CBS COPENHAGEN BUSINESS SCHOOL
HANDELSHØJSKOLEN

Forecasting Revenue in Limited Data Settings

A comparative study of statistical and Machine Learning methods for start-ups and scale-ups in B2B SaaS

Master thesis

Programme: MSc in Business Administration and Data Science
Author: Faye Hahn (149875)
Supervisor: Daniel Hain
Date of Submission: 14.05.2023

Characters: 158.419

Pages: 72

Abstract

Forecasting is vital to any successful business and applies to various functions and industries. One important application is in Financial Planning, e.g., when estimating revenue targets. In this context, this thesis investigates using quantitative forecasting methods to support revenue planning for B2B SaaS start-up and scale-up companies. The research addresses two main questions: Q1) Can quantitative methods generate realistic forecasts for short, volatile revenue time series? And Q2) Which method works best and should thus be adopted for different types of datasets? The study focuses on Growblocks, a company specializing in Revenue Operations software and services. As part of their daily business of generating revenue forecasts for their customers, Growblocks faces several data-related challenges which require them to rely on expert knowledge for their predictions. To answer the research questions, several experiments were conducted using SARIMAX, NeuralProphet, and Temporal Fusion Transformer models on different dataset groups with varying characteristics. The findings ultimately show that the suitability of quantitative methods depends on the complexity and length of the time series. Simple dynamics can be captured with a rather basic model like SARIMAX trained on shorter time series. At the same time, complex patterns require more advanced models like the NeuralProphet and more historical data for training. A very low number of data points hinders the effectiveness of quantitative models. Ultimately, the results emphasize using a combination approach in which qualitative and quantitative methods are used jointly to leverage the advantages of both classes. This research provides insights into leveraging quantitative forecasting methods for revenue planning and emphasizes the importance of considering dataset characteristics when selecting appropriate models. It further provides a reevaluation of the Growblocks approach with recommendations on how to use the findings to improve their existing setup.

Keywords: Time series, Forecasting, RevOps, Machine Learning, limited data, volatility

Table of Contents

Table of Figures

Table of Tables

Table of Symbols

Abbreviations

ACF	Autocorrelation function
ACV	Average contract value
ADF test	Augmented Dickey-Fuller test
AICc	Akaike's Information Criterion with small sample correction
AR	Autoregression model
B2B	Business-to-Business
CVR	Conversion rate
EDA	Exploratory data analysis
FTE	Full-time equivalent
I	Integration
MA	Moving Averages model
MQL	Marketing-qualified lead
ML/DL	Machine Learning and Deep Learning
MSE	Mean squared error
NP	NeuralProphet
RevOps	Revenue Operations
RMSE	Root mean squared error
SaaS	Software-as-a-Service
SARIMAX	Seasonal Autoregressive Integrated Moving Averages model with exogenous regression component
SDR	Sales Development Representative

TFT	Temporal Fusion Transformer

Variables

t	time step
T	length of a time series
m	periodicity of a time series
y_t	target value at time step t
h	forecasting horizon
p	order of the non-seasonal AR component; number of lags to include in the (non-seasonal) AR model
d	order of the non-seasonal I component
q	order of the non-seasonal MA component
P	order of the seasonal AR component
D	order of the seasonal I component
Q	order of the seasonal MA component
l	number of hidden layers
hs	size of hidden layers
$epochs$	number of training epochs

1 Introduction

To start this thesis, I first want to draw on its motivation and relevance in Chapter 1.1. Based on that, the resulting objectives of the work and how it contributes to existing research and practice is presented in Chapter 1.2

1.1 Motivation

The Role of Forecasting in Business

The importance of forecasting in business has been known for long. That is no surprise, as whenever there is decision-making involved, strategic leaders need to think about the future and the effect of their actions in that context (Robinson, 1965). However, business forecasting has become even more widespread with the development of statistical and mathematical models in the 20th century through works such as Box & Jenkins (1970) and Makridakis et al. (1983). It is essential to any business by now.

