 2$ is shown in Eq. (2.6) [14].

$$\begin{cases} x'_t = x_t - x_{t-1}, & d = 1 \\ x^*_t = x'_t - x'_{t-1} = (x_t - x_{t-1}) - (x_{t-1} - x_{t-2}) = x_t - 2x_{t-1} + x_{t-2}, & d = 2. \end{cases} \tag{2.6}$$

ARIMA can also be used with a modification in order to include a seasonal component. When using the seasonal version of ARIMA, a seasonal part is created that is then multiplied with the non-seasonal components [12, ch. 8.9].

Note that if no differencing is done in the ARIMA model, it will in theory be an ARMA model.

2.1.2 Triple exponential smoothing

Triple exponential smoothing, also known as Holt-Winters exponential smoothing, works by decomposing a time-series into a number of components. The components that the Holt-Winters method assumes are *Trend*, *Season*, and *Level* [12, ch. 7.3]. All these components will be explained in further detail in this section.

The way that the components can be combined in the method can either be *Additive* or *Multiplicative*. The difference between these combinations is how they are calculated and combined for forecasting.

The Additive model captures if the seasonal effects are roughly constant, e.g. do not change much throughout the series. In contrast, the multiplicative model uses a ratio of the time-series value-, trend- and level components at time-step t to express the seasonal changes. This means that this model has advantages if the seasonal changes are not constant (roughly not the same changes every season).

Which one of these should be used can be determined by testing both methods and selecting the one that models the training data better according to some metric (see section 2.2).

Regarding the data that belongs to the provided example, the model parameters α, β and

γ (which are explained in the respective component sections below) are optimized by the *Exponential smoothing*-model (see section 3). These optimized parameter values are based on maximizing the log-likelihood [16]. This could intuitively also mean that some parameters will end up being close to zero.

The likelihood is calculated using a function that returns all the different probabilities of observing some sample data when this data is extracted from a probability distribution [17]. Maximizing this probability is simply taking the parameter values that generated the highest likelihood.

Level ($L(t)$)

The Level component can be interpreted as a smoothing of the time-series at time t. Previous Level-values are weighted exponentially according to the model parameter α [12, ch. 7.3].

$$L(t) = \begin{cases} \alpha \cdot (x_t - S(t-m)) + (1-\alpha) \cdot (L(t-1) + T(t-1)), & \text{if additive} \\ \alpha \cdot \frac{x_t}{S(t-m)} + (1-\alpha) \cdot (L(t-1) + T(t-1)), & \text{if multiplicative.} \end{cases} \tag{2.7}$$

Note that $T(t)$ and $S(t)$ in Eq.(2.7) are the Trend and Seasonality components, which are discussed later in this section.

The parameter α shows how important historical values of the Level component are, where α has the constraint $0 \leq \alpha \leq 1$. If α is small, $(1-\alpha)$ will be close to 1 and historical Level-values will have a large impact on the current Level-value.

The Additive method at time t can be interpreted as the weighted (known) training value, without the seasonal effect at time $t-m$ weighted against the added values of the Level- and Trend components at the previous time-step $t-1$.

The Multiplicative method at time t can be interpreted as the ratio of the known training value and the seasonal value from the current time-step at the previous season. This is again weighted against the Level- and Trend components at the previous time-step $t-1$.

Older Level-values should therefore have an increasingly small impact, depending on α, as more values are iterated through.

The Level component for the example data was plotted in both the additive and multiplicative cases. This is plotted in Figure 2.3 and Figure 2.4.

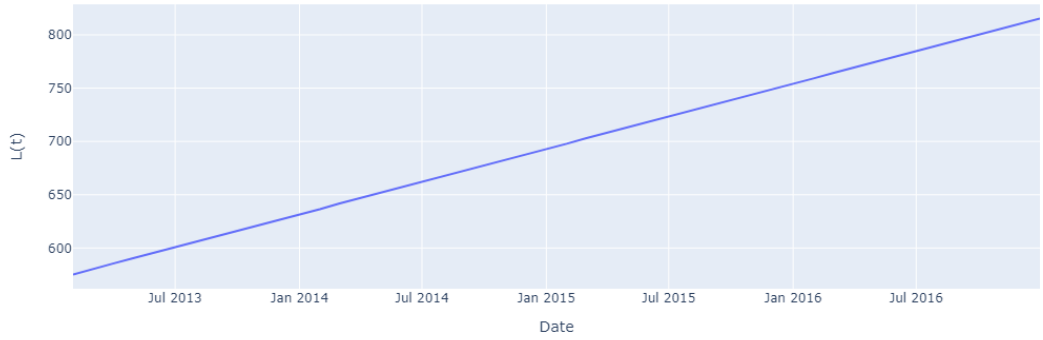


Figure 2.3: The Additive case for the level component

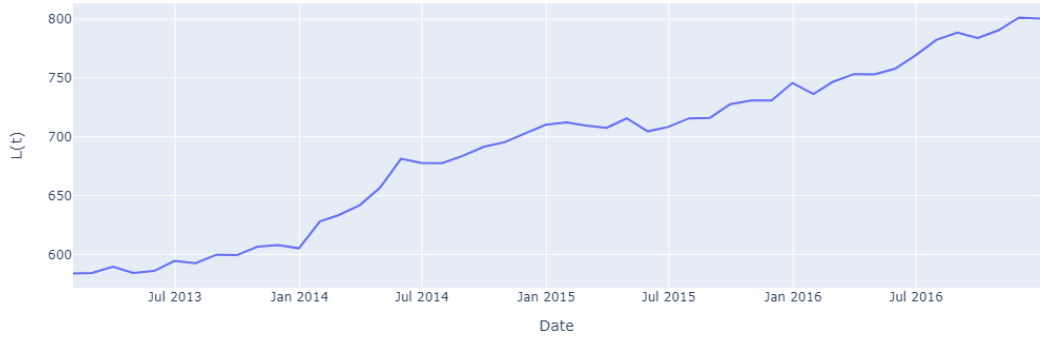


Figure 2.4: The Multiplicative case for the level component

Note that, for the provided example, α was optimized close to zero for the additive case and 0.2525 for the multiplicative case.

Trend ($T(t)$)

The Trend component aims to determine an estimate of the trend in the data at time t [12, ch. 7.2].

$$T(t) = \beta \cdot (L(t) - L(t-1)) + (1-\beta) \cdot T(t-1) \quad . \tag{2.8}$$

As both the Additive- and Multiplicative methods utilize *Holt's Linear Trend* [12, ch. 7.3], the Trend component is calculated the same in both cases.

The first part of Eq. (2.8) can be interpreted as taking the difference between the current estimated Level-value and the previously estimated Level-value. The Trend component is

then a weighted combination of this difference and the previous Trend component value.

Here, β shows how much influence historical values have on the current Trend-value. As with α, β has the constraint $0 \leq \beta \leq 1$ [12, ch. 7.2].

Note that in order to calculate $T(t)$, $L(t)$ is required to be calculated first, as it is only dependent on $T(t-1)$ and $L(t-1)$. Note also that if β is small (close to zero), the effect of the level component will diminish.

The Trend component can be extracted and plotted separately. Figure 2.5 and Figure 2.6 show the extracted Trend component from using Holt-Winters on the data presented in Figure 2.1.

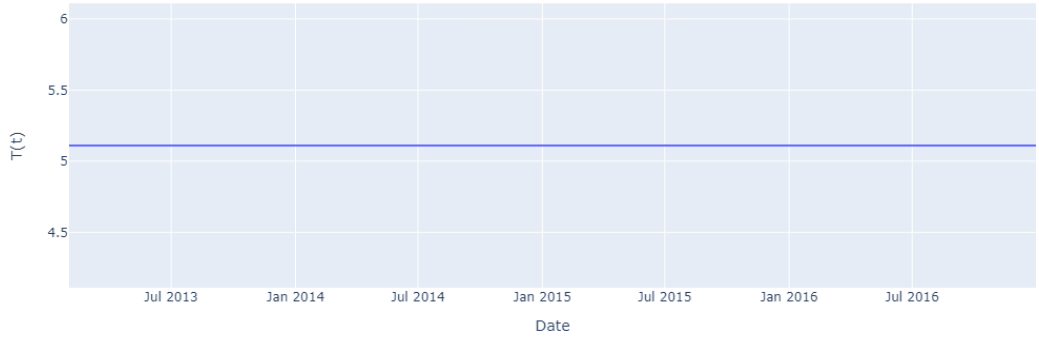


Figure 2.5: The Additive case for the trend component

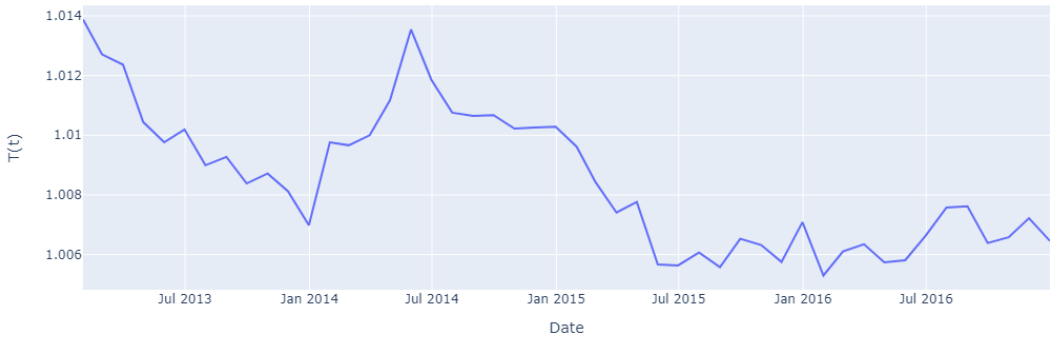


Figure 2.6: The Multiplicative case for the trend component

When looking at the original data in Figure 2.2, one may argue that there exists a small upwards trend in the data. However, studying Figures 2.5 and 2.6, this trend is not captured.

This is due to the fact that, in both cases, the β parameter is optimized close to zero, neglecting the values of the level components, yielding $T(t)$ almost constant.

Seasonality ($S(t)$)

The Seasonality component captures the seasonal pattern of the data [12, ch. 7.3].

$$S(t) = \begin{cases} \gamma \cdot (x_t - L(t-1) - T(t-1)) + (1-\gamma) \cdot S(t-m), & \text{if additive} \\ \gamma \cdot \frac{x_t}{L(t-1)+T(t-1)} + (1-\gamma) \cdot S(t-m), & \text{if multiplicative.} \end{cases} \tag{2.9}$$

The Additive method at time t can be interpreted as the weighted (known) training value, without the Level and Trend effect at time $t-1$, weighted against the added Seasonal value for the time-point in the previous season $t-m$. E.g. if it is June, the value for the previous season is the seasonal value for the previous June given a Seasonal period of 1 year.

The multiplicative method at time t is composed of two parts. The first part is a ratio of the known training value at time t and the Level and- Trend values at time $t-1$. The second part is the seasonal component at time $t-m$. This is the same logic as in the additive case. The two parts are then weighted against each other using γ.

Here, γ is a parameter that shows how important historical values of the Seasonal component are to calculate the current Seasonal component. The parameter γ has the constraint $0 \leq \gamma \leq 1-\alpha$.

The Seasonality component can be extracted and plotted separately. Figure 2.7 and Figure 2.8 show the extracted Seasonality component from using Holt-Winters on the data presented in Figure 2.1.

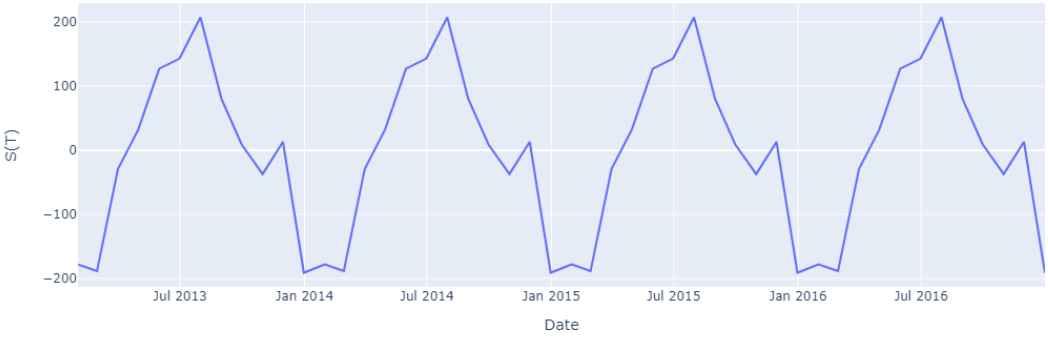


Figure 2.7: The Additive case for the seasonal component

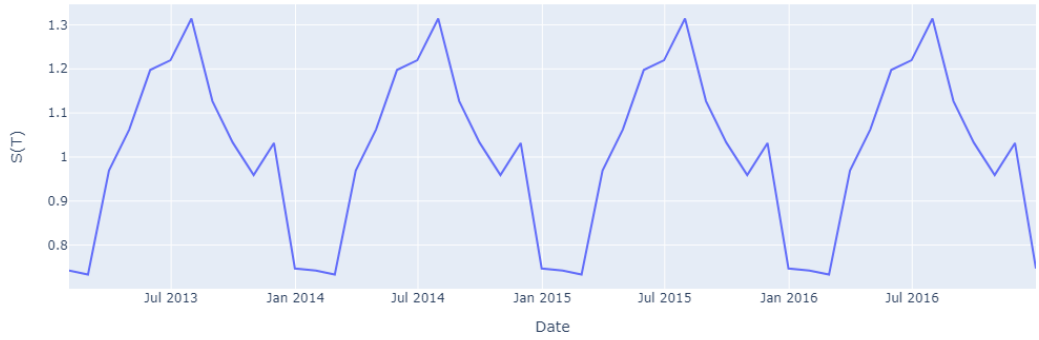


Figure 2.8: The Multiplicative case for the seasonal component

The plots in Figure 2.7 and Figure 2.8 look similar. However, they are not identical. First, the scale of the sub-figures is different. Second, it can be noticed that in the additive case (especially at the lows), the data fluctuates more than in the multiplicative case.

Note that, in this case, γ was optimized close to zero (for both cases). This means that $S(t) = S(t-m)$. In other words, there is a clear pattern that can be seen to be repeating every seasonal cycle.

Forecasting

Note: The values of α, β, and γ would have to be determined by a user or through an optimization algorithm that determines them.

The method iterates over a training dataset and calculates the given components according to whether the Additive or Multiplicative method is being used.

After all the components for the test dataset have been determined, future values can be predicted [12, ch. 7.3].

$$x_{t+h} = \begin{cases} L(t) + h \cdot T(t) + S\left(t + h - m\left(\left\lfloor \frac{h-1}{m} \right\rfloor + 1\right)\right), & \text{if additive} \\ (L(t) + h \cdot T(t))\, S\left(t + h - m\left(\left\lfloor \frac{h-1}{m} \right\rfloor + 1\right)\right), & \text{if multiplicative} \end{cases}. \tag{2.10}$$

Figure 2.9 shows the forecast from using Holt-Winters on the data presented in Figure 2.1.

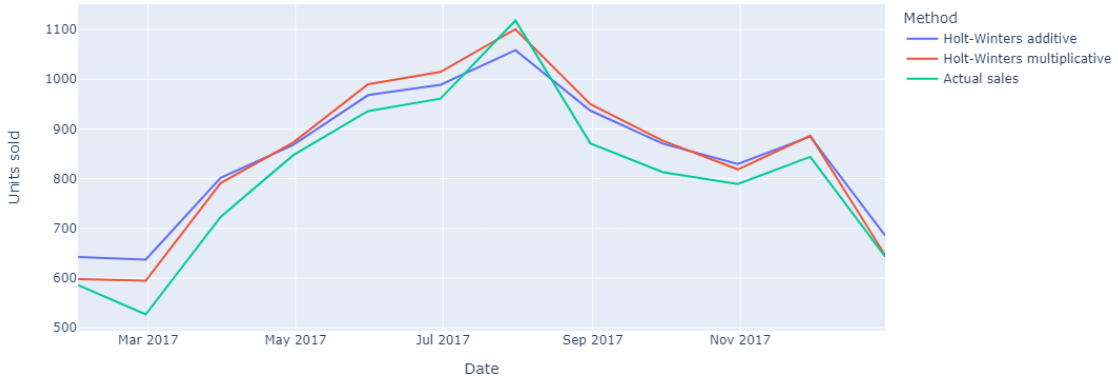


Figure 2.9: The example forecast for the additive and multiplicative cases for the example time-series

2.1.3 BATS and TBATS

There exist methods that further build on the principle of exponential smoothing. Two such methods are BATS and TBATS [18].

BATS (Box-Cox, ARMA Errors, T Seasons)

By assuming that the time-series is strictly positive, a *Box-Cox* transformation can be performed [18, p. 1516].

$$x_t^{(\omega)} = \begin{cases} \frac{x_t^{\omega}-1}{\omega}, & \text{if } \omega \neq 0 \\ \log x_t, & \omega = 0. \end{cases} \tag{2.11}$$

In Eq. (2.11), $x_t^{(\omega)}$ is the result from using the Box-Cox transformation. Here, ω is a parameter to tune the transformation.

The optimal value of ω is the value after which the transformed data is the best approximation of a normal (Gaussian) distribution. To determine ω, all possible values of ω within a certain range are evaluated [19].

The forecast for BATS is defined as

$$\hat{x}_t^{(\omega)} = L(t-1) + \phi T\,(t-1) + S(t) + d_t \ , \tag{2.12}$$

where $L(t-1)$ is the level component at time $t-1$ and ϕ is a dampening parameter that reduces the magnitude and therefore the impact of the previous short-term trend $T(t-1)$ [18, p. 1516].

The error of the model, d_t, is modeled as an ARMA process (hence the name ARMA errors). This means that the term d_t in Eq. (2.12) is modeled the same way as x_t in Eq. (2.4).