Forecasting in business can serve various purposes across different functions and industries. One possible field of application is for managing inventory levels, production planning, and transportation in the supply chain, as seen in a study by Babai et al. from 2013. Their results emphasize how accurate demand forecasting can lead to reduced lead times, decreased stockouts, and optimized resource utilization. Another example can be found in Human Resources Planning. Forecasting labor needs helps organizations determine the right size of their workforce, hire the right people, and allocate resources efficiently – Bulla & Scott (1987) highlights the importance of workforce forecasting in human resource management. Often, businesses also use forecasting for financial planning and budgeting. Here, it supports them in estimating revenues, expenses, and profits, enabling them to develop budgets, assess potential investments, and manage their cash flow. An example of this is Dechow et al. (2000) which investigates the relationship between financial statement information and analysts' long-term revenue forecasts, demonstrating the practical application of forecasting in financial planning. A research chapter by Coad & Hölzl from 2012 further provides insights into how forecasting is applied in the growth planning of firms, highlighting its importance in shaping financial strategies and resource allocation.

Altogether, it shows how forecasting is ever-present in today's businesses and plays a critical role in their success, enabling them to anticipate future trends and challenges.

Setting realistic growth targets

Especially when it comes to setting revenue targets, forecasting can be highly valuable. Setting realistic revenue targets is vital to any business that wants to keep running and stay profitable.

For a long time, revenue targets in B2B have been set by CEOs in a top-down manner. In this approach, the company's growth potential was mainly assessed based on assumptions, e.g., about what share of the available market they can capture, while not taking information on how the business is working to generate revenue into account (Growblocks, personal communication, 2023).

Over time, Sales and Marketing operations as critical drivers of revenue generation have become increasingly complex. This poses a new challenge for target planning as top-down approaches lack to account for the specialties in how different businesses operate and drive revenue (Growblocks, personal communication, 2023). As a consequence, companies struggle to hit their set targets which has several risks associated with it, as discussed in a recent blog post by the online incubator and accelerator *FasterCapital*. First, not hitting revenue targets can keep the company from getting funding. Investors want to see the business grow. In turn, if the company is not hitting its growth targets, it can be a red flag for investors, rendering them less likely to invest. Secondly, it could negatively affect the motivation and engagement of the employees who want to see the company doing well. This, in turn, can then harm the company's overall performance. Another risk is that customers might wander off if they do not feel that the company is growing and progressing and instead take their business elsewhere. But not only is there a risk of existing employees and customers losing trust in the business, it could also make it harder to attract new ones if targets are unmet. The reason for this is that not hitting targets could eventually lead to a damaged reputation of the company not performing well, which can discourage potential new clients or employees (FasterCapital, 2023). Overall, not hitting the revenue targets, in the long run, will keep a company from growing and expanding, which is why it is essential to set realistic, achievable targets.

The Rise of Revenue Operations

A new business function has emerged in B2B companies in the last couple of years to mitigate the problem of not hitting targets. Revenue Operations (RevOps) has been implemented to provide a countermeasure to top-down targets by bridging the gap between those setting them and those driving them (Growblocks, personal communication, 2023).

Traditionally, organizations worked in isolated departments with disparate missions. RevOps aims to break down data silos by aligning operations across Marketing, Sales, and Customer

Success creating one streamlined end-to-end process. By this, they attempt to achieve full-funnel visibility and accountability to maximize the business's revenue potential. In addition, when done correctly, RevOps will change how internal teams collaborate, improve customer acquisition, create a smoother customer journey, and establish a company culture focused on driving growth (Digitopia, Inc., 2023).

The process in the center of RevOps is the (B2B) Sales funnel. Different definitions exist of this funnel, mainly varying in how they split it. For this thesis, I will use a description featuring four stages as can be seen, e.g., in a blog post by the revenue intelligence software company *6Sense*.