Instead of only using a single seasonal pattern, as is done in Holt-Winters, it is possible to use N seasonal patterns. Therefore $S(t)$ in Eq. (2.12) is defined as

$$S(t) = \sum_{i=1}^{N} S^{(i)}(t - m_i) \quad . \tag{2.13}$$

where $S^{(i)}(t - m_i)$ is the current seasonal component for the i:th seasonal pattern and m_i is the period of this seasonal pattern.

The **Level Component ($L(t)$)** is calculated as

$$L(t) = L(t-1) + \phi T(t-1) + \alpha d_t \quad , \tag{2.14}$$

where α is a smoothing parameter.

The **Seasonal Component ($S^{(i)}(t)$)** is given by

$$S^{(i)}(t) = S^{(i)}(t - m_i) + \gamma_i d_t \quad , \tag{2.15}$$

where γ_i is the smoothing parameter for the given seasonal pattern. This means that there are N different γ, one for each seasonal pattern.

The **Trend Component ($T(t)$)** is calculated as

$$T(t) = (1 - \phi)\, b + \phi T(t-1) + \beta d_t \quad , \tag{2.16}$$

where b is the long-term trend and β is a smoothing parameter as in the Holt-Winters method.

TBATS (Trigonometric Seasonal BATS)

TBATS is a modified version of the BATS method. TBATS extends the BATS model by creating a trigonometric seasonal representation.

This is done by using a Fourier series $S^{(i)}(t)$ to represent the seasonal patterns. Like in the case of BATS, multiple seasonal patterns are supported.

$$S^{(i)}(t) = \sum_{j=1}^{k_i} S_j^{(i)}(t) \quad , \tag{2.17}$$

In Eq. (2.17), k_i describes the number of harmonics required for the i:th component. It can be set as $k_i = \frac{m_i}{2}$ for even m_i and $\frac{m_i - 1}{2}$ for odd m_i and

$$S_j^{(i)}(t) = S_j^{(i)}(t-1)\cos\lambda_j^{(i)} + S_j^{*}(t-1)\sin\lambda_j^{(i)} + \gamma_1^{(i)} d_t \quad , \tag{2.18}$$

where $\gamma_1^{(i)}$ is a smoothing parameter, $\lambda_j^{(i)} = \frac{2\pi j}{m_i}$ and

$$S_j^{*(i)}(t) = -S_j(t-1)\sin\lambda_j^{(i)} + S_j^{*(i)}(t-1)\cos\lambda_j^{(i)} + \gamma_2^{(i)} d_t \quad , \tag{2.19}$$

where $\gamma_2^{(i)}$ is is a smoothing parameter. As in the case of BATS, i refers to the i:th seasonal pattern.

2.1.4 Prophet

Prophet is a model developed by Facebook in order to handle explainable time-series data for businesses [4, p. 5]. The idea is to decompose the data into three components: Seasonality, trend and holidays.

The Model

The components in Prophet are combined in order to calculate the estimate at time t:

$$x_t = T(t) + S(t) + E(t) + \epsilon_t \quad , \tag{2.20}$$

where $T(t)$, $S(t)$ and $E(t)$ is the trend-, seasonality- and holiday components and ϵ_t is the error term at time t [4, p. 7]. The following sections will explain these components in further detail.

Note that some of the components in Eq. (2.20) share the same name and functionality as in previously mentioned methods. However, they need to be redefined to suit Prophet as a model.

The Trend Component ($T(t)$)

The Trend component in Prophet is calculated using two different methods [4, p. 8]. The first trend model is a saturating growth model, which is an extended version of the logistic growth model

$$T(t) = \frac{C_P}{1 + e^{-k(t-o)}} \quad , \tag{2.21}$$

where C_P is the carrying capacity, k is the growth rate and o is the time-value of the functions midpoint (offset).

C_P is the maximum growth that an environment can support. In the case of demand, this can be a maximum limit of sales that can be supported. k is the growth rate at which the demand can be estimated to grow.

Fitting Prophet on the example in Figure 2.1 with the logistic trend model using $C_p = 500$, the Trend is extracted and shown in Figure 2.10.

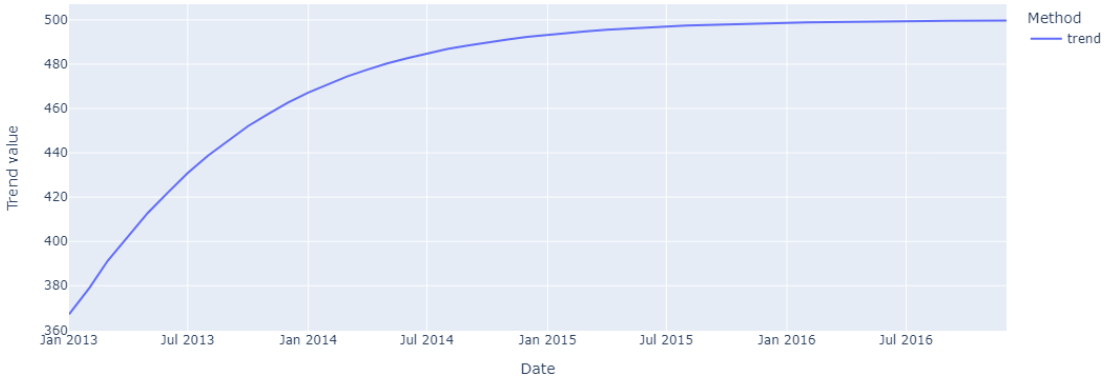


Figure 2.10: The Prophet model fitted to the example data using a logistic model with a constant carrying capacity of 500

The problem that the authors of Prophet identify with the approach is that the carrying capacity can vary [4, p. 9].

The growth rate might also not be constant, therefore, this rate would need to vary. This is done by specifying *changepoints*. These are points in time where the growth rate changes.

An intuitive way to think of this is that the rate is adjusted with a value δ_i at a point ψ_i. The set of all adjustments can be written as $\boldsymbol{\delta} = \begin{bmatrix}\delta_1 & \cdots & \delta_j\end{bmatrix}$ for j changepoints. Each rate change belongs to a time point at which the change of rate happens. This set is refered to as $\boldsymbol{\Psi} = \begin{bmatrix}\psi_1 & \psi_2 & \cdots & \psi_j\end{bmatrix}$ for the j changepoints. This means that δ_i is the rate-change that happens at timepoint ψ_i.

This is used in the calculation by defining the vector $\boldsymbol{a(t)}$ where each element $a_j(t)$ is given by

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq \psi_j \\ 0, & else \end{cases}. \tag{2.22}$$

$a_j(t)$ gives the change to the trend at each changepoint j. This means that the current rate at changepoint ψ_t, given the initial rate k, will be $k + \boldsymbol{a}(t)^T\boldsymbol{\delta}$.

The problem with using the above approach is that when the rate is adjusted, the offset o will also need to be adjusted. This is done by computing the adjustment, γ_j, at each changepoint.

$$\gamma_j = \left(\psi_t - o - \sum_{l<j}\gamma_l\right)\left(1 - \frac{k + \sum_{l<j}\delta_l}{k + \sum_{l\leq j}\delta_l}\right). \tag{2.23}$$

Using Eq. (2.22) and Eq. (2.23) in Eq. (2.21), the growth model can be re-written as

$$T(t) = \frac{C_P(t)}{1 + e^{-\left(k+\boldsymbol{a}(t)^T\boldsymbol{\delta}\right)\left(t-\left(o+\boldsymbol{a}(t)^T\boldsymbol{\gamma}\right)\right)}}. \tag{2.24}$$

The second trend model which is used if no saturating growth is present is the linear trend model. This model assumes that the trend is a straight line. In this case, the slope is the same calculation as in the saturating growth model with changepoints. The offset is still calculated according to $o + \boldsymbol{a}(t)^T \boldsymbol{\gamma}$, however $\boldsymbol{\gamma}$ is calculated differently. In this case, $\boldsymbol{\gamma}$ is instead calculated as $\gamma_j = -\psi_j \delta_j$ [4, p. 10].

The calculation of $\boldsymbol{T}(t)$ using the linear trend model is given by

$$\boldsymbol{T}(t) = \left(k + \boldsymbol{a}(t)^T \boldsymbol{\delta}\right) t + \left(o + \boldsymbol{a}(t)^T \boldsymbol{\gamma}\right) \tag{2.25}$$

When fitting Prophet on the example in Figure 2.1, the Trend in Figure 2.11 is extracted.

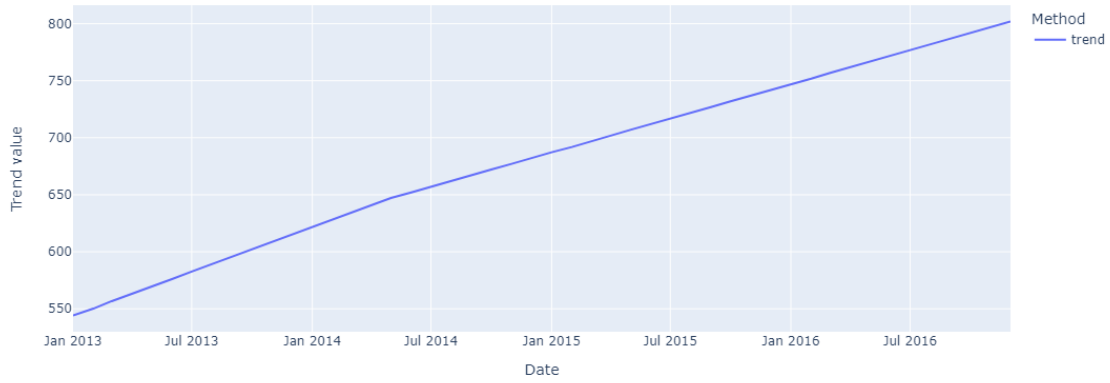


Figure 2.11: The trend using Prophet with the example dataset

In Figure 2.11 one can see an upward trend in the data. A change in the trend can also be observed in May 2014. This is an example of a changepoint.

The Seasonal component ($S(t)$)

The seasonal periods in the Prophet model are determined using a Fourier series in order to create flexible models of periodic behavior the time-series could exhibit.

$$S(t) = \sum_{n=1}^{N} \left(a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \right) , \tag{2.26}$$

where $\boldsymbol{S}(t)$ is the seasonality component and $\boldsymbol{P}$ is the time period [4, p. 11].

Example: If the unit of the time period is in days, then $P = 7$ would indicate that the data has a weekly seasonality. Similarly, if the unit of the time period is in weeks, then $P = 52$ would refer to a yearly seasonal pattern.

The expression of $\boldsymbol{S}(t)$ can be simplified by using vectors $\boldsymbol{\beta}$ and $\boldsymbol{\zeta(t)}$, where $\boldsymbol{\beta}$ is a vector that stores the Fourier coefficients $(a,\ b)$ and $\boldsymbol{\zeta(t)}$ stores the cosine and sine expressions.

$$\boldsymbol{\beta} = \begin{bmatrix} a_1 & b_1 & a_2 & b_2 & \cdots & a_N & b_N \end{bmatrix} \tag{2.27}$$

$$\boldsymbol{\zeta(t)} = \begin{bmatrix} \cos\left(\frac{2\pi \cdot 1t}{P}\right) & \sin\left(\frac{2\pi \cdot 1t}{P}\right) & \cdots & \cos\left(\frac{2\pi \cdot Nt}{P}\right) & \sin\left(\frac{2\pi \cdot Nt}{P}\right) \end{bmatrix} \tag{2.28}$$

$s(t)$ can then be rewritten as

$$s(t) = \boldsymbol{\zeta(t)\beta} \tag{2.29}$$

where $\boldsymbol{\beta} \sim \mathcal{N}(0, \sigma^2)$ [4, p. 12].

The series is truncated at N points to filter the seasonality. Increasing N will allow fitting rapidly changing seasonal patterns (with some risk of overfitting). Taylor and Letham suggest the values $N = 3$ or $N = 10$, depending on the problem at hand.

When fitting the Prophet model on the example data in Figure 2.1, Figure 2.12 is obtained.

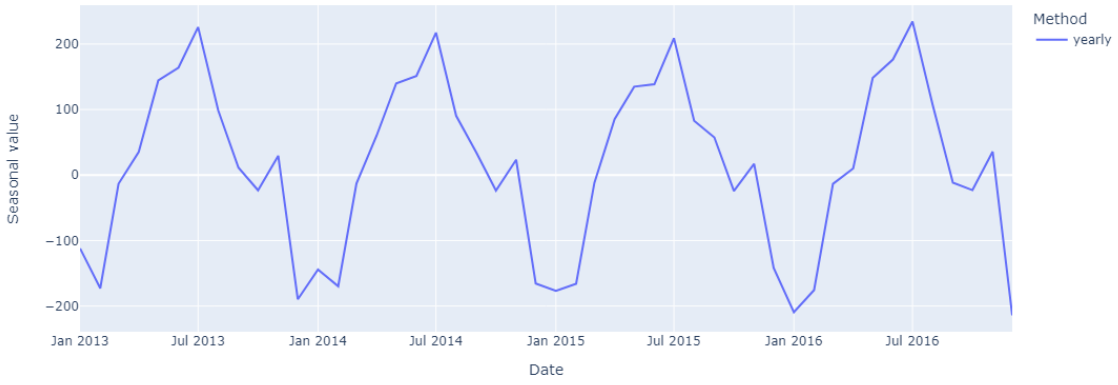


Figure 2.12: The Seasonal component from the Prophet model

Figure 2.12 shows that there is a strict seasonal cycle. This shows a peak every July and a smaller peak every November. This is a strictly repeating pattern every year. The yearly cycle is also the only one that is identified. This is because the example data is sampled monthly. This means that the sampling rate is too large to identify a weekly or daily pattern.

The Holiday component ($E(t)$)

Holidays and special events are important to account for in forecasting as these are events that are somewhat predictable. These events often occur at approximately the same time each year. Three examples are the Swedish "Midsummer", the American "Thanksgiving" and different countries' "National holiday" [4, p. 12].

The analyst has the option to provide a list of past and future events, identified by the name of the holiday. This list can then be incorporated into the model with the assumption that

the effects of the holidays are independent.

For each holiday i, there exists a set D_i that includes all past and future dates for i. The model will check whether the time t coincides with i. Each holiday is assigned a parameter $\kappa = \left[\kappa_1, ..., \kappa_L\right]$, which represents the corresponding change in the forecast. The holiday component can then be calculated by generating a matrix of regressors (variables that are used to predict the response)

$$Z(t) = \left[\mathbf{1}(t \in D_1), \cdots, \mathbf{1}(t \in D_L)\right] \ , \tag{2.30}$$

where $Z(t)$ is the matrix of regressors, L represents the number of holidays and $\mathbf{1}(t \in D_i)$ is a vector that contains 1 only on the indices that correspond to the days which are part of holiday i.

The holiday component $E(t)$ is now calculated by

$$E(t) = Z(t)\kappa \tag{2.31}$$

where $\kappa \sim \mathcal{N}(0, \sigma^2)$ is an array with the corresponding changes in the forecast for all included holidays [4, p. 13].

Instead of a single event, holidays often stretch over more than one day. These extra days are accounted for in the same way as an event, such that each day is treated as a separate holiday.

The Error term (ϵ_t)

Lastly, Prophet contains an error term to model the behavior of the time-series that are not accommodated by the model. This is assumed to be normally distributed [4, p. 7].

2.1.5 NeuralProphet

NeuralProphet was designed as a successor to Prophet. The authors describe NeuralProphet as a "hybrid forecasting framework based on PyTorch and trained with standard deep learning methods" [2, p. 1]. In other words, NeuralProphet extends Prophet by introducing deep learning methods such as neural networks for forecasting.

The Model

NeuralProphet is an additive model that combines its components together to produce a forecast. For a forecast with a forecast horizon, $h = 1$, the output can be written as

$$x_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t) \ , \tag{2.32}$$

where $T(t)$, $S(t)$, $E(t)$, $F(t)$, $A(t)$ and $L(t)$ is explained in further detail in this section [2, p. 4].

Trend ($T(t)$)

As was the case for Prophet, NeuralProphet uses a linear trend model. This is done by combining an offset o and a growth rate k. To calculate the trend effect at time t_1, the growth rate is multiplied by the difference in time since the starting point t_0. After the multiplication is done, the offset o is added [2, p. 5].

$$T(t_1) = k(t_1 - t_0) + o \quad . \tag{2.33}$$

NeuralProphet allows k to change value. This in turn means that $T(t)$ is modeled as a continuous piece-wise linear series.

$T(t)$ can be generalized by defining a growth rate $\delta_{NP}(t)$ and an offset $\rho(t)$ that are both time-dependent

$$T(t) = \delta_{NP}(t) \cdot t + \rho(t) \quad . \tag{2.34}$$

With these time-dependent parameters, NeuralProphet calculates the trend in the same way as the linear trend with changepoints as Eq. (2.25) in section 2.1.4, where $k = \delta_{NP}(0)$ and $o = \rho(0)$ [2, p. 6].

After running NeuralProphet on the example data in Figure 2.1, the trend in Figure 2.13 is obtained.

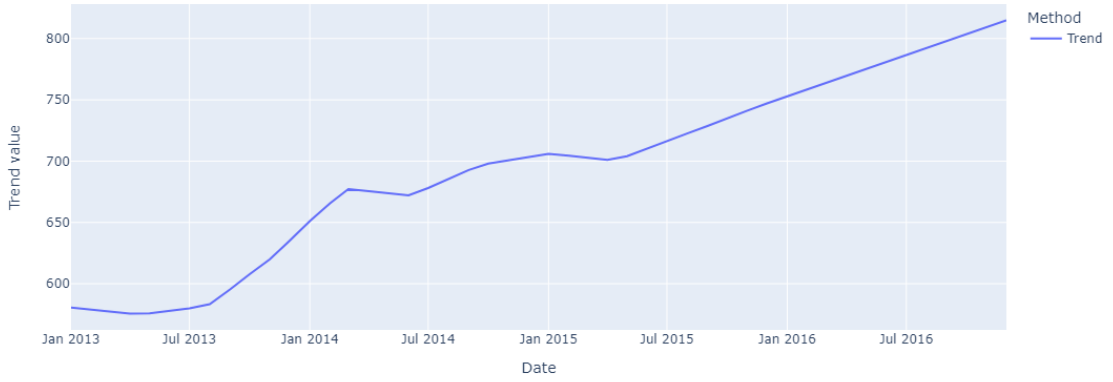


Figure 2.13: The trend from NeuralProphet

Seasonal component($S(t)$)

As with Prophet, NeuralProphet uses a Fourier series in order to model the seasonal effect (see Eq. (2.26)) [2, p. 7].