The definition focuses on the persona at the center of each customer journey stage. In the first stage, the focus is on generating 'lead'. A lead can be defined as people that might have a use for the company's product but first need to learn about the brand to realize their need for it. To generate new leads, typical activities include advertising, content marketing, social media, and similar channels that can be used to raise awareness about the business. Usually, this is done by the Marketing team (6sense, 2023).

After being evaluated based on whether they match the ideal customer profile as well as on their likelihood of being converted into paying customers, leads convert into prospects – often referred to as qualified leads – which is the focus of the second stage. Prospects have already become aware of the product. Still, they are not yet sure whether to become buyers and can therefore be located in what is often referred to as a consideration phase (6Sense, 2021).

As soon as prospects realize their need for the product and have shown to be qualified, they move to the buyer stage and start moving through the Sales pipeline. In this stage, they are actively in conversation with the company's Sales team regarding product demonstrations and deal negotiations later in the process. Ultimately, the goal is to nudge the people in the buying stage to a purchase decision and close the deal on them (6Sense, 2021).

Once they decide on the purchase, they move to the last stage of the Sales funnel, which focuses on nurturing customers. As it is not only relevant to generate new customers but further to retain and potentially upsell the existing paying customer base, the funnel does not end as soon as the business wins a new customer. Instead, the funnel also includes the Customer Success team to drive a smooth overall customer experience (6Sense, 2021). A visualization of the resulting funnel can be found in Appendix 1.

Growblocks

One company specializing in RevOps – or, more precisely, revenue planning – is the Danish software start-up *Growblocks ApS*. Existing solutions have so far mainly focused on providing a tool to track the movements in the Sales funnel. The founders of Growblocks wanted to take this a step further by including an engine to support strategic target planning and execution. Co-founder and CEO Toni Holbein, who has led RevOps teams in companies like *Planday* and *Falcon.io* before, described their initial idea as follows:

> *"We would have loved to have a tool that connected the planning of growth with the operational reality of driving it, but there was no product in the market that focused on revenue execution across management layers & data silos – so we decided to build it." (Growblocks, 2023)*

They did that by creating a customized model that aims to mimic the individual behavior of the customers' Sales funnel based on historical data on the metrics included in the funnel. Based on that, the customer can add future actions and initiatives and simulate their effect on the projected revenue. This can be used then to derive a more realistic bottom-up target for revenue growth and continuously track the company's performance against it (Growblocks, 2023). The customers of Growblocks are typically start-up and scale-up companies operating in the B2B SaaS industry. By becoming customers, they get access to their revenue model and are offered consultation by a team of experienced RevOps practitioners. This team can support them, e.g., in analyzing their current performance and optimizing their revenue potential.

The problem

Working together with many customers, who themselves are start-up and scale-up companies, also shows how challenging it is to come up with a good revenue projection. Not only are the provided datasets generally limited in size, they also show to be very heterogeneous. While some time series follow a relatively stable course, we often see unstable dynamics and volatility in many other datasets. Especially in the case of a low data setting, these dynamics can be very hard to predict.

Currently, the Growblocks model builds largely upon expert knowledge. This is needed to encode known dynamics and specialties of the series, which the model itself cannot capture from the historical data alone. Consequently, the model and its forecasts need to be understood as being subjective (at least to a certain degree). They can, thus, potentially be biased, as discussed in Hyndman & Athanasopoulos (2021).

1.2 Objective and Contribution

In this thesis, I want to investigate how statistical models and Machine Learning can be leveraged for forecasting the revenue of start-up and scale-up companies operating in B2B SaaS. This objective includes two underlying research questions I will try to answer in this thesis.

> **Q1) Can quantitative methods generate realistic forecasts for short, volatile revenue time series?**
>
> **Q2) Which method works best and should thus be adopted for different types of datasets?**

While Q1 focuses more on providing a high-level evaluation of the suitability of purely quantitative models for the use case we see at Growblocks, Q2 builds upon its results. It extends them by providing individual recommendations for different customer types that the business can choose to act upon.