As both additive and multiplicative seasonal patterns are supported in NeuralProphet, the seasonal component will differ based on what pattern is used.

The seasonal component can be defined as

$$S(t) = \sum_{p \in \mathbb{P}} S_p^*(t) \ , \tag{2.35}$$

where $\mathbb{P}$ is a set of all the periodicities and

$$S_p^*(t) = \begin{cases} T(t)S_p(t), & \text{If multiplicative} \\ S_p(t), & \text{else.} \end{cases} \tag{2.36}$$

where $S_p(t)$ is the Fourier series in Eq. (2.26) and $T(t)$ is the trend component.

This means that the seasonal component is the sum of all different modeled seasonal periods that may be scaled using the Trend component if the multiplicative component is chosen.

After running NeuralProphet on the example data in Figure 2.1, the seasonal component in Figure 2.14 is obtained.

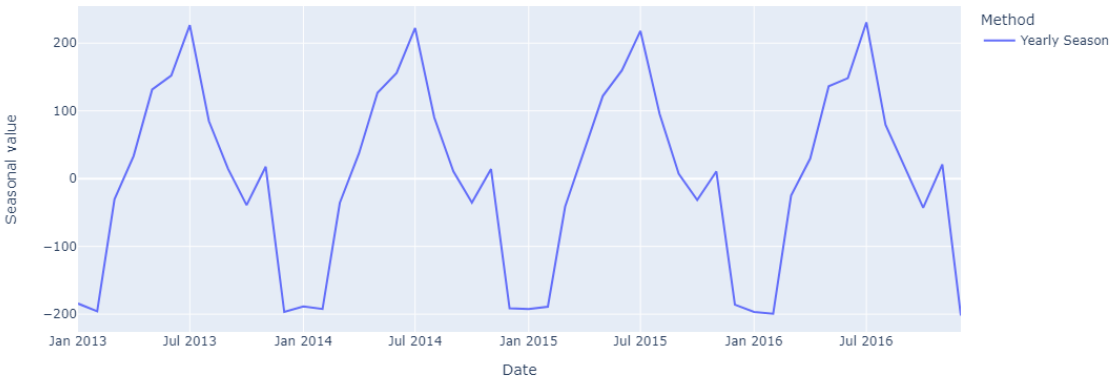


Figure 2.14: The yearly seasonal component from NeuralProphet (additive by default)

Depending on the frequency and length of the data, NeuralProphet will activate different seasonality components. The conditions for activation of a seasonality component are that the data frequency is of higher resolution than the respective periodicity together with having at least two full periods of data available [2, p. 7-8].

In the provided example, the training data is sampled monthly over four years. This means that taking the conditions into account, a yearly seasonality component can be identified.

Event and holiday components ($E(t)$)

Similar to Prophet, NeuralProphet attempts to capture the effect of holidays and different events.

This is done by defining an event e as a binary variable to indicate if the event occurs on a particular day. The event component $E(t)$ will then include the effect from all events (at time-step t) in a set $\mathbb{E} \in \mathbb{R}^{l \cdot n_e}$, where l is the length of the time-series and n_e is the number of events [2, p. 12].

As with Prophet, the days around a holiday are treated as their own separate holidays in order to determine their effect on the time-series

$$E(t) = \sum_{e \in \mathbb{E}} E_e^*(t) \ , \tag{2.37}$$

where

$$E_e^*(t) = \begin{cases} T(t) \cdot E_e(t), & \text{if multiplicative} \\ E_e(t), & \text{else} \end{cases} \tag{2.38}$$

using

$$E_e(t) = z_e e(t) \ , \tag{2.39}$$

where z_e is a coefficient related to that particular event [2, p. 13].

The effect of future regressors ($F(t)$)

A regressor is defined as a variable used for regression [12, ch. 5]. For this component, both future and past values of the variable have to be known. E.g. if the value of the price is known both during the training period and in the future (prediction period), it can be included in order to catch the effect [2, p. 12].

$F(t)$ will be the effect from all future values in the set $\mathbb{F} \in \mathbb{R}^{l \cdot n_f}$. Here, l is the length of the time-series and n_f is the number of regressors.

$$F(t) = \sum_{f \in \mathbb{F}} F_f^*(t) \ , \tag{2.40}$$

where

$$F_f^*(t) = \begin{cases} T(t) \cdot F_f(t), & \text{if multiplicative} \\ F_f(t), & \text{else,} \end{cases} \tag{2.41}$$

and d_f is the coefficient of the model for the future regressor f.

Auto-regression effects component ($A(t)$)

This component uses a modified AR model, called AR-Net, in order to calculate its effect [2, p. 8]. This method is meant to mimic a classic AR process using neural networks [20, p. 4].

A drawback of the classic AR model (see Eq. (2.2) in section 2.1.1) is that it only forecasts one step into the future (h=1). If a longer forecast horizon is desired, the model needs to be refitted for each time step in the forecast [2, p. 8].

With AR-Net, it is possible to use a forecast horizon of $h \geq 1$. The input to the AR-Net is defined as *lags* and is made up of the p last observations for the target in the AR-model ($x_{t-1}, ..., x_{t-p}$). The outputs will then be the AR-effect for each future time-step.

$$A^t(t),\ A^t(t+1),\ ..., A^t(t+h-1) = \text{AR-Net}(x_{t-1}, x_{t-2}, \cdots, x_{t-p}) \tag{2.42}$$

Studying Eq. (2.42) it can be noticed that for each forecast at time t, h predictions are obtained. Below, an example from the NeuralProphet article is shown [2, p. 9]:

Example: Assume an $AR(p = 5)$ model (see section 2.1.1) is used with forecast horizon $h = 3$. Then, at time $t = 3$, three different predicted values of the AR-effect for $\hat{x}_{t=3}$ are obtained.

$$A^{t=1}(t = 3),\ A^{t=2}(t = 3) \text{ and } A^{t=3}(t = 3)$$

The specifics of the neural network computations are out of the scope of this project, however, they can be found in the NeuralProphet article [2].

The effect of lagged regressors ($R(t)$)

Lagged regressors, or *covariates*, are used to correlate other observed variables to our target time-series [2, p. 11].

As was the case for future regressors, the future value of the lagged regressors is not known in advance (only the values up to $t-1$ are known). The idea is to determine the effect of each covariate on the forecast.

In Eq. (2.43), $R(t)$ is calculated using a set of covariates $\mathbb{X} \in \mathbb{R}^{l \cdot n_l}$. Here, l is the length of the time-series and n_l is the number of covariates x.

$$R(t) = \sum_{x \in \mathbb{X}} R_x(x_{t-1},\ x_{t-2},\ ..., x_{t-p}) \ . \tag{2.43}$$

To allow each individual covariate to attribute its effect on the prediction, a separate lagged regressor module is created for each covariate. These modules are functionally identical to the AR-Net described in Section 2.1.5. However, the lagged regressor module uses different inputs.

Instead, the lagged regressor module uses the p last observations of the covariate as inputs. The outputs are of identical form to that of $A(t)$, where each module will produce h additive components [2, p. 11-12].

$$R_x^t(t),\ R_x^t(t+1),\ ..., R_x^t(t+h-1) = \text{AR-Net}(x_{t-1}, x_{t-2}, \cdots, x_{t-p}) \tag{2.44}$$

2.1.6 Neural Network Models

In addition to the previously mentioned models, one can also use neural networks in order to make a prediction.

Recurrent Neural Networks (RNN)

A Recurrent Neural Network (RNN) is similar to a conventional neural network with the main difference being that the output from a node can be fed back into the network. This allows RNNs to be used for sequences of data.

A single hidden layer neural network is shown in Figure 2.15 [21, p. 112].

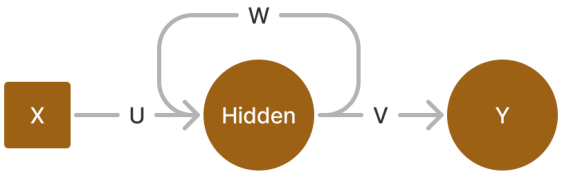


Figure 2.15: Structure of a simple RNN

The feedback in Figure 2.15 makes the output in a recurrent neural network different from a conventional neural network. The output from the specific example is

$$Y(t) = g_o\left(V \cdot \text{Hidden}\,(t)\right) \quad , \tag{2.45}$$

where g_o is the output activation function [21, p. 111].

An activation function is a non-linear function that is used in order to determine the output of the respective layer [21, p. 5].

Without the activation function, the layer is simply a linear combination of its inputs. If a linear activation function is used, the output would be the result of applying a linear function to a linear combination. This can be expressed as another linear combination. This would make the use of it superfluous. Therefore the activation function is chosen to be non-linear. A commonly used activation function is the Sigmoid function.

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad . \tag{2.46}$$

The calculation for the hidden layer is

$$\text{Hidden}\,(t) = g_h\left(U \cdot X\,(t) + W \cdot \text{Hidden}\,(t-1)\right) \quad . \tag{2.47}$$

Here, g_h is the activation function for the hidden layer [21, p. 111].

LSTM

An LSTM (Long-Short-Term-Memory) is a kind of RNN. It allows the user to input a *"window"* of time-series information in order to e.g. fit a line to the series [21, p. 115-116].

A detailed explanation of how LSTMs work is beyond the scope of this project. However, a simple network was constructed to use as a forecasting method.

When doing a forecast using the LSTM model, a window of past values is examined. The LSTM can then feed into a *dense* layer with h nodes. A dense layer is a fully connected layer in a neural network where every neuron in the layer takes every output of the previous layer as an input.

The setup in this project will result in the network having h outputs, which are the complete forecast for all values h steps into the future.

Transformer Models

Transformer models work on the concept of *Self-Attention*. This means that the model will draw from its states in order to perform its predictions [22]. In this project, the network will be predicting future values by looking at past values.

Similarly to LSTMs, a window of past values is then passed to the model in order to perform h predictions.

The architecture and mathematics of the transformer model are out of the scope of this project. However, as it can be used to forecast a time-series, it will be included in the analysis.

2.2 Metrics used to evaluate the forecasts

After having fitted- and predicted results using several models, the analyst needs to know which of the models that they have constructed is the best to use for the forecast. In order to achieve this, the analyst can compare how close the forecasts from the different models are to the expected output. To do this, the data would need to be divided into data used for fitting the model (training data) and data used for evaluating the model (testing data) as can be seen in Figure 2.1. A forecast can then be performed on the testing data using the model that was created with the training data. The results from this forecast is then compared to the known samples for the time points.

In this section, the ground truth of the prediction $\hat{x}$ will be represented by $\bar{x}$.

2.2.1 RMSE and NRMSE

The root-mean-squared error (RMSE) is a common measure used for evaluating the quality or accuracy of a prediction. Looking at the Euclidean distances, the RMSE shows how far off the predicted value is compared to the measured value [23].

The RMSE is calculated by taking the squared difference between the predicted value, $\hat{x}_i$, and the ground truth, $\bar{x}_i$, for a data point. This procedure is then done for each data point in the dataset. The mean from all data points is calculated, and the final result will be obtained by taking the square root of this mean.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\bar{x}_i - \hat{x}_i)^2} \quad . \tag{2.48}$$

A variation of Eq. (2.48) is the normalized RMSE (NRMSE). This is commonly used when comparing datasets with different scales. To calculate the NRMSE, the RMSE is simply divided by the difference in the maximum- and minimum values of the data points in the dataset.

$$\text{NRMSE} = \frac{\text{RMSE}}{x_{max} - x_{min}} \quad . \tag{2.49}$$

The idea behind normalizing the RMSE is that it puts the error of the model in relation to the range of the dataset.

Example: Let e define the error-value of a model. In the case where the mean of the ground truth is 1000 units, $e = 10$ would represent an error of 1%. However, in a case where the mean is 1 unit for the ground truth, an error $e' = 2 < e$ would represent an error of 100%.

This is significant as with a dataset of a large scale, a large error can still fall within some tolerance while this is not the case in a dataset with a smaller scale.

2.2.2 MAE and MAPE

The mean absolute percentage error (MAPE) is the average error percentage for the forecast.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left(\left(\frac{|\bar{x}_i - \hat{x}_i|}{\bar{x}_i}\right)\cdot 100\right) \quad . \tag{2.50}$$

The mean absolute error (MAE) measures the average error in the forecast.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}(|\bar{x}_i - \hat{x}_i|) \quad . \tag{2.51}$$

Note that in Eq. (2.51), the more deviations in the data the higher the error rate. This in turn would mean that the lower the deviation in the data, the closer the forecast is to the test set.

As in the case of the NRMSE, MAPE is independent of scale as the errors are represented as a percentage. However, note that if $\bar{x}_i = 0$, then Eq. (2.51) will get division by zero. The division will also mean that the loss might differ with the same error value.

Example: Let the actual value be 100 and the predicted value be 50. Then the loss would be $\frac{50}{100} \cdot 100 = 50\%$. Now, let the actual value be 10 and the predicted value is 60. Then the loss would be $\frac{50}{10} \cdot 100 = 500\%$. Note that in both cases, the error is the same.

2.2.3 Forecast Value Add

Forecast Value Add (FVA) is a means of evaluating two different forecasts, comparing them against each other using a metric score such as e.g. MAPE [24, p. 12].

This allows the analysis of what parts of a forecasting system improve the forecast and what parts that decrease it. This can e.g. compare how much a part of the forecast is improved when including causalities compared to a simple approach. Also, this can be used to decide what parts of the forecast to eliminate.

Example: Holt-Winters has been used to perform two forecasts, both with- and without taking causalities into account. An FVA is performed on the results to see which forecast is preferred, where the preferred method is the method with the smallest MAPE score.

2.3 Causalities and Estimation Methods

In order to be able to add the effect of the causal signal to a forecast, the strength of the causal relationship between the time-series has to be determined. This can be achieved by using the techniques described in this section.

2.3.1 EconML

EconML is a Python package that can be used to estimate causal responses from data with the help of machine learning. This package aims to perform the task of estimating causal inference and evaluate how strong the causal relationship is [25].

Causal Inference

The idea behind causal inference is to quantize the effect some action has on related signals. E.g. if there would be any effect on one signal based on if there was a change in some other signal. This causal relationship can be expressed using components such as *Treatments*, *Outcomes*, *Confounders* and *Features* [7].

Example: Seeing an advertisement (treatment) of a video game could potentially have an increase in the number of sold copies (outcome). The advertisement might be more effective if targeting potential customers with either certain gaming habits (confounder) or a specific

gaming system (feature) related to the advertised game.

Considering the above example, it may be considered that the treatment (causal signal) is the signal which has some effect on the outcome. Also, the features can be seen as observable characteristics of the outcome and the confounders are other signals that may have an effect on the outcome. The outcome itself is the signal that is affected by the treatment.

Using this formulation it is possible to test whether a causal relationship exists and attempt to estimate the impact. First, a model is created to give the structure of the relationship between the outcome and treatment variable (structural equation) [5, p. 4]. The hypothesis is then made that a causal relationship exists. The null hypothesis is also made such that no causal relationship exists.

Using different approaches it is then attempted to refute the null hypothesis. Some refuters could be to replace the effect with noise. Another is assuming an underlying common cause. This means that the variables do not influence each other directly but rather both changes are caused by the same unobserved cause [5, p. 5].

If there is no statistical significance between the outcome of the estimated refute and treatment, then this could mean that the null hypothesis should not be refuted and therefore any causality can not be proven.

Double Machine Learning

Double machine learning is a method that uses machine learning to estimate causal effects given that the potential confounders are known [26]. The idea behind the double machine learning approach is to perform two predictions:

1. Predicting the outcome, $\boldsymbol{Y}$, from control variables $\boldsymbol{K}$ and $\boldsymbol{W}$.
2. Predicting the treatment, $\boldsymbol{M}$, from the control variables $\boldsymbol{K}$ and $\boldsymbol{W}$.

These two predictions are then combined to create a model of the treatment effect. Additionally, different prediction models may be used for the prediction.

The *Linear double machine learning estimate* uses linear regression in order to combine predictions. In order to explain this approach, several variables need to be explained.

Let $\boldsymbol{M}$ be the chosen treatment(s) and $\boldsymbol{Y}$ be the observed outcomes. Then $\boldsymbol{W}$ is the variables (confounders) that potentially affected the choice of $\boldsymbol{M}$ together with potentially having a direct effect on $\boldsymbol{Y}$. Also, let $\boldsymbol{K}$ be a set of observable characteristics of the treated samples (features).

Let $\theta(\boldsymbol{K})$ be the constant marginal *CATE* (Conditional Average Treatment Effect) which is the "expected treatment effect in a subgroup of the population" [27, p. 2]. To estimate $\theta(\boldsymbol{K})$, the model makes the following structural equation assumptions [26]:

$$Y = \theta(K) \cdot M + g(K, W) + \epsilon \ , \tag{2.52}$$

where $g(K, W)$ is a function that shows how W and K affect the outcome variable [28, p. 2] and

$$M = f(K, W) + \eta \ , \tag{2.53}$$

$$\mathbb{E}(\eta \cdot \epsilon | K, W) = 0 \ , \tag{2.54}$$

$$\mathbb{E}(\epsilon | K, W) = 0 \ , \tag{2.55}$$

$$\mathbb{E}(\eta | K, W) = 0 \ , \tag{2.56}$$

where ϵ and η are noise random variables independent of Y, M, W and K. However, they may be correlated with each other [26] [29].