From a business perspective, this is valuable to the extent that Growblocks can use it to improve the existing offering for their customers. Further, the results might also be helpful for start-ups and scale-ups operating revenue planning independently and considering which method might lead to the most accurate results in their case.

From a scientific perspective, this thesis will add to the discussion about forecasting short time series, forecasting revenue time-series as well as around forecasting time series with volatility. Further, it will extend the existing literature by providing insights on the specific intersection of challenges inherent to this use case's forecasting task, as in most literature, they are covered separately (see also Chapter 3.4).

To answer my research questions, the thesis is structured as follows. Chapter 2 starts by introducing the process of time series forecasting as well as the main terminology around it. In Chapter 3, I walk through this process again, this time describing the practical execution of it in the present use case. Therefore, I start by diving into a more detailed overview of the forecasting problem, followed by a description of the data collection approach in Chapter 3.1. After that, Chapter 3.2 explains the Growblocks forecasting model to provide a better understanding of how it works and how a new approach could potentially add to that. Before going into the selection of models, Chapter 3.3 classifies the available datasets into groups and elaborates on the results of the exploratory data analysis for each group. Further, to get a better intuition of how the forecasting task can be approached, the results of my research on related work are presented in Chapter 3.4. The last subchapter of Chapter 3 covers the experimentation

approach, which further includes a description of the selected methods and their key components. After having covered the methodology of my experiments in Chapter 3, Chapter 4 focuses on the presentation and analysis of the results to answer the overarching research questions. Finally, I bring it together in Chapter 6 by summarizing the approach and my main findings and putting them into context not only of my research questions but also of existing research in this field. Additionally, this chapter provides some recommendations for Growblocks regarding the integrations of findings in their existing process. Lastly, it elaborates on some of the limitations of this thesis and draw attention to potential areas for future research based on my findings in this thesis.

2 Time Series Forecasting

Before heading into the practical part of this thesis, this chapter should serve to build an understanding of the theoretical background of time series forecasting. Therefore, Chapter 2.1 starts with a short overview of the time series and the main summary statistics in that context. Chapter 2.2 then introduces the task of forecasting, including all steps which are part of the process.

2.1 Time Series

When collecting sequential data points over time, this is referred to as time series data (Hyndman & Athanasopoulos, 2021). This sort of data can be observed in various fields like meteorology (e.g., daily temperatures in Denmark (World Bank Climate Change Knowledge Portal, 2023)), finance (e.g., daily exchange rates for foreign currencies (Statistics Denmark, 2023)) or consumption (e.g., yearly meat consumption per person (OECD, 2023)).

And even in business, time series data is omnipresent, playing a crucial role in various aspects of operations and decision-making. From sales and financial data to customer behavior and supply chain management, businesses rely on time series analysis to extract meaningful insights (Franses et al., 2014). For example, analyzing sales time series helps identify demand patterns, enabling effective inventory management and revenue forecasting (Ensafi et al., 2022). In financial markets, time series analysis aids in understanding price movements and developing investment strategies (Baz et al., 2015; Shynkevich et al., 2017).

In theory, time series data can be described as a sequence of random variables $\{Y_t | t = 0, \pm 1, \pm 2, \pm 3 \dots\}$, also called a stochastic process (Cryer & Chan, 2008). The mean of a stochastic process can be defined as

$$\mu_t = E(Y_t) \qquad \text{for } t = 0, \pm 1, \pm 2, \ldots \tag{1}$$

Thereby, it can be understood as the expected value of the process up to time t which further implies that the mean can be different at each time step t (Cryer & Chan, 2008). Additionally, following the notation from the book, the variation of the process is defined as

$$Var(Y_t) = E[(Y_t - \mu_t)(Y_t - \mu_t)] \qquad \text{for } t = 0, \pm 1, \pm 2, \ldots \tag{1}$$

Other summary statistics that are often included when analyzing time series are the minimum, first quartile, median, third quartile, and maximum values (Hyndman & Athanasopoulos, 2021).