$\theta(K)$ is then estimated by rewriting Equation (2.52) [26].

$$Y - \mathbb{E}[Y|K, W] = \theta(K) \cdot (M - \mathbb{E}[M|K, W]) \ . \tag{2.57}$$

The next step is to define the conditional expectation functions

$$q(Y, W) = \mathbb{E}[Y|K, W] \ , \tag{2.58}$$

$$f(Y, W) = \mathbb{E}[M|K, W] \ . \tag{2.59}$$

Using the conditional expectation functions, the following residuals can be determined:

$$\tilde{Y} = Y - q(K, W) \ , \tag{2.60}$$

$$\tilde{M} = M - f(K, W) = \eta \ . \tag{2.61}$$

As $E[\epsilon \cdot \eta | K] = 0$, $\theta(K)$ can be estimated by regressing $\tilde{Y}$ over K and $\tilde{M}$.

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \mathbb{E}_n \left[\left(\tilde{Y} - \theta(K) \cdot \tilde{M} \right)^2 \right] \ . \tag{2.62}$$

After the model has been fitted, the performance of the CATE model can be evaluated, where EconML provides an attribute score in order to tune both this model and the chosen signals.

Chapter 3
Implementation

The implementations in this project were done using the Python programming language. Using Python, the aim was to implement both purely statistical methods- and more advanced machine learning algorithms. These models were implemented with the goal of identifying if certain causality signals would impact the demand forecasting of some products. An example of a causality signal is the price of a product.

3.1 Project execution

1. Select models and algorithms that may be used to forecast demand such as:
   (a) Statistical methods.
   (b) NeuralProphet.
   (c) Neural network architectures.
2. Apply the models to example data to get a benchmark of the data using accuracy metrics.
3. Identify techniques and algorithms for “Causality Modeling” that help to connect demand to certain signals.
4. Identify techniques for “Explainable Forecasting" models that describe the weighted contributing factors to a forecast and explain how the model has arrived at its prediction.

3.2 Datasets

For this project, several datasets were examined. These are explained in this section.

3.2.1 Kaggle

Kaggle is an online community that contains, among other things, public datasets that can be used for model evaluations and machine learning [30].

The dataset that was initially used in this project was found on Kaggle. This is the *"Store Item Demand Forecasting Challenge"* dataset.

This dataset contains sanitized data and includes three files. The relevant file, in this case, is the *"train"* CSV (comma-separated values) file [31]. This file contains the fields **"ID"**, **"date"**, **"store"**, **"item"**, and **"sales"**. The field named **"date"** includes the dates that the data was collected, **"store"** is an ID given to each store, **"item"** is an ID given to each item, and **"sales"** is the units sold on this specific date. The date data in this dataset is daily collected data.

For the implementation, the units sold were extracted for each day and the data was *"resampled"* into monthly data. This was done by taking the sum of the **"sales"** fields belonging to each month and setting the date of data collection to the last day of each respective month.

This dataset did not include a causal signal therefore it was not used for the results but rather in the theory section of the report. This was to visualize the different components of the outlined models.

Note that Figures 2.1-2.14, with the exception of Figure 2.2, have been produced from data that is included in this dataset.

3.2.2 Dominick Orange Juice

One of the datasets that were looked at in this project was an older dataset, spanning from **1989** to **1994**. This dataset contains, among other things, the store-level data and sales of different brands of orange juice [32].

This dataset was considered due to the fact that orange juice can be considered a non-necessary good and therefore might include some form of causality regarding price. This means that if the price of orange juice would increase, the demand for the product might decrease (and vice versa).

The data is sampled weekly with the week being represented with a number. This number can be converted to a week using the dataset description. The dataset then contains the price and number of sales. This means that an attempt can be made to examine the impact of the price on future sales.

For the experiments, the brand labeled *Tropicana* was examined.

3.2.3 Statistics Sweden (SCB)

SCB is responsible for coordinating the system for the official statistics in Sweden. As of December 2022, the official statistics are divided into 23 subject areas and 115 statistics areas [33].

The database *Statistikdatabasen* is open for anyone to access. Here, it is possible to create and download datasets directly from the website [34].

One of the datasets that were retrieved from SCB was statistics regarding apartment sales in Stockholm County, which contained several columns:

- The year that the data was taken.
- The number of apartments sold in Stockholm county.
- The average cost per square meter for apartments in Stockholm County.
- The median cost per square meter for apartments in Stockholm County.

The year was used as the index of the data as this would give a time point for the time-series. Secondly, data that was aimed to be forecasted is the number of apartments that were sold in that year. Lastly, the average apartment price over the year was used as the causal signal. Here it was also assumed that the price signal over the forecasting period is known.

3.2.4 Generating data

A way to generate a signal was implemented in order to prove the effect of known causality signals. For this approach, a model as in the additive case of e.g. Holt-Winters and/or Prophet was considered.

This means that the signals were generated by creating a set of additive components to generate a signal. The components are as follows.

- A long-term trend (e.g. a linear function).
- A Monthly cyclical pattern (e.g. a sine-function).
- A noise term (e.g. a Gaussian function).
- A causal signal.

Trend

The Trend was modeled linearly. This was simply done by using a one-variable linear equation with respect to time.

Cyclical seasonal patterns

By using a cyclical function such as the *sin* and *cos* functions, a repeating seasonal pattern was created.

Noise term

The idea behind a noise term is to generate an error in the measurements in order to model errors in measuring the data or to account for some random effects that might occur in the sales process.

Causal signal

The causal signal is the signal that is linearly added to the generated signal in order to add influence from another signal into the data. This is also the signal which is assumed to be known over the forecasting horizon.

Generated Dataset 1

The first generated dataset aimed to create a time-series where the causal signal has some influence on the output.

This signal is implemented using the following components:

For the ideal signal

$$\begin{cases} \text{Trend} = 1.2t + 500 \ , \\ \text{Season}_M = 50\cos\left(2\pi \frac{12}{365.25} t\right) \ , \\ \text{Causality} = AB\sin\left(\frac{2\pi}{T} t\right) \ , \end{cases} \tag{3.1}$$

and for the noisy signal

$$\begin{cases} \text{Trend} = 1.2t + \mathcal{N}(500, 5) \ , \\ \text{Season}_M = 50\cos\left(2\pi \frac{12}{365.25} t + \mathcal{N}(0, 0.25)\right) \ , \\ \text{Causality} = \mathcal{N}(1, 0.4)AB\sin\left(\frac{2\pi}{T} t\right) \ , \end{cases} \tag{3.2}$$

where

$$\begin{cases} A = 100, & \text{Amplitude of the signal} \ , \\ B = 2, & \text{Scaling factor} \ , \\ T = \frac{365.25}{4}, & \text{Period of the causal signal} \ . \end{cases} \tag{3.3}$$

Note: The causal signal was, in all cases, modeled to be periodic, due to the assumption that observable factors, such as e.g. temperature, can have a seasonal impact.

Generated Dataset 2

The second generated dataset aimed to create a time-series where the causal signal has a larger impact on the output.

This signal is implemented using the following components:

For the ideal signal

$$\begin{cases} \text{Trend} = 1.2t + 1000 \ , \\ \text{Season}_M = 50\cos\left(2\pi \frac{12}{365.25} t\right) \ , \\ \text{Causality} = AB\sin\left(\frac{2\pi}{T} t\right) \ , \end{cases} \tag{3.4}$$

and for the noisy signal

$$\begin{cases} \text{Trend} = 1.2t + \mathcal{N}(1000, 10) \ , \\ \text{Season}_M = 50\cos\left(2\pi \frac{12}{365.25} t + \mathcal{N}(0, 0.25)\right) \ , \\ \text{Causality} = \mathcal{N}(1, 0.4)AB\sin\left(\frac{2\pi}{T} t\right) \ , \end{cases} \tag{3.5}$$

where

$$\begin{cases} A = 300, & \text{Amplitude of the signal} \ , \\ B = 2, & \text{Scaling factor} \ , \\ T = \frac{365.25}{4}, & \text{Period of the causal signal} \ . \end{cases} \tag{3.6}$$

3.3 Forecasting

A forecast is performed in several steps.

1. Divide the dataset into a testing and training period. The last seasonal cycle (m samples) was used in order to achieve this.
2. Fit the forecasting model on the training period. This means that we model a mathematical equation to fit the input data as closely as possible. This will be the underlying model for future predictions.
3. Forecast over the training period. E.g. setting h to the length of the test set of the dataset. This means that both the expected and estimated values for the forecast are obtained.
4. Compare the forecast according to the previously discussed metrics and method. A flowchart describing this process is presented in Figure 3.1.

To illustrate this, a visual representation of a forecast is given in Figure 3.1.

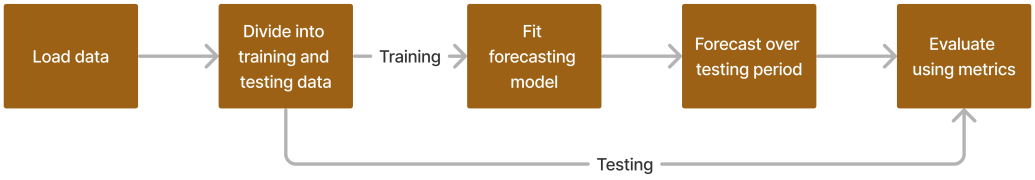


Figure 3.1: A visual representation of how the supervised learning occurs to fit and evaluate the forecasting methods

3.4 Libraries

No method used during this project was implemented from scratch as this would be too time-consuming. For each model (and many more) that was presented in section 2, there exists a Python library that can be used to fit a model to data and then predict future values.

For the **Holt-Winters** method, the Statsmodels Python library was used [35]. Statmodels is a Python library that aims to provide the functionality of different statistical models [36].

The **ARIMA** model came from the *pmdarima* library. This library uses Statsmodels to achieve the forecast but provides a different interface [37].

The **tbats**, **prophet** and **neuralprophet** libraries are stand-alone libraries that implement the BATS- and TBATS, Prophet and NeuralProphet methods that are described in Section 2 [2] [4] [38].

The **Transfomer** and **LSTM** models were implemented using **Tensorflow** and **Keras**. These libraries make it possible to create neural network models such as SimpleRNNs, LSTMs and Transformers that are described in Section 2 [39].

3.5 Implementation of a simple baseline

The simple baseline was implemented using publicly available datasets from Kaggle without analyzing any causalities. The baseline did however include looking for some form of seasonality.

The models looked at for developing this simple baseline were mostly statistical (Holt-Winters, ARIMA, BATS, TBATS and Prophet), with only some small tests including the machine learning approach (NeuralProphet, LSTMs and Transformers).

3.6 Inclusion of causalities

During this project, several different methods of including a causality were tested. The final implementation is explained in this section.

3.6.1 EconML

A double ML estimator was used as an initial model (see Section 2.3.1). Gradient boosting regressor, which is an ensemble of decision trees (combination of multiple models), was used as the estimator for both fitting the response- and the treatment to the features [40]. A polynomial of degree ten, without bias, was also used as a featurizer. This featurizer expands the dimension of the features by transforming them using the same logic as shown in Eq. (3.7) [41, Ch: 6.3.7.1].

$$\begin{pmatrix} Z_1 & Z_2 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & Z_1 & Z_2 & Z_1^2 & Z_1 Z_2 & Z_2^2 \end{pmatrix} \tag{3.7}$$

Here, Z_1 and Z_2 are features that are transformed using a featurizer.

The model was fit, using the training data, with the help of DoWhy. DoWhy is another Python package that allows further functionalities, such as refute [42]. Refute is used to validate a potential causal relationship and can be implemented in different ways, four of which are [43]:

- Generate a random confounder and add it to the data.
- Take a confounder associated with both the treatment and the outcome and add it to the data.
- Completely replace the treatment with some placebo that is randomly generated.
- Remove some random subset of the data.

The results from the listed choices will be in the form of the estimated effect, the new effect and a p-value.

Example: Looking at the placebo treatment, the hypothesis is that the signal has some causal effect on the outcome. If the p-value is small ($p < 0.05$) between the estimate of the effect and the placebo, there is no significant statistical difference between the two and the hypothesis can be refuted.

3.6.2 Combining EconML with a forecast

This approach aims to utilize the causal effect estimation present in EconML in order to augment a forecast. An overview of the approach is given in Figure 3.2.

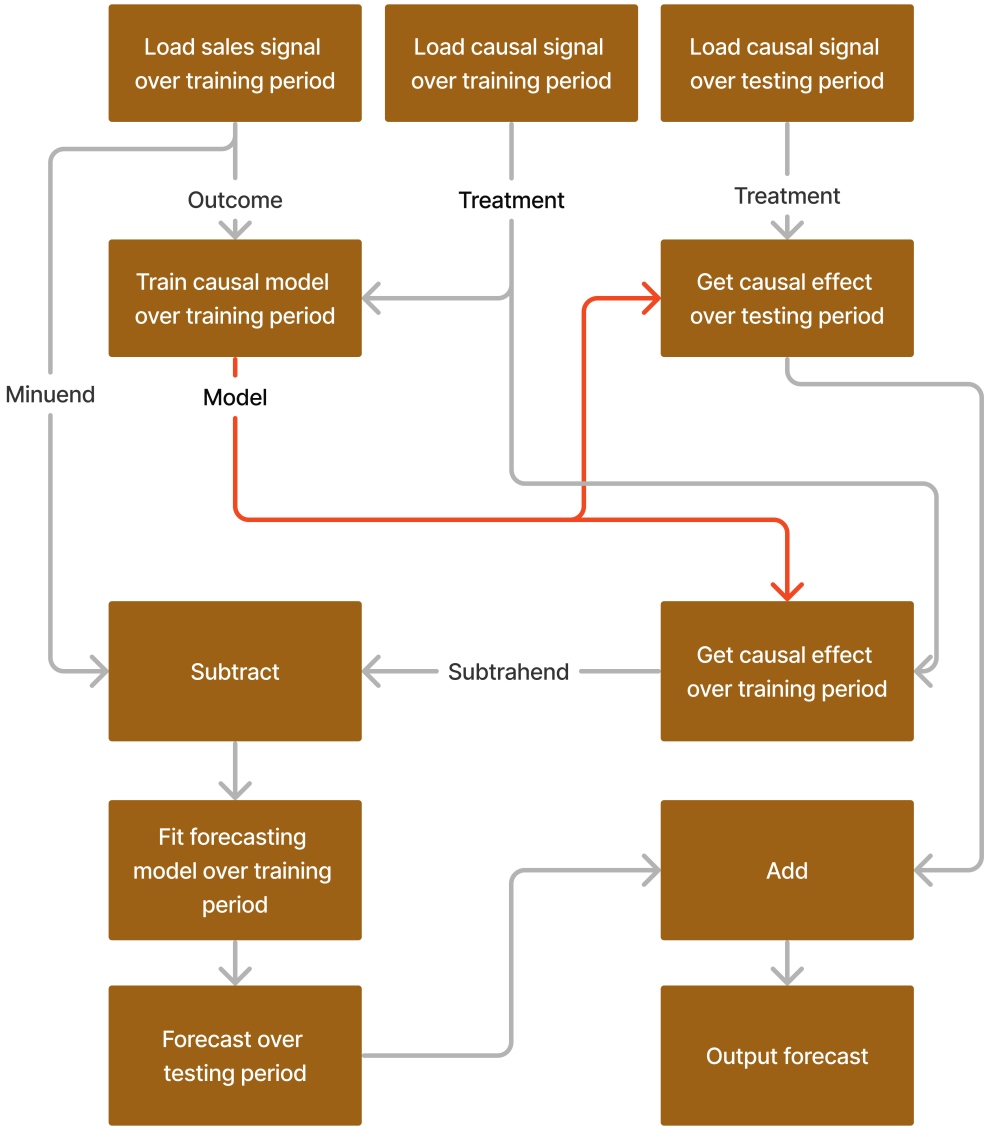


Figure 3.2: A visual representation of combining a forecast with the causal effect

The EconML *effect* method calculates the expected change in the outcome variable when the value of the treatment variable changes from $\mathbf{M_0}$ to $\mathbf{M_1}$ [44]. The cumulative effect is then part of the original time-series that is created through the causal signal.

The change may then be subtracted from the training data. This is done in order to prevent the forecasting method to attempt to fit data that is already taken into account.

A given forecasting method may then be fitted. The approach will then add the new component to the model. This can be compared to what is done in the case of additive methods such as e.g. Prophet or NeuralProphet.

This also presents a benefit of the approach as it is able to be added to any of the forecasting methods given in this project. The causal effect may then be calculated over the test set. The result of this can then be added to the forecast made with the chosen forecasting method in order to obtain the final forecast.

Chapter 4
Results

In order to obtain the results, the datasets that were used were split into two parts. One part for training- and one part for testing. The training part of the dataset is then used to fit the various models, whereas the testing part is to validate the results of the forecasts.

The results in this section will be presented with, among other things, tables that show the metric scores of each implemented method. The best method for each model will be underlined for both the baseline and causal cases.

Note: For both generated datasets, the seasonality component in both Holt-Winters methods, BATS- and TBATS are the same as the seasonality for the ideal signal (see Eq. (3.1)). Also, the second seasonality component for BATS and TBATS is the same as the seasonality for the causal signal (see Eq. (3.3)).

4.1 Generated Dataset 1

Both the generated signal and the causal signal were segmented into training- and testing data. The different segmentations can be seen in Figure 4.1 and Figure 4.2 respectively.

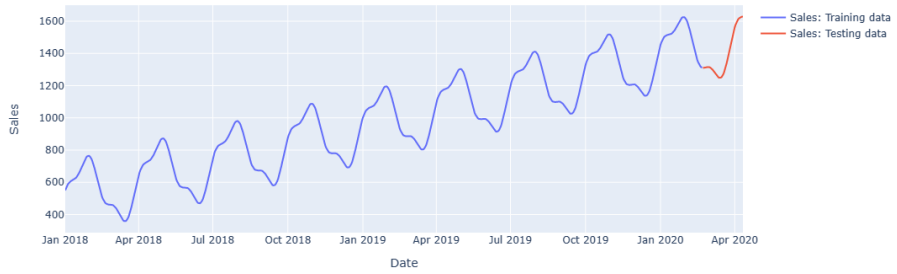


Figure 4.1: The segmentation of the data on Generated Dataset 1

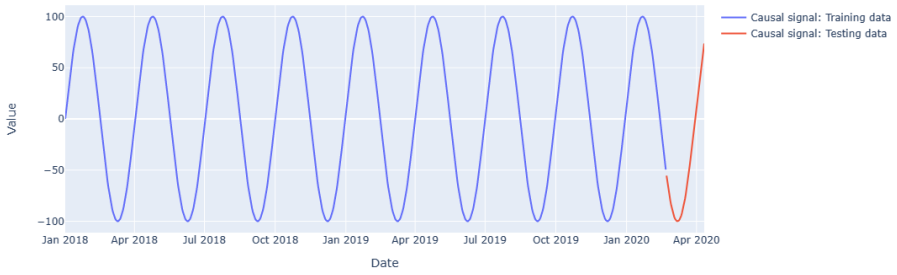
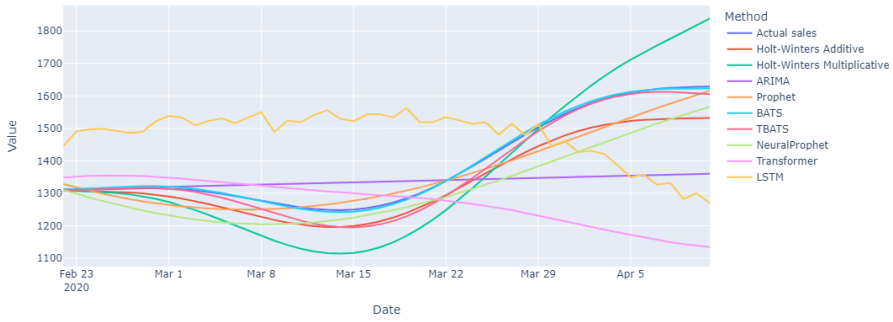
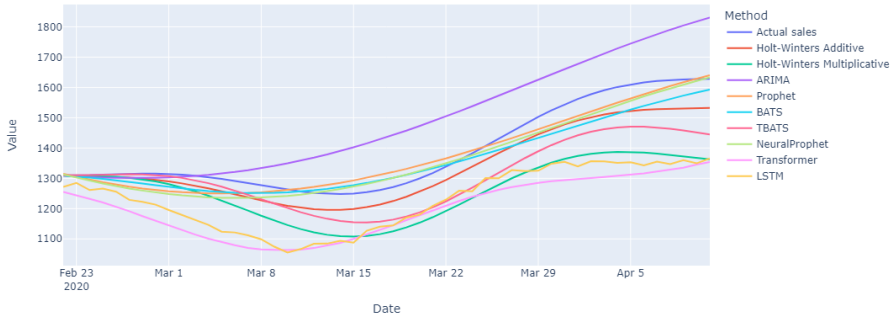


Figure 4.2: The causal signal from Generated Dataset 1

The forecasting models, discussed in Section 2, were then used to make a prediction both using the baseline approach (no causalities) and by using our proposed method (see Section 3.6.2). The forecast results, using both methods, are presented in Figure 4.3.



(a) The results of running the baseline forecasting methods on Generated Dataset 1



(b) The results of running Generated Dataset 1, taking the causality signal into account

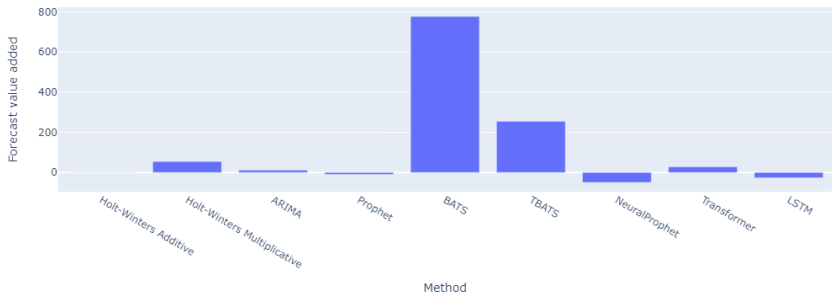
Figure 4.3: The results of running the baseline methods on Generated Dataset 1 without (a) and with (b) taking the causalities into account

A full table of the metrics evaluated on both forecasts is given in Table 4.1.

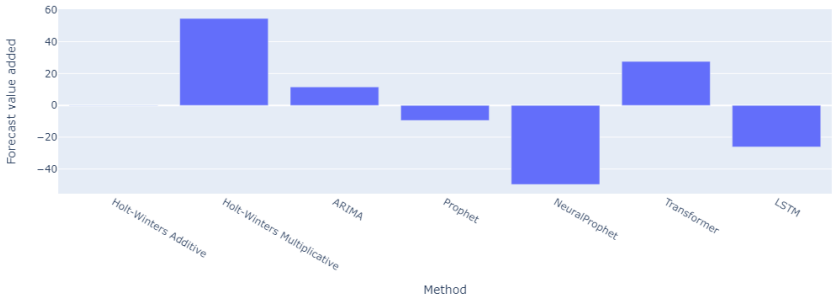
Table 4.1: A table of the metrics evaluated on the baseline forecast (T)- and with taking the causality signal into account (B) for Generated Dataset 1

Method	MAE	MAPE	RMSE	NRMSE
(T) Baseline				
Holt-Winters Additive	49.996	3.505	56.215	0.148
Holt-Winters Multiplicative	80.970	5.854	96.512	0.254
ARIMA	97.016	6.495	135.130	0.355
Prophet	38.482	2.684	46.209	0.121
BATS	4.201	0.305	4.684	0.012
TBATS	21.572	1.628	29.315	0.077
NeuralProphet	72.150	5.056	81.066	0.213
Transformer	158.758	10.449	231.090	0.607
LSTM	207.642	15.162	223.236	0.587
(B) Causal				
Holt-Winters Additive	49.924	3.500	56.132	0.148
Holt-Winters Multiplicative	129.606	9.051	149.760	0.394
ARIMA	102.181	7.244	120.869	0.318
Prophet	33.448	2.429	37.947	0.100
BATS	38.694	2.675	46.701	0.123
TBATS	82.943	5.771	99.355	0.261
NeuralProphet	35.445	2.543	41.466	0.109
Transformer	188.284	13.332	199.807	0.525
LSTM	158.314	11.200	171.563	0.451

The forecast value add for the results using the MAPE metric is given in Figure 4.4.



(a) The FVA for Generated Dataset 1



(b) The FVA for Generated Dataset 1, excluding BATS and TBATS

Figure 4.4: The FVA for the Generated 1 Dataset with (a)- and without (b) BATS and TBATS included

Note: A negative FVA means that the value of the loss function has decreased which, in this case, is a positive outcome.

From the FVA described in Figure 4.4 it can be seen that, in this case, a variation of increases and decreases in the forecasting accuracy happens for the evaluated methods.

Note: A version both with- and without BATS and TBATS is given. This is done as these methods performed so well for the baseline approach that the FVA of the other methods can hardly be seen in the plot.

In this case, these methods almost mirrored their respective ideal signal in the baseline forecast. Even if the results from the forecast taking the causality signal into account were good, it was not better than ideal.

The improvement of a forecast can never be better than 100% as this would result in a loss of 0. The accuracy decrease can however be large. This can be seen in Table 4.1. Here, the MAPE for e.g. BATS is 0.305%, which means that the value for MAPE using the proposed method (2.675%) represents a large decrease in accuracy.

4.2 Generated Dataset 1 (with added noise)

Both the generated signal and the causal signal were segmented into training- and testing data. The different segmentations can be seen in Figure 4.5 and Figure 4.6 respectively.

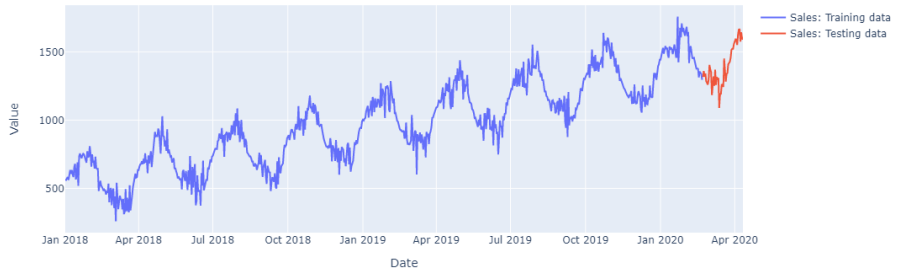


Figure 4.5: The segmentation of the data on Generated Dataset 1, with added noise

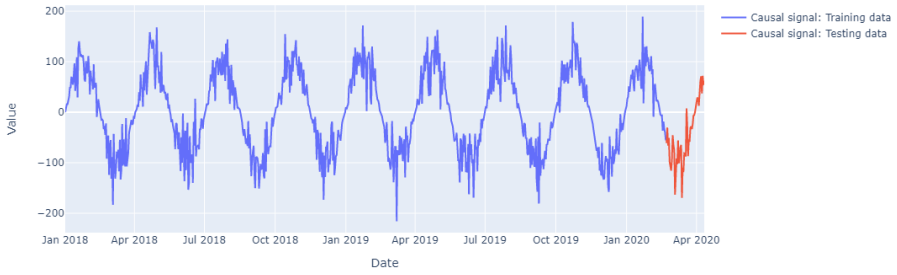
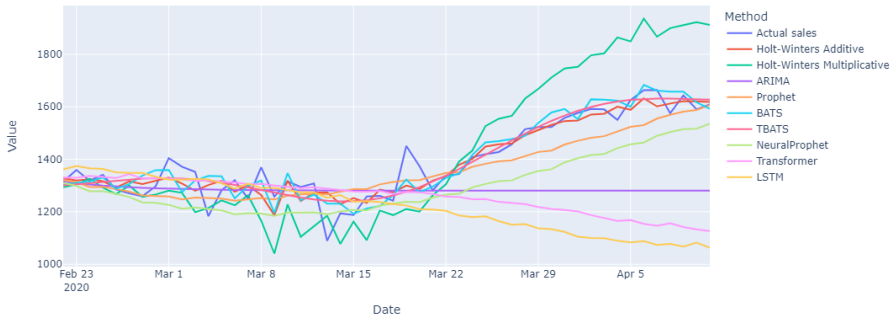
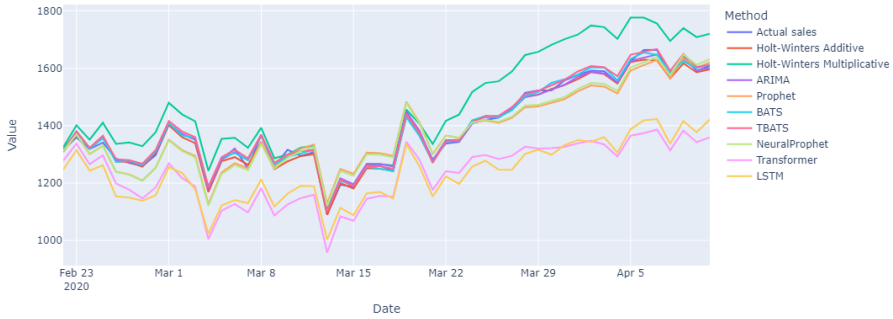


Figure 4.6: The causal signal from Generated Dataset 1, with added noise

The forecasting models, discussed in Section 2, were then used to make a prediction both using the baseline approach (no causalities) and by using our proposed method (see Section 3.6.2). The forecast results, using both methods, are presented in Figure 4.7.



(a) The results of running the baseline forecasting methods on Generated Dataset 1, with added noise



(b) The results of running Generated Dataset 1, with added noise, taking the causality signal into account

Figure 4.7: The results of running the baseline methods on Generated Dataset 1, with added noise, without (a)- and with (b) taking the causalities into account

A full table of the metrics evaluated on both forecasts is given in Table 4.2.

Table 4.2: A table of the metrics evaluated on the baseline forecast (T)- and with taking the causality signal into account (B) for Generated Dataset 1, with added noise

Method	MAE	MAPE	RMSE	NRMSE
(T) Baseline				
Holt-Winters Additive	41.472	3.106	55.139	0.096
Holt-Winters Multiplicative	129.921	8.934	160.145	0.279
ARIMA	136.574	9.156	183.431	0.319
Prophet	61.263	4.388	75.836	0.132
BATS	42.630	3.169	54.867	0.096
TBATS	38.402	2.880	52.977	0.092
NeuralProphet	100.831	7.018	116.725	0.203
Transformer	172.765	11.491	241.999	0.421
LSTM	209.184	13.899	287.971	0.501
(B) Causal				
Holt-Winters Additive	11.188	0.790	14.697	0.026
Holt-Winters Multiplicative	75.991	5.233	90.368	0.157
ARIMA	8.891	0.653	11.3751	0.020
Prophet	33.670	2.446	38.565	0.067
BATS	9.554	0.690	11.797	0.021
TBATS	9.944	0.719	12.334	0.021
NeuralProphet	31.675	2.313	36.380	0.063
Transformer	159.111	11.186	173.058	0.301
LSTM	155.878	10.967	165.676	0.288

The forecast value add for the results using the MAPE metric is given in Figure 4.8.

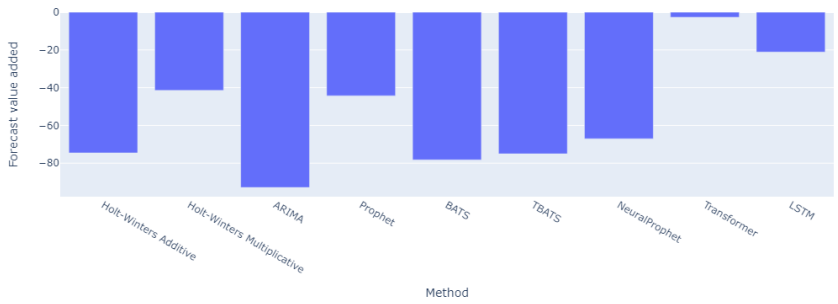


Figure 4.8: The FVA for Generated Dataset 1, with added noise

Note: A negative FVA means that the value of the loss function has decreased which, in this case, is a positive outcome.

From the FVA described in Figure 4.8 it can be seen that, in this case, an increase in the forecasting accuracy happens for all evaluated methods.

4.3 Generated Dataset 2

Both the generated signal and the causal signal were segmented into training- and testing data. The different segmentations can be seen in Figure 4.9 and Figure 4.10 respectively.

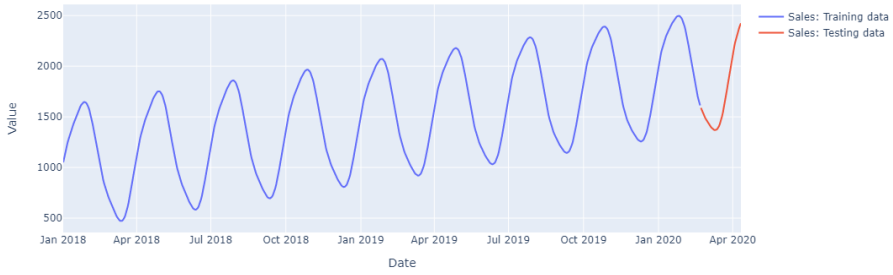


Figure 4.9: The segmentation of the data on Generated Dataset 2

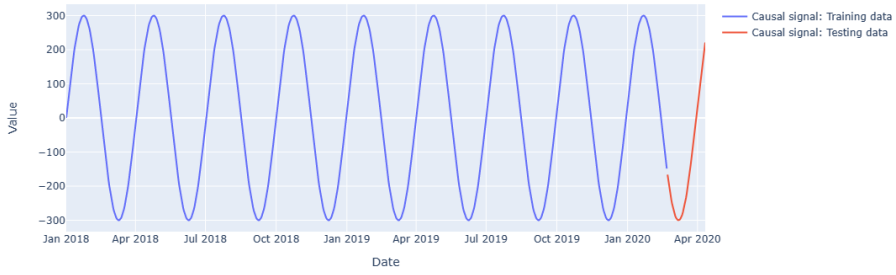
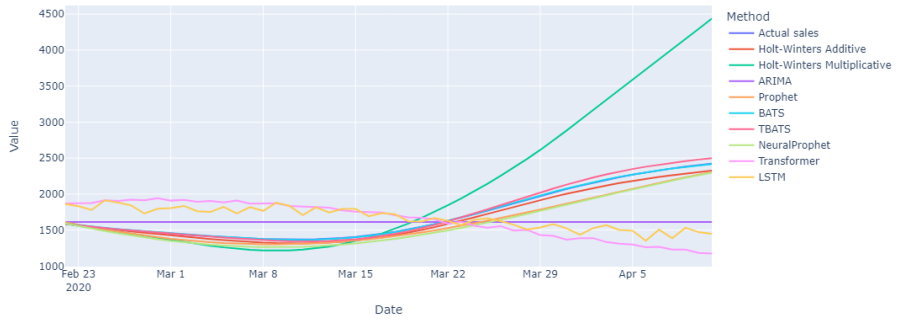
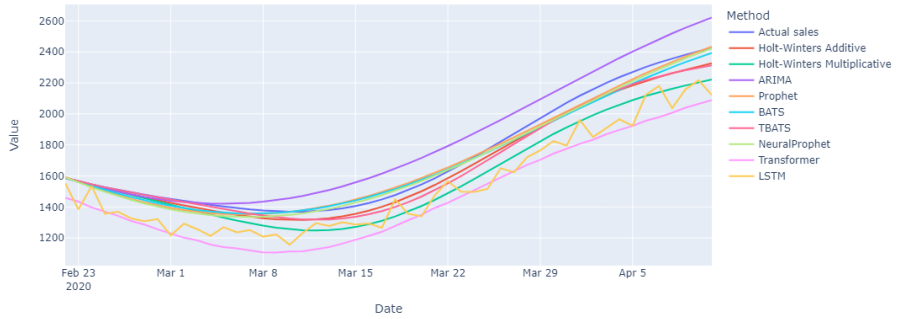


Figure 4.10: The causal signal from Generated Dataset 2

The forecasting models, discussed in Section 2, were then used to make a prediction both using the baseline approach (no causalities) and by using our proposed method (see Section 3.6.2). The forecast results, using both methods, are presented in Figure 4.11.