When the time series captures the development of multiple variables at once, we can further calculate the covariance for each pair of variables. The covariance thereby measures how changes in a first variable X_t affect a second variable Y_t (Lindgren, 1993). For any combination of variables X_t and Y_t, the covariance can be defined as

$$Cov(X_t, Y_t) = E[(X_t - \mu_{X_t})(Y_t - \mu_{Y_t})] \qquad \text{for } t = 0, \pm 1, \pm 2, \ldots \tag{3}$$

where μ_{X_t} and μ_{Y_t} are the respective means for each time series. To exclude the effect of scale from the covariance, we often also look at the correlation between two variables which is defined as

$$Corr(X_t, Y_t) = \frac{Cov(X_t, Y_t)}{\sqrt{Var(X_t)Var(Y_t)}}. \qquad \text{for } t = 0, \pm 1, \pm 2, \ldots \tag{4}$$

It is also possible to compute the Covariance and Correlation for a time series with itself. This is then called Autocovariance and Autocorrelation and measures the linear relationship of the data at each time step with respect to its preceding, i.e., *lagged* values (Cryer & Chan, 2008; Hyndman & Athanasopoulos, 2021).

An overview of the presented summary statistics for analyzing time series is displayed in Table 1.

Table 1: Summary statistics for time series

Metric name	Formula
Expectation value	$\mu_t = E(Y_t)$
Variance	$Var(Y_t) = E[(Y_t - \mu_t)(Y_t - \mu_t)]$
Minimum	$Min(Y_t)$
First Quantile	$Q1(Y_t) = Y_t\left[\frac{1}{4}(t+1)\right]$
Median	$Med(Y_t) = Y_t\left[\frac{1}{2}(t+1)\right]$

Third Quantile	$Q3(Y_t) = Y_t\left[\frac{3}{4}(t+1)\right]$
Maximum	$Max(Y_t)$
Covariance	$Cov(X_t, Y_t) = E[(X_t - \mu_{X_t})(Y_t - \mu_{Y_t})]$
Correlation	$Corr(X_t, Y_t) = \frac{Cov(X_t, Y_t)}{\sqrt{Var(X_t)Var(Y_t)}}$

2.2 Forecasting process

In time series forecasting, we typically attempt to predict future values of a time series based on available information, including historical data and knowledge of future events that might affect it in the future (Hyndman & Athanasopoulos, 2021). In business, it is often used to inform decisions and helps in long-term strategic planning. Hyndman & Athanasopoulos (2021) structure the task of forecasting into five main steps: Problem definition, gathering information, exploratory analysis, model selection and fitting, and finally, applying and evaluating the forecasting model. In the following, each step is described in detail. Therefore, I primarily rely on the book mentioned above by Hyndman & Athanasopoulos, as it covers a broad range of topics in time series analysis and forecasting.

Step 1: Problem Definition

According to the authors, the first step of a forecasting task might also often be the most difficult one. To define the problem correctly, we first need to understand how the forecast will be used, where it will be used, and who will use it. A forecaster should thus ensure to gather enough information and spend time talking to key stakeholders and anyone involved in the data collection and maintenance of databases.

This step also includes the decision about the forecasting horizon, i.e., how far our model needs to predict in the future.

Step 2: Gathering Information

After defining the problem thoroughly, we must gather relevant information and data. Thereby, Hyndman & Athanasopoulos highlight that this does not only include statistical data like time series but also the expertise of people who collect the data and will be using the forecasts. Especially in cases where historical data is scarce, the background of the data generating forecast needs to be understood even better to produce reliable forecasts. On the other hand, they point out that it is sometimes reasonable to use only the most recent part of the data in a time series and thus intentionally reduce the amount of data. This action makes sense, e.g., if older data is not representative anymore due to structural changes in the system to be forecast.