(a) The results of running the baseline forecasting methods on Generated Dataset 2



(b) The results of running Generated Dataset 2, taking the causality signal into account

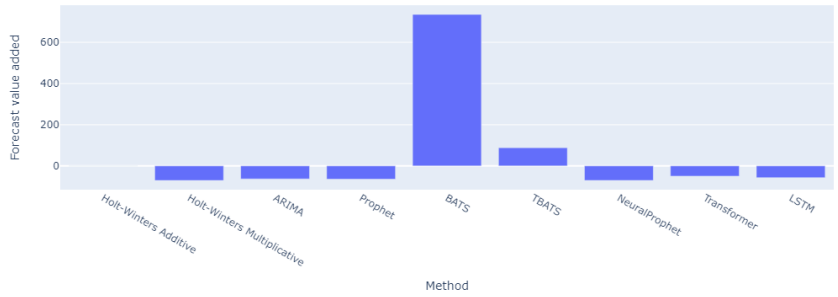
Figure 4.11: The results of running the baseline methods on Generated Dataset 2 without (a), (b)- and with (c) taking the causalities into account

A full table of the metrics evaluated on both forecasts is given in Table 4.3.

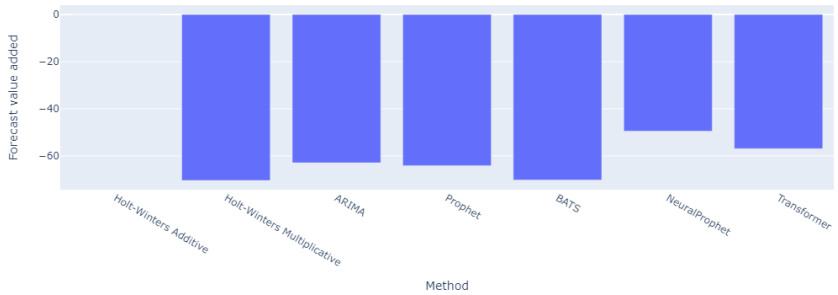
Table 4.3: A table of the metrics evaluated on the baseline forecast (T)- and with taking the causality signal into account (B) for Generated Dataset 2

Method	MAE	MAPE	RMSE	NRMSE
(T) Baseline				
Holt-Winters Additive	49.918	2.822	56.125	0.053
Holt-Winters Multiplicative	465.135	22.432	738.171	0.700
ARIMA	289.422	15.497	368.421	0.350
Prophet	103.584	5.718	119.805	0.114
BATS	4.201	0.254	4.684	0.004
TBATS	30.063	1.565	40.922	0.039
NeuralProphet	128.518	7.330	140.470	0.133
Transformer	514.258	28.784	598.043	0.567
LSTM	421.081	23.717	487.638	0.463
(B) Causal				
Holt-Winters Additive	49.929	2.823	56.139	0.053
Holt-Winters Multiplicative	116.767	6.634	131.029	0.124
ARIMA	100.520	5.745	118.861	0.113
Prophet	33.761	2.052	38.329	0.036
BATS	37.701	2.122	44.172	0.042
TBATS	52.531	2.950	62.156	0.059
NeuralProphet	36.721	2.183	43.076	0.041
Transformer	247.626	14.545	254.840	0.242
LSTM	176.857	10.214	191.386	0.182

The forecast value add for the results using the MAPE metric is given in Figure 4.12.



(a) The FVA for Generated Dataset 2



(b) The FVA for Generated Dataset 2, excluding BATS

Figure 4.12: The FVA for the Generated 2 Dataset with (a)- and without (b) BATS included

Note: A negative FVA means that the value of the loss function has decreased which, in this case, is a positive outcome.

From the FVA described in Figure 4.12 it can be seen that, in this case, a variation of increases and decreases in the forecasting accuracy happens for the evaluated methods.

Note: A version both with- and without BATS is given. This is done as BATS performed so well for the baseline approach that the FVA of the other methods can hardly be seen in the plot.

In this case, BATS almost mirrored the ideal signal in the baseline forecast. Even if the results from the forecast taking the causality signal into account were good, it was not better than ideal.

The improvement of a forecast can never be better than **100%** as this would result in a loss of **0**. The accuracy decrease can however be large. This can be seen in Table 4.3. Here, the MAPE for BATS is below **0.01%**, which means that the value for MAPE using the proposed method (**2.989%**) represents a large decrease in accuracy.

4.4 Generated Dataset 2 (with added noise)

Both the generated signal and the causal signal were segmented into training- and testing data. The different segmentations can be seen in Figure 4.13 and Figure 4.14 respectively.

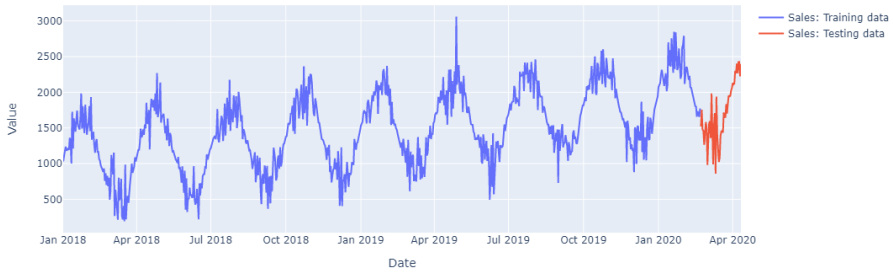


Figure 4.13: The segmentation of the data on Generated Dataset 2, with added noise

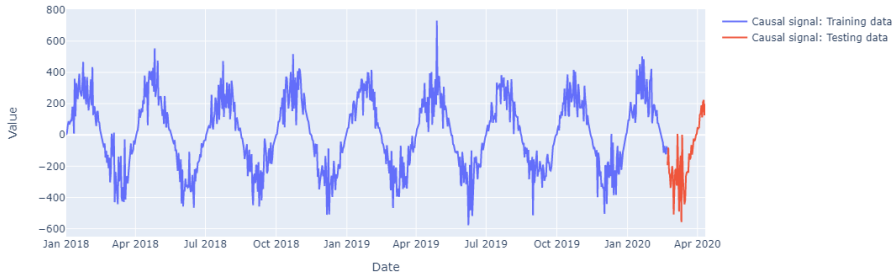
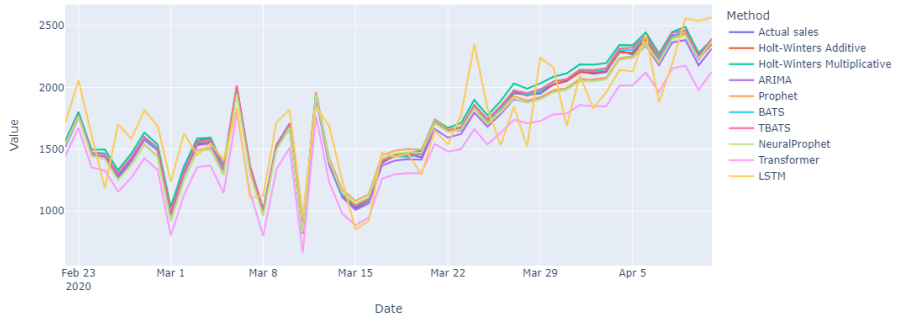


Figure 4.14: The causal signal from Generated Dataset 2, with added noise

The forecasting models, discussed in Section 2, were then used to make a prediction both using the baseline approach (no causalities) and by using our proposed method (see Section 3.6.2). The forecast results, using both methods, are presented in Figure 4.15.

(a) The results of running the baseline forecasting methods on Generated Dataset 2, with added noise



(b) The results of running Generated Dataset 2, with added noise, taking the causality signal into account

Figure 4.15: The results of running the baseline methods on Generated Dataset 2, with added noise, without (a)- and with (b) taking the causalities into account

A full table of the metrics evaluated on both forecasts is given in Table 4.4.

Table 4.4: A table of the metrics evaluated on the baseline forecast (T)- and with taking the causality signal into account (B) for Generated Dataset 2

Method	MAE	MAPE	RMSE	NRMSE
(T) Baseline				
Holt-Winters Additive	140.176	10.216	223.836	0.143
Holt-Winters Multiplicative	250.988	14.886	327.132	0.209
ARIMA	425.739	23.730	539.390	0.345
Prophet	161.684	11.335	220.004	0.141
BATS	145.414	10.518	221.791	0.142
TBATS	140.732	10.029	203.963	0.130
NeuralProphet	215.711	13.354	259.822	0.166
Transformer	535.679	31.350	653.205	0.417
LSTM	597.825	36.766	701.573	0.448
(B) Causal				
Holt-Winters Additive	15.739	0.977	19.886	0.013
Holt-Winters Multiplicative	38.209	2.272	44.852	0.029
ARIMA	34.989	2.009	40.468	0.026
Prophet	35.906	2.295	41.240	0.026
BATS	16.771	1.038	19.638	0.013
TBATS	17.929	1.090	21.733	0.014
NeuralProphet	38.375	2.391	45.617	0.029
Transformer	192.886	11.598	200.697	0.128
LSTM	186.447	11.538	218.514	0.140

Note: For the baseline approach, TBATS is deemed to yield the best results due to the fact that it has the lowest metric scores for the majority of the metrics used. In the causal approach, Holt-Winters Additive is deemed to yield the best results.

The forecast value add for the results using the MAPE metric is given in Figure 4.16.

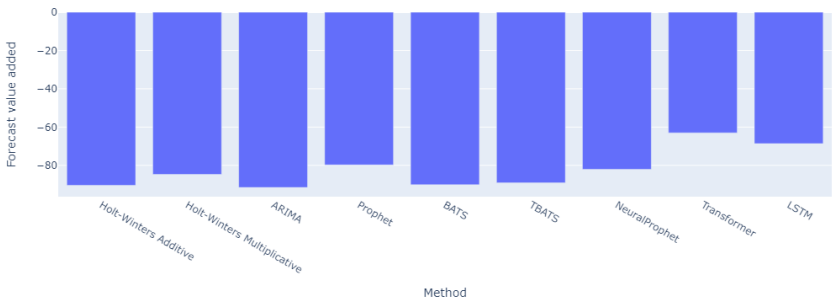


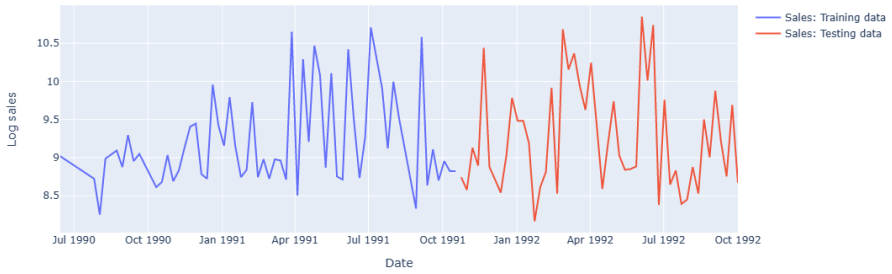
Figure 4.16: The FVA for Generated Dataset 2, with added noise

Note: A negative FVA means that the value of the loss function has decreased which, in this case, is a positive outcome.

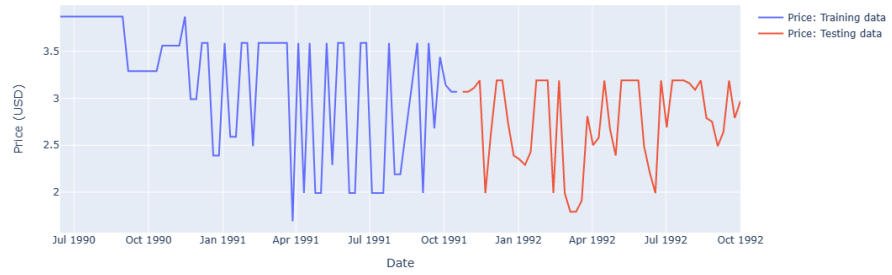
From the FVA described in Figure 4.16 it can be seen that, in this case, an increase in the forecasting accuracy happens for all evaluated methods.

4.5 Dominick Orange juice

Both the sales signal and the price signal were segmented into training- and testing data. The different segmentations can be seen in Figure 4.17.



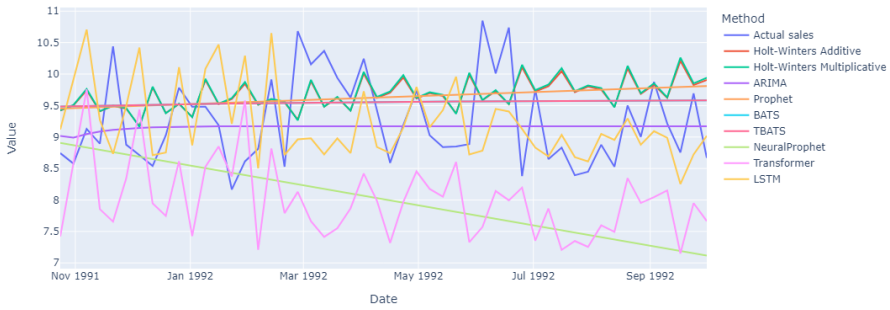
(a) The segmentation of the sales data on Orange Juice



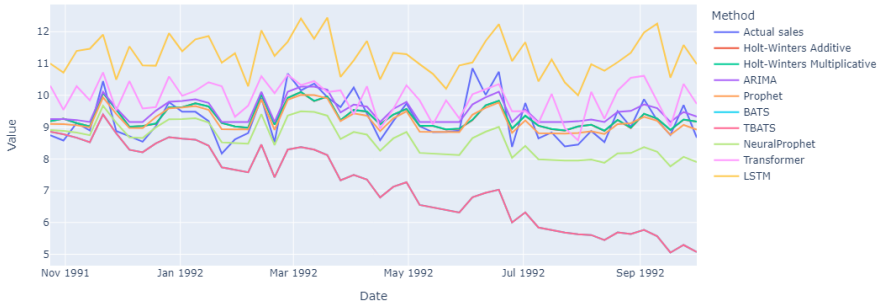
(b) The segmentation of the price data on Orange Juice

Figure 4.17: The segmentation of the sales- and price data on Orange Juice

The forecasting models, discussed in Section 2, were then used to make a prediction both using the baseline approach (no causalities) and by using our proposed method (see Section 3.6.2). The forecast results, using both methods, are presented in Figure 4.18.



(a) The results of running the baseline forecasting methods on Orange Juice



(b) The results of running Orange Juice, taking the causality signal into account

Figure 4.18: The results of running the baseline methods on Orange Juice without (a) and with (b) taking the causalities into account

A full table of the metrics evaluated on both forecasts is given in Table 4.5.

Table 4.5: A table of the metrics evaluated on the baseline forecast (T)- and with taking the causality signal into account (B) for Orange Juice

Method	MAE	MAPE	RMSE	NRMSE
(T) Baseline				
Holt-Winters Additive	0.735	8.137	0.860	0.321
Holt-Winters Multiplicative	0.744	8.244	0.874	0.326
ARIMA	0.565	5.972	0.688	0.256
Prophet	0.687	7.606	0.792	0.295
BATS	0.654	7.202	0.747	0.278
TBATS	0.654	7.202	0.747	0.278
NeuralProphet	1.275	13.358	1.511	0.563
Transformer	1.352	14.224	1.546	0.576
LSTM	0.694	7.357	0.884	0.330
(B) Causal				
Holt-Winters Additive	0.362	3.893	0.471	0.176
Holt-Winters Multiplicative	0.362	3.896	0.470	0.175
ARIMA	0.413	4.564	0.483	0.180
Prophet	0.322	3.408	0.430	0.160
BATS	2.095	22.495	2.428	0.905
TBATS	2.095	22.495	2.428	0.905
NeuralProphet	0.692	7.279	0.846	0.315
Transformer	0.742	8.234	0.869	0.324
LSTM	1.988	21.833	2.064	0.769

The forecast value add for the results using the MAPE metric is given in Figure 4.19.

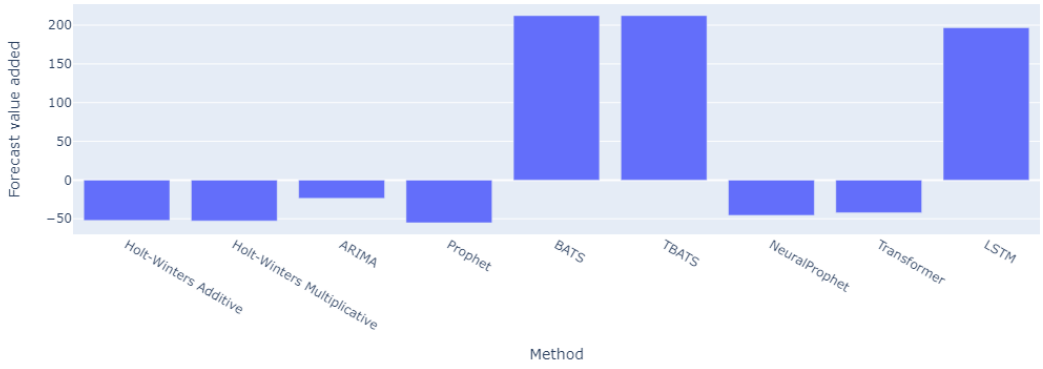


Figure 4.19: The FVA for Orange Juice

Note: A negative FVA means that the value of the loss function has decreased which, in this

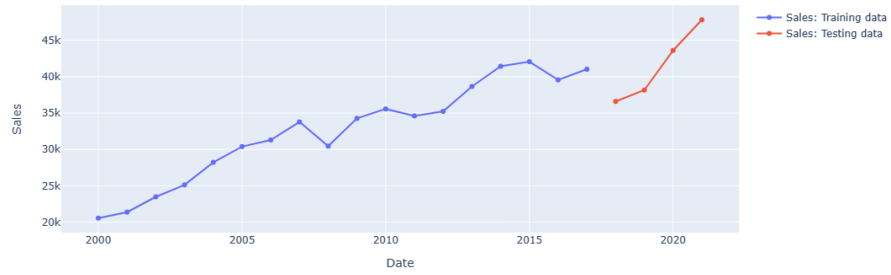
case, is a positive outcome.

From the FVA described in Figure 4.19 it can be seen that, in this case, an increase in the forecasting accuracy happens for most of the evaluated methods.

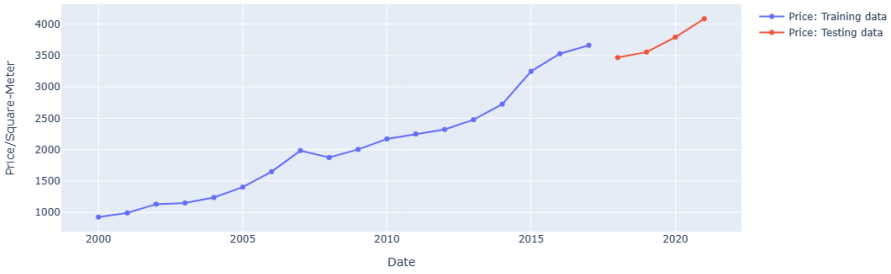
Note: In Table 4.5, BATS and TBATS yield the same results both with the baseline approach and when taking causalities into account. This can be explained by examining the signal in Figure 4.17. Here, it is noticeable that there is no definable seasonal pattern. Since the only difference between BATS and TBATS is the seasonal component, the methods would yield the same result if this component is optimized to not contribute to the forecast.

4.6 Apartment Sales

Both the sales signal and the price signal were segmented into training- and testing data. The different segmentations can be seen in Figure 4.20.



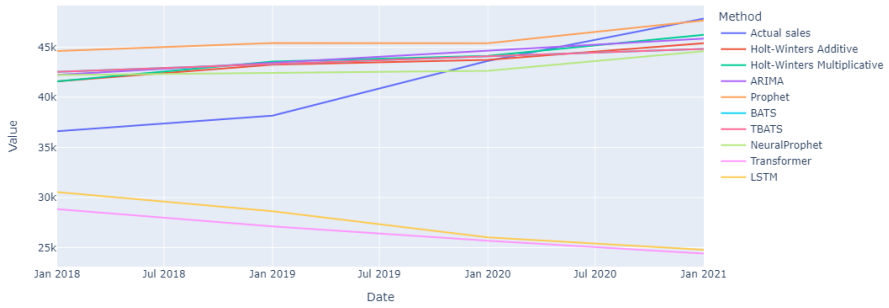
(a) The segmentation of the sales data on Apartment Sales



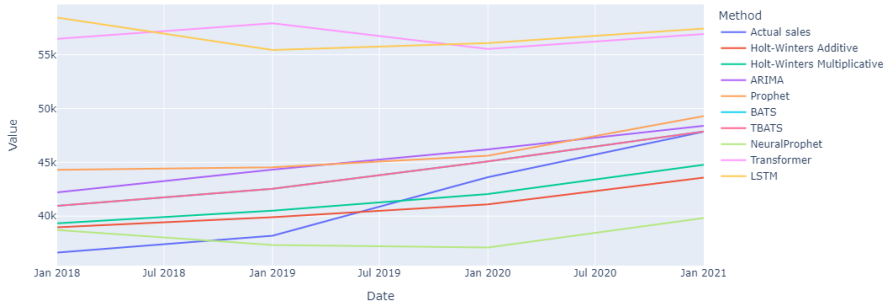
(b) The segmentation of the price data on Apartment Sales

Figure 4.20: The segmentation of the sales- and price data on Apartment Sales

The forecasting models, discussed in Section 2, were then used to make a prediction both using the baseline approach (no causalities) and by using our proposed method (see Section 3.6.2). The forecast results, using both methods, are presented in Figure 4.21.



(a) The results of running the baseline forecasting methods on Apartment Sales



(b) The results of running Apartment Sales, taking the causality signal into account

Figure 4.21: The results of running the baseline methods on Apartment Sales without (a) and with (b) taking the causalities into account

A full table of the metrics evaluated on both forecasts is given in Table 4.6.

Table 4.6: A table of the metrics evaluated on the baseline forecast (T)- and with taking the causality signal into account (B) for Apartment Sales

Method	MAE	MAPE	RMSE	NRMSE
(T) Baseline				
Holt-Winters Additive	3154.587	8.070	3759.283	0.335
Holt-Winters Multiplicative	3128.577	8.081	3771.073	0.336
ARIMA	3481.044	8.930	4016.660	0.358
Prophet	4304.211	11.330	5470.360	0.487
BATS	3639.019	9.261	4210.958	0.375
TBATS	3639.019	9.261	4210.958	0.375
NeuralProphet	3529.021	8.892	3913.907	0.348
Transformer	15038.550	35.050	16218.911	1.444
LSTM	14063.075	32.523	15562.630	1.385
(B) Causal				
Holt-Winters Additive	2718.820	6.418	2876.383	0.256
Holt-Winters Multiplicative	2421.448	5.886	2486.005	0.221
ARIMA	3722.271	9.625	4364.391	0.388
Prophet	4378.231	11.330	5143.873	0.458
BATS	2554.964	6.692	3170.737	0.282
TBATS	2554.964	6.692	3170.737	0.282
NeuralProphet	4375.528	9.925	5296.552	0.471
Transformer	15161.257	38.100	15891.968	1.415
LSTM	15299.777	38.405	15997.526	1.424

Note: For the baseline approach, Holt-Winters Additive is deemed to yield the best results due to the fact that it has the lowest metric scores for the majority of the metrics used.

The forecast value add for the results using the MAPE metric is given in Figure 4.22.

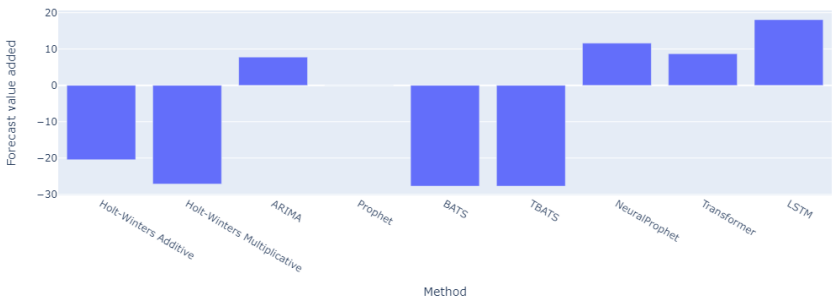


Figure 4.22: The FVA for Apartment Sales

Note: A negative FVA means that the value of the loss function has decreased which, in this

case, is a positive outcome.

From the FVA described in Figure 4.22 it can be seen that, in this case, a variation of increases and decreases in the forecasting accuracy happens for the evaluated methods.

Note: In Table 4.6, BATS and TBATS yield the same results both with the baseline approach and when taking causalities into account. An explanation of why this occurs is presented in Section 4.5.

Chapter 5
Discussion

In this section, the results from the previous section will be discussed in more detail. This will be done by comparing the different results, including plots- and tables.

5.1 Generating our own datasets

As real-world examples of data were hard to come by, the decision to generate our own datasets was made. In these datasets, all information regarding components such as seasonality and trend will be known.

In real-world scenarios, every aspect of a signal is seldom known. Generated datasets have the benefit that every aspect of a signal can be taken into account when performing a forecast.

This means that it is possible to prove if any meaningful improvements can be made, taking causality signals into account, in an ideal case. If this was to be possible, one could argue that it should be possible even for real-world scenarios.

5.2 Generated Dataset 1

Two separate datasets were created from the same basic idea. One ideal signal and one signal where Gaussian noise was added to all components of this ideal signal. These signals can be seen in Figure 4.1 and Figure 4.5 respectively. The datasets were implemented containing daily data over more than two years.

The idea behind these datasets was to implement a signal with both a noticeable trend- and seasonality. The causality signal should be modest, not impacting the total signal in any significant way. The two causal signals can be seen in Figure 4.2 and Figure 4.6 respectively.

5.2.1 Baseline

We will begin to discuss the baseline forecasting results from this dataset, both in the ideal- and noisy case. It will then be compared to the results from the forecast where the causality signal was taken into account.

Ideal signal

Looking at Figure 4.3a, some methods manage to forecast the actual sales signal somewhat accurately. Some methods however were not able to make a good forecast.

Comparing all the methods, the LSTM, Transformer- and ARIMA models have the weakest performances, whereas the LSTM and Transformer models even start to decrease when the actual signal is increasing. The weak performance of these two machine learning methods may be due to a limitation in the tuning of the hyper-parameters of the model.

Both BATS and TBATS appear to have captured the data very well. It should be noted that both BATS and TBATS have the same settings for the seasonal frequencies and seems to leverage their seasonal representations well.

On the other hand, Prophet and NeuralProphet which, like TBATS, feature a Fourier series as a representation of the seasonal component do not perform as well as BATS and TBATS. While Prophet and NeuralProphet can be argued to have a somewhat good forecast, ARIMA appears to have forecasted a straight line.

Noisy signal

Looking at Figure 4.7a, the following can be observed:

As was the case for the ideal signal, the methods with the worst performance on the noisy signal dataset were LSTM, Transformer- and ARIMA with the addition of the multiplicative Holt-Winters method. Here, the LSTM and Transformer methods still decrease when the signal increases, while the Holt-Winters multiplicative method has a larger increase than the actual sales.

Together with the addition of the Holt-Winters additive method, BATS and TBATS still seem to have the best performance with added noise. It can be noted that the same frequencies are used for the noisy signal as for the ideal signal.

While not as good as BATS and TBATS, Prophet and NeuralProphet seem to make a somewhat good forecast. ARIMA however still manages to forecast a somewhat straight line.

5.2.2 Causal signal

The causal signal was added to the forecast as described in Section 3.6.2. The results are discussed below and will be compared to the baseline results.

Ideal signal

Looking at the FVA in Figure 4.4, some results were improved while others were weakened when the causal effect was added to the forecasting model.

The largest improvement can be seen in NeuralProphet, however, this does not mean that this method is the best performer, as can be observed by looking at the MAPE score in Table 4.1. Prophet, which only saw a minor increase in performance, can now be noted to be the best method according to the MAPE score.

BATS and TBATS both show a drop in accuracy when the baseline is compared to the causal case. This is most likely due to the fact that both of these methods perform quite well in the baseline case. It should however be noted that in spite of the decrease in performance, BATS is still among the best performers according to the MAPE scores.

Both the Holt-Winters multiplicative- and ARIMA methods appear to have a reduction in accuracy when used together with EconML. Also, the Holt-Winters additive method does not seem to improve at all and gives almost identical results compared to its baseline performance.

The neural network approaches saw minor changes in their performance. The LSTM model decreased its MAPE score slightly while the Transformer model increased its MAPE score. However, both of these methods now follow the overall pattern of the actual sales, something which they did not do in the baseline (see Figure 4.3b).

Noisy signal

When looking at the FVA, for the noisy signal, seen in Figure 4.8, it may be seen that all methods increase the forecasting accuracy.

ARIMA shows the largest improvement of all the methods. BATS, TBATS and the additive Holt-Winters methods all also show large improvements (about a 70-80% decrease in loss). This could be because of the fact that the added irregularity (noise) in the baseline scenario makes it harder for the algorithms to fit the seasonal cycles for the methods.

Overall, most methods seem to, more or less, perform accurate forecasts. The only methods that deviate in this case are the LSTM, Transformer- and Holt-Winters multiplicative methods. In the case of the LSTM and Transformer methods, this could be due to the amount of data that was available.

Compared to the ideal scenario, it seems that in the case where the signals are not regular but rather have some random oscillations, using the causal signal in the forecast helps a lot in increasing the accuracy of the forecast.

5.3 Generated Dataset 2

The same approach that was used for Generated Dataset 1 was used for Generated Dataset 2 (see Section 5.2). These signals can be seen in Figure 4.9 and Figure 4.13 respectively. The datasets were implemented containing daily data over more than two years.

The idea behind these datasets was to implement a signal with both a noticeable trend- and seasonality. The causality signal should be large, impacting the total signal in a significant way. The two causal signals can be seen in Figure 4.10 and Figure 4.14 respectively.

5.3.1 Baseline

We will begin to discuss the baseline forecasting results from this dataset, both in the ideal- and noisy case. It will then be compared to the results from the forecast where the causality signal was taken into account.

Ideal signal

Looking at Figure 4.11, some methods manage to forecast the actual sales signal somewhat accurately. Some methods however were not able to make a good forecast.

Comparing all the methods, the LSTM, Transformer, ARIMA- and Holt-Winters multiplicative models have the weakest performances, where the LSTM and Transformer models even start to decrease when the actual signal is increasing. One can also note for the case of the LSTM and Transformer models, that the starting point of the forecast has a severe offset. The weak performance of these two machine learning methods may be due to a limitation in the tuning of the hyper-parameters of the model.

Both BATS and TBATS appear to have captured the data well. This is most likely due to the same reason as was the case for Generated Dataset 1 (see Section 5.2).

On the other hand, Prophet and NeuralProphet which, like TBATS, feature a Fourier series as a representation of the seasonal component do not perform as well as BATS and TBATS. While Prophet and NeuralProphet can be argued to have a somewhat good forecast, ARIMA appears to have forecasted a straight line.

Noisy signal

Looking at Figure 4.15a, the following can be observed:

As was the case for the ideal signal, the methods with the worst performance on the noisy signal dataset were LSTM, Transformer- and ARIMA. In this case, however, the multiplicative method of Holt-Winters performs better than in the ideal case. Here, the LSTM and Transformer methods still decrease when the signal increases. This is also true for ARIMA.

Looking at Table 4.4, BATS, TBATS and the Holt-Winters additive method all have about

the same MAPE score. It can however be noted that TBATS has the best MAPE score for this forecast. For the case of BATS and TBATS, it can be noted that the same frequencies are used for the noisy signal as for the ideal signal.

While not as good as the TBATS, NeuralProphet seems to make a somewhat good forecast. As was the case in Generated Dataset 1, ARIMA forecasts a somewhat straight line.

5.3.2 Causal signal

The causal signal was added to the forecast as described in Section 3.6.2. The results are presented below and will be compared to the baseline results.

Ideal signal

Looking at the FVA in Figure 4.12, all results except BATS, TBATS and the Holt-Winters additive method have improved. Also, looking at Figure 4.11b, all models follow the same behavior as the actual sales. As was the case in Generated Dataset 1, the MAPE score of the Holt-Winters additive method is almost the same as in its baseline.

It should be noted that even though BATS and TBATS saw large decreases in performance, this is percentage based. Looking at the MAPE score in Table 4.3, these methods will still yield a good forecast.

The largest improvement can be seen in the Holt-winters multiplicative- and BATS methods. However, Prophet has the best performance according to the MAPE scores.

While following the same behavior as the actual sales, it can be argued that the ARIMA, LSTM, Transformer- and Holt-Winters multiplicative models still do not perform well on the data.

This in turn means that choosing BATS, TBATS, Prophet, NeuralProphet or the Holt-Winters additive method to perform this forecast could result in a good prediction.

Noisy signal

When looking at the FVA, for the noisy signal, seen in Figure 4.16, again all methods appear to have an increase in the forecasting accuracy.

While all methods show large improvements, both ARIMA and the Holt-Winters additive methods appear to have improved the most. The reason for the overall improvement could again be because of the increased irregularity due to the noise. This could make the signal harder to forecast without the inclusion of the causal signal.

When looking at Figure 4.15b, all models, with the exception of the LSTM and Transformer models, appear to perform an accurate forecast. One could note that these results are even better than the results from Generated Dataset 1. This could be because of the increased

impact that the causal signal has on the forecasted time-series.

It seems that again using the causal signal helps a lot when the signal has a lot of irregularities, especially if the impact of the causal signal is large.

5.4 Publicly available datasets

Finding good datasets to perform demand forecasting on was a challenge in itself. However, we managed to find one dataset that included the sales- and price of Orange Juice over a period of time (see Section 3.2.2).

In order to get a second non-generated dataset, data from SCB (see Section 3.2.3) was used. This dataset includes data on the sales- and price of apartments in Stockholm County over a period of time.

5.5 Orange Juice

The results from the Orange Juice dataset are analyzed and discussed in this section. This dataset was chosen as a good real-world example as it included both price and sales of the product over a period of time.

5.5.1 Baseline

Studying Figure 4.18a, it is apparent that no forecast has a good performance. There is a total of five methods that more or less have a prediction that resulted in a straight line: Prophet, BATS, TBATS, ARIMA and NeuralProphet. This could be because of the fact that the signal does not have clear patterns and therefore the methods cannot fit a good representation.

While the LSTM and Transformer methods seem to have a similar pattern to the actual sales, none of these makes a good prediction.

In fact, looking at the MAPE scores in Table 4.5, ARIMA has the lowest score. In this particular case, it is probable that the ARIMA model simply predicted the closest to the mean value.

5.5.2 Causal signal

Looking at the FVA in Figure 4.19, most of the methods show that an improvement has been made in the forecast accuracy for the testing period. Only BATS, TBATS and LSTM have decreased in their performance. One should note though that these are major decreases.

While a major improvement in performance is made both in the NeuralProphet and Transformer models, the forecasting result can still be argued to be bad.

The best forecasting methods are instead both Holt-Winters methods together with Prophet. All of these methods closely follow the actual sales, where Prophet has the best MAPE score of 3.408.

It can be noted that all methods appear to have roughly the same *shape* after the causal signal was applied to them. This is the causal effect that was fitted by EconML, which appears to help a lot to forecast this signal. This is however not the case for TBATS, as when the causal effect is removed before the forecast the method fits a bad trend to the signal which causes an inaccurate forecast.

5.6 Apartment Sales

This dataset was created as an example of what would happen to a forecast when using a causality that was probable, but not obvious. To use the square meter price of apartments in Stockholm county could be a causality regarding the demand, however, there are certainly more causalities involved such as e.g. interest rates, production rates, unemployment rate, etc.

Note that we have taken the prices of the whole of Stockholm County as a causality of the demand. This data could e.g. also be heavily skewed due to the high prices in the inner city of Stockholm. This most likely would cause the forecast to be misrepresentative.

5.6.1 Baseline

Looking at Figure 4.21a, it can be seen that both the LSTM and Transformer models have not managed to fit anything at all. This could be due to the fact that out of all the datasets this one had by far the least data. As has been discussed earlier, neural network models are highly data dependent, which could mean that the performance of these methods could suffer here, due to a lack of data.