Including it, nevertheless, could then potentially introduce some incorrect bias to the forecasting model.

Step 3: Exploratory Data Analysis

During the Exploratory Data Analysis, often referred to as EDA, the forecaster would start having a closer look at the data to see whether any reoccurring patterns or other characteristics can be detected. The motivation behind this is that it is these features which we need to account for in our forecasts and that further also drive our model selection. There are several methods and tools from which the forecaster can attempt the EDA. For the sake of this thesis, I will focus on using time plots, time-series decomposition, and ACF plots in combination with stationarity as a closely related concept.

Time plot

The EDA's first step is to plot the data in a graph to get a first feeling for the components like trends, seasonality, or cyclic behavior. Hyndman & Athanasopoulos thereby define a trend as a long-term increase or decrease in the data. Often, we would think of a linear trend here, but in general, a trend does not have to be linear and can, e.g., also change direction from an increasing to a decreasing behavior. Seasonality is further described as a reoccurring dynamic of a fixed length affected by seasonal factors like time of the year or day of the week. Lastly, the authors define a cycle similar to seasonal effects. The critical difference is that cycles do not follow a fixed frequency and usually stretch over a more extended period of at least two years. Thus, Hyndman & Athanasopoulos state that cyclic behavior is usually due to economic conditions instead of being affected by seasonal factors.

An exemplary plot of the monthly sales of antidiabetic drugs in Australia is illustrated in Figure 1 (Hyndman & Athanasopoulos, 2021).[1] From this, we can see how a first intuition about the general dynamics of the series and the dimensions and length, can be won.

[1] In the book, all plots were created in R. As my thesis focuses on time-series forecasting in python, the plots were recreated in the respective environment, mainly using the *statsmodels* library by (Josef Perktold et al., 2023).

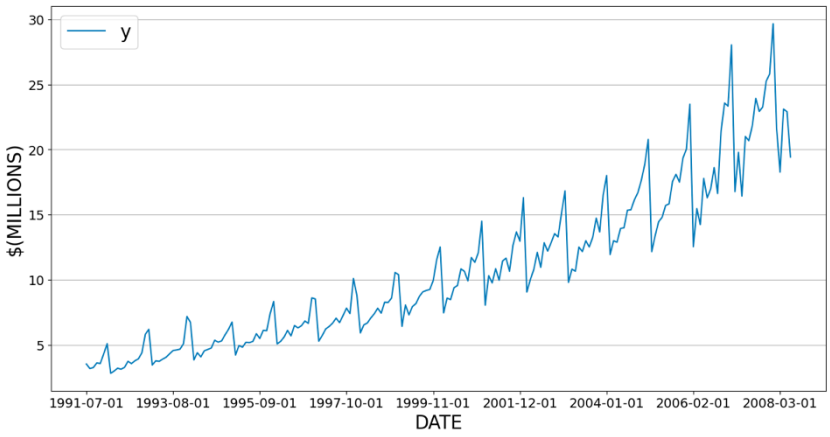


Figure 1: Exemplary time plot for Monthly Sales of Antidiabetic drugs in Australia

Time-series decomposition

In most cases, plotting the data is accompanied by a time-series decomposition. This method, again described in Hyndman & Athanasopoulos (2021), aims to split a time series into several components, each representing one of the pattern categories as mentioned earlier. Trend and cycle are thereby combined into a trend-cycle component. Further, a decomposition features a remainder component that captures every other dynamic in the time series that can neither be explained by the trend-cycle component nor by the seasonality in the data.