Furthermore, none of the methods seem to capture the initial *dip* in the testing data, where all methods appear to have forecasted straight lines. This could indicate that the models have fitted a trend, but not the oscillations, in the data.

5.6.2 Causal signal

Looking at the forecast in Figure 4.21b, it may be seen that the forecast of the LSTM- and Transformer models are still quite off. It can also be noted that the *spread* of all the methods has increased. There is also an offset, for all methods, from the initial value of the actual signal in the testing period.

Looking at the FVA in Figure 4.22, one can notice that both Holt-Winters approaches, as well as BATS and TBATS appear to all increase their performance from using the causal relationship. It can also be noted that Prophet shows no improvement from its baseline. All other methods appear to show a decrease in forecasting accuracy.

One could however argue that, in the causal case, the same behavior as the actual sales can be spotted in some of the forecasts. Given more data or maybe one more causality as an input, the forecasts could be of higher quality.

Chapter 6
Conclusion

The generated datasets showed that a forecast can, in an ideal case, be improved by taking the causality signal into account. Knowing the type of signal and what seasonal patterns to expect, a forecast loss could be lowered significantly. In some cases, the loss was lowered as much as 90% or more. One can also draw the conclusion that the impact of the causality signal matters. Even though the forecast for both datasets improved by a lot, the more accurate forecast would be the one from Generated Dataset 2, where the causal impact was larger. The self-implemented neural network methods did however not perform as well as we had hoped. This could however possibly be improved by using more data.

In the case of the dataset including Orange Juice and its accompanied price, the causality signal being used actually improved the forecast of some methods by a lot. It is however notable that not all forecasting methods yielded a great result on the dataset. Even with this in mind, the data pattern (behavior of the forecast) was in almost all cases changed for the better. This is one of the reasons that more than a few different methods should be evaluated for each forecast. One method might be better for a specific case and quite bad for another.

As was seen in the results from Apartment Sales, using EconML with causality signals that might not influence the demand will not improve the forecast significantly. As EconML allows evaluation using refute, the signal that the user wants to take into account is allowed to be refuted. If the user still wants to proceed, it might be good to inform this user of a potential faulty forecast.

6.1 Research questions

The report will be concluded by restating the research questions in Section 1.2.1 with provided answers.

6.1.1 Research question 1

To what extent is it possible to improve the prediction of future demand using statistical models that take components such as e.g. seasonalities and trends into account?

The discussed methods such as Holt-Winters, BATS, TBATS, ARIMA, Prophet, NeuralProphet, LSTM- and Transformer models all appear to be able to forecast time-series while not taking causalities into account. It can be seen in the generated datasets that, if the signal does not have a lot of irregularities, these perform well even with including several seasonalities. However, as can be seen for the other datasets, the forecast might not always be the best and is probably heavily dependent on choosing the correct forecasting model.

Research question 1.1

Which statistical models can be used?

What can be seen in the ideal cases of the generated datasets is that BATS and TBATS perform very well. In the case where the seasonalities are known, no other method even comes close to the performance of these two methods. In some cases both Holt-Winters methods also manage to perform well, giving accurate forecasts. This is most likely due to the fact that the two Holt-Winters methods beforehand already know the seasonality pattern of the inputted signal.

However, it is not really possible to decide beforehand which of the methods that should be used on a signal that is not ideal. The best way to handle such a forecast would be to use as many methods as possible and evaluate on a case-to-case basis.

All decomposition models could be argued to have some level of explainability as the different components can be plotted and presented to e.g. an analyst. ARIMA, which is based on the ARMA process could perhaps be seen to be the least explainable out of all the models.

6.1.2 Research question 2

To what extent is it possible to predict future demand using machine learning methods such as e.g. neural networks?

The Transformer- and LSTM models, as well as NeuralProphet, all attempt to forecast the data using neural networks. These machine-learning methods appear to need a lot of data in order to perform well. One can note, that even in the ideal cases, the Transformer- and LSTM models still were not able to make a somewhat good forecast, even with including the causalities. NeuralProphet however managed to provide some good results.

The major hurdle which might make the methods less usable than the statistical decomposition models is that they are less explainable. This is due to that the weights in the different network layers do not explain as much as being able to plot the different components.

Research question 2.1

Which machine learning methods are preferable?

NeuralProphet, which is also a hybrid method that incorporates methods from e.g. Holt-Winters to make its forecast performs the best of the three in all provided cases. In terms of explainability, NeuralProphet will be the most explainable out of the three machine learning methods. This is due to that this method uses decompositional components that can, as in the statistical case, be plotted and presented to an analyst.

6.1.3 Research question 3

To what extent is it possible to improve the prediction of future demand using models that take causality signals into account?

Using a library such as EconML, which does causal analysis, it is possible to analyze the causal effect and combine the results with the already existing forecasting methods to alter the results of a forecast. From the provided datasets, the majority of them can be argued to have gotten a vast improvement in forecasting accuracy when using EconML. It also appears that the larger the impact of the causal signal, the better the performance, while using the causal signal to aid in the forecast.

By utilizing the approach of creating a component that aims to extend existing models, it can be argued that some level of explainability is created in the final model. This is due to the fact that a causal effect can be plotted and shown to a user to explain what the estimated effect of the causal signal is on the forecasted time-series.

Research question 3.1

Which methods regarding model building are preferable (statistical, neural networks, etc)?

Which method performs best really appears to depend on a case-to-case basis. It would make sense to once again, try a set of several methods and select the one that performs best. It however appears that if the dataset is small, neural networks are not preferable over statistical methods.

In order to create some measure of explainability, the analyst can be presented with a plot of the causal effect to get feedback on the choice of the signal.

Research question 3.2

To what extent is the forecasting performance impacted when looking at a signal that has a large causal effect compared to a signal that has a smaller causal effect?

When comparing the two generated signals, it can be seen that the causality signal with the larger impact on the total signal gives a larger improvement in performance. Also, one can argue that the price data for Orange Juice has a larger impact on its forecast than the price per square meters data in Apartment sales has on its forecast, for the reasons explained in Section 5.6. One can see in the results that the forecast for Orange Juice when taking the causality signal into account, is superior to the forecast of Apartment Sales.

6.2 Future work

One way this project can be extended is by going deeper into the EconML ecosystem. This is a powerful tool that can do way more than what is done in this project. If implemented correctly, it might even boost the performance of the forecasts further. A few examples are listed below.

It would be possible to add multiple causal terms using the same approach as discussed in Section 3.6.2. Furthermore, EconML is able to make much more complicated causal models than those utilized in this project. This means that more complicated relationships can be modeled and used for forecasting.

The causal effects of several different signals can be compared in order to establish a ranking. This ranking could help to determine which signals to use for forecasting from a potential catalog of causal signals.

Another way to extend this project is by diving deeper into neural networks. It would be possible to make a whole project just on how to tune the neural networks to perform the forecasts.

Some effects have not been handled in this project, e.g. if the effect of the causal signal is lagged. This could be handled in a similar way as the lagged regressors in NeuralProphet and could perhaps be an interesting project in itself.

References

[1] A. Zafer Acar and Batuhan Kocaoğlu. *An Intelligent Approach to Demand Forecasting*. June 2014. URL: https://www.researchgate.net/publication/327225380_An_Intelligent_Approach_to_Demand_Forecasting (visited on 12/14/2022).

[2] Oskar Triebe et al. *NeuralProphet: Explainable Forecasting at Scale*. Nov. 29, 2021. URL: http://arxiv.org/abs/2111.15397 (visited on 10/30/2022).

[3] *Prophet*. URL: http://facebook.github.io/prophet/ (visited on 08/30/2022).

[4] Sean J. Taylor and Benjamin Letham. *Forecasting at scale*. en. Tech. rep. e3190v2. ISSN: 2167-9843. PeerJ Inc., Sept. 2017. DOI: 10.7287/peerj.preprints.3190v2. URL: https://peerj.com/preprints/3190 (visited on 10/22/2022).

[5] Amit Sharma and Emre Kiciman. *DoWhy: An End-to-End Library for Causal Inference*. arXiv:2011.04216 [cs, econ, stat]. Nov. 2020. URL: http://arxiv.org/abs/2011.04216 (visited on 12/15/2022).

[6] *EconML: A Python Package for ML-Based Heterogeneous Treatment Effects Estimation*. original-date: 2018-04-30T21:02:52Z. Mar. 2023. URL: https://github.com/microsoft/EconML (visited on 03/04/2023).

[7] *Introduction to Causal Inference — econml 0.13.1 documentation*. URL: https://econml.azurewebsites.net/spec/causal_intro.html (visited on 11/14/2022).

[8] *About Causal ML — causalml documentation*. URL: https://causalml.readthedocs.io/en/latest/about.html (visited on 11/05/2022).

[9] Huigang Chen et al. *CausalML: Python Package for Causal Machine Learning*. arXiv:2002.11631 [cs, stat]. Mar. 2020. URL: http://arxiv.org/abs/2002.11631 (visited on 02/27/2023).

[10] *Interpretable Causal ML — causalml documentation*. URL: https://causalml.readthedocs.io/en/latest/interpretation.html (visited on 02/27/2023).

[11] Lindgren, Georg, Rootzén, Holger, and Sandsten, Maria. *Stationary stochastic processes for scientists and engineers*. English. CRC Press. ISBN: 978-1-4665-8618-5.

[12] *Forecasting: Principles and Practice (2nd ed)*. URL: https://otexts.com/fpp2 (visited on 10/22/2022).

[13] Glen Stephanie. *ARMA model*. en-US. Jan. 2019. URL: https://www.statisticshowto.com/arma-model/ (visited on 12/14/2022).

[14] *Autoregressive integrated moving average*. Page Version ID: 1116193509. Oct. 2022. URL: https://en.wikipedia.org/w/index.php?title=Autoregressive_integrated_moving_average&oldid=1116193509 (visited on 11/25/2022).

[15] Jamal Fattah et al. "Forecasting of demand using ARIMA model". In: *International Journal of Engineering Business Management* 10 (Oct. 2018), p. 184797901880867. DOI: 10.1177/1847979018808673.

[16] *statsmodels.tsa.holtwinters.ExponentialSmoothing.fit — statsmodels*. URL: https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.fit.html#statsmodels.tsa.holtwinters.ExponentialSmoothing.fit (visited on 12/14/2022).

[17] Marco Taboga. *Log-likelihood : Lectures on probability theory and mathematical statistics*. Publisher: [Kindle Direct Publishing]. URL: https://www.statlect.com/glossary/log-likelihood (visited on 03/05/2023).

[18] Alysha M. De Livera, Rob J. Hyndman, and Ralph D. Snyder. "Forecasting Time Series With Complex Seasonal Patterns Using Exponential Smoothing". In: *Journal of the American Statistical Association* 106.496 (2011). Publisher: [American Statistical Association, Taylor & Francis, Ltd.], pp. 1513–1527. ISSN: 0162-1459. URL: http://www.jstor.org/stable/23239555 (visited on 10/22/2022).

[19] *Box Cox Transformation: Definition, Examples*. en-US. URL: https://www.statisticshowto.com/probability-and-statistics/normal-distributions/box-cox-transformation/ (visited on 12/14/2022).

[20] Oskar Triebe, Nikolay Laptev, and Ram Rajagopal. *AR-Net: A simple Auto-Regressive Neural Network for time-series*. arXiv:1911.12436 [cs, stat]. Nov. 2019. URL: http://arxiv.org/abs/1911.12436 (visited on 11/22/2022).

[21] Ohlsson, Mattias and Edén, Patrik. *Introduction to Artificial Neural Networks and Deep Learning*. English. Lund University.

[22] *Neural machine translation with a Transformer and Keras | Text*. en. URL: https://www.tensorflow.org/text/tutorials/transformer (visited on 02/07/2023).

[23] Padhma M. *Evaluation Metric for Regression Models*. en. Oct. 2021. URL: https://www.analyticsvidhya.com/blog/2021/10/evaluation-metric-for-regression-models/ (visited on 11/27/2022).

[24] Filip Chybalski. "Forecast Value Added (FVA) Analysis as a Means to Improve the Efficiency of a Forecasting Process". In: *Operations Research and Decisions* 27 (Jan. 2017), pp. 5–19. DOI: 10.5277/ord170101.

[25] *Overview — econml 0.14.0 documentation*. URL: https://econml.azurewebsites.net/spec/overview.html (visited on 12/15/2022).

[26] *Orthogonal/Double Machine Learning — econml 0.13.1 documentation*. URL: https://econml.azurewebsites.net/spec/estimation/dml.html (visited on 11/14/2022).

[27] Pierre Gutierrez and Jean-Yves Gérardy. "Causal Inference and Uplift Modelling: A Review of the Literature". en. In: *Proceedings of The 3rd International Conference on Predictive Applications and APIs*. ISSN: 2640-3498. PMLR, July 2017, pp. 1–13. URL: https://proceedings.mlr.press/v67/gutierrez17a.html (visited on 12/15/2022).

[28] Victor Chernozhukov et al. *Double/Debiased Machine Learning for Treatment and Causal Parameters*. arXiv:1608.00060 [econ, stat]. Dec. 2017. URL: http://arxiv.org/abs/1608.00060 (visited on 12/11/2022).

[29] *Problem Setup and API Design — econml 0.14.0 documentation*. URL: https://econml.azurewebsites.net/spec/api.html (visited on 12/11/2022).

[30] *What is Kaggle, Why I Participate, What is the Impact? | Data Science and Machine Learning*. en. URL: https://www.kaggle.com/getting-started/a (visited on 12/15/2022).

[31] *Store Item Demand Forecasting Challenge*. en. URL: https://kaggle.com/competitions/demand-forecasting-kernels-only (visited on 10/23/2022).

[32] *Dominick's Dataset*. en. URL: http://www.chicagobooth.edu/research/kilts/datasets/dominicks (visited on 12/02/2022).

[33] *Official Statistics of Sweden*. en. URL: https://www.scb.se/en/About-us/official-statistics-of-sweden/ (visited on 12/15/2022).

[34] *Statistikdatabasen - Välj tabell*. URL: https://www.statistikdatabasen.scb.se/pxweb/sv/ssd/ (visited on 12/15/2022).

[35] *statsmodels.tsa.holtwinters.Holt — statsmodels*. URL: https://www.statsmodels.org/stable/generated/statsmodels.tsa.holtwinters.Holt.html (visited on 10/24/2022).

[36] *Introduction — statsmodels*. URL: https://www.statsmodels.org/stable/index.html (visited on 10/26/2022).

[37] *pmdarima · PyPI*. URL: https://pypi.org/project/pmdarima/ (visited on 10/24/2022).

[38] Grzegorz Skorupa (intive). *tbats: BATS and TBATS for time series forecasting*. URL: https://github.com/intive-DataScience/tbats (visited on 10/24/2022).

[39] Keras Team. *Keras documentation: About Keras*. en. URL: https://keras.io/about/ (visited on 11/14/2022).

[40] *sklearn.ensemble.GradientBoostingRegressor*. en. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html (visited on 02/12/2023).

[41] *6.3. Preprocessing data*. en. URL: https://scikit-learn.org/stable/modules/preprocessing.html (visited on 03/05/2023).

[42] *py-why/dowhy: DoWhy is a Python library for causal inference that supports explicit modeling and testing of causal assumptions. DoWhy is based on a unified language for causal inference, combining causal graphical models and potential outcomes frameworks*. URL: https://github.com/py-why/dowhy (visited on 02/12/2023).

[43] *4. Refute the obtained estimate — DoWhy documentation.* URL: https://www.pywhy.org/dowhy/v0.8/user_guide/effect_inference/refute.html (visited on 02/12/2023).

[44] *econml.dml.DML — econml 0.14.0 documentation.* URL: https://econml.azurewebsites.net/_autosummary/econml.dml.DML.html (visited on 02/12/2023).

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2023-03-16

EXAMENSARBETE Explainable Demand Forecasting using Causalities and Machine Learning Models
STUDENTER Marcus Sundell, Marlon Abeln
HANDLEDARE Volker Krueger (LTH), Bahram Zarrin (Microsoft)
EXAMINATOR Jacek Malec (LTH)

The Perfect Match? Demand Forecasting and Causality Analysis

POPULÄRVETENSKAPLIG SAMMANFATTNING **Marcus Sundell, Marlon Abeln**

To what extent does the combination of demand forecasting algorithms together with causal models improve the forecasting accuracy? With this new proposed model, it is found that this approach could yield an increase in the forecasting accuracy given that the affecting signal is of significant impact.

Demand forecasting is a necessary discipline for businesses to try and predict all aspects of future demand for their products. This would give a business the ability to plan its production more accurately. This could e.g. help with decisions regarding the number of resources to buy for the coming period.

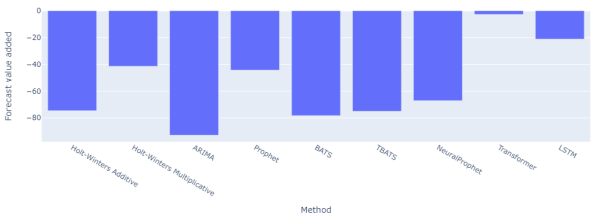
There exist several methods that are used to forecast demand sampled as time series. In this project, three of these methods are considered, with the main focus on decomposition models.

Decomposition models break the time series into a set of additive or multiplicative components. Examples of these could be a seasonal component and/or a long-term trend component.

Another topic of research is the analysis of causal relationships between different time series data. This would allow a forecaster to estimate the magnitude of influence one time series could have on another.

The proposed model is a combination of traditional demand forecasting models and causality analysis. It is also a decomposition model, where a causal component is added to a forecasting model. This is argued to, in some cases, increase the accuracy of a forecast.

Using a generated dataset where the causality signal is known, the proposed model shows a significant increase in forecasting accuracy. The figure shows a percentage decrease in the loss of the forecast over a testing period.



Percentage decreases in loss using the method

However, real-world scenarios are not as simple. The reader will also be shown demand forecasting done on real-world examples. Comparisons between the results before and after taking the causalities into account are presented, in both good and bad examples of chosen causality signals. An explanation of the performance of the neural network methods is also provided.