We can choose between additive and multiplicative decomposition. If we decide for an additive mode, the decomposition can be written as

$$y_t = S_t + T_t + R_t \quad (5)$$

with y_t being our data, S_t the seasonal component, T_t the trend-cycle component and R_t the remainder component – all at period t. This mode is suitable in cases where the magnitude of trend-cycle and seasonal effects does not vary over time. If, on the contrary, a dependency of the effects' magnitudes to the level of the time series can be observed, a multiplicative mode might be more suitable. The multiplicative decomposition can be written as

$$y_t = S_t \times T_t \times R_r \quad (6)$$

using the same definitions as before.

The authors further explain that as an alternative to a multiplicative decomposition, the data can first be transformed (e.g., by using a log transformation) to remove the variation over time and then decomposed in an additive manner.

Even if it does not reveal its importance at first sight, it is often a very insightful task to look at the decomposition's remainder component. Optimally, it turns out to be white noise,

meaning it behaves like a standard-distributed random variable independent of time and has an autocorrelation close to zero. According to the authors, this implies that all relevant data patterns are already captured within the trend-cycle and seasonal components, requiring a fairly simple model. But with real-life data, we instead often observe the remainder component to be different from white noise. This difference implies that there is relevant information in the data that could neither be accounted for in the trend-cycle component nor the seasonal component, thus requiring a more complex model.

An exemplary decomposition of the series from the previous step can be seen in Figure 2 (Hyndman & Athanasopoulos, 2021).

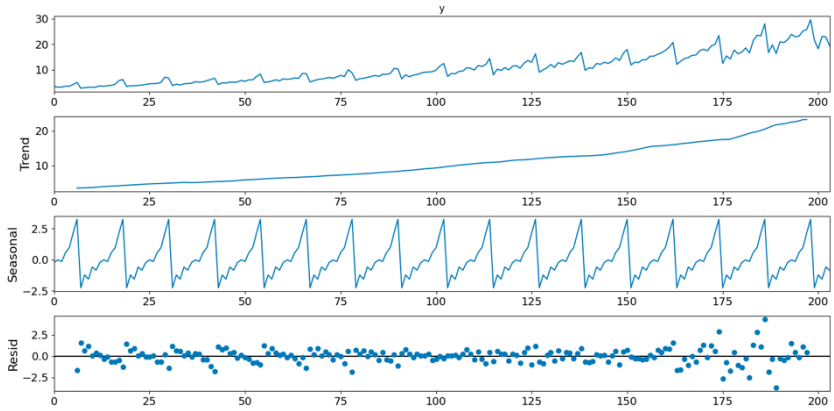


Figure 2: Exemplary decomposition for Monthly Sales of Antidiabetic drugs in Australia

In contrast to Figure 1, the components can be observed separately now, which allows for a closer and more focused inspection.

The plot was thereby created using classical decomposition which is the originator of decomposition methods. But while classical decomposition is still widely used, it is also known to have some issues, e.g., not being able to account for changes in the seasonality over time (Hyndman & Athanasopoulos, 2021). For this reason, more recent techniques have emerged over time, like STL (Cleveland et al., 1990), X-11, or SEATS (Dagum & Bianconcini, 2016).

ACF plots & stationarity

Another tool to gain insights into the time series under inspection is the autocorrelation function or ACF. Following again the definitions in Hyndman & Athanasopoulos (2021), for each lag k, the ACF measures the autocorrelation between y_t and y_{t-k}, which is called an autocorrelation coefficient. We can plot the ACF in what is sometimes called a correlogram to show all autocorrelation coefficients up to a given number of lags. If an autocorrelation coefficient lies outside $\pm 2/\sqrt{T}$ with T being the length of the time series, we refer to the lag as

significant. Typically, the ACF plots include some visual feature to illustrate the boundary between insignificant and significant values.

According to the authors, in case of a trend, the ACF shows a pattern of large, i.e., significant, positive values that decay over time. When data is seasonal, on the other hand, the measured autocorrelation will be more significant for seasonal lags than for other lags. An illustration of the ACF in the case of both strong trend and seasonality is shown in Figure 3 (Hyndman & Athanasopoulos, 2021).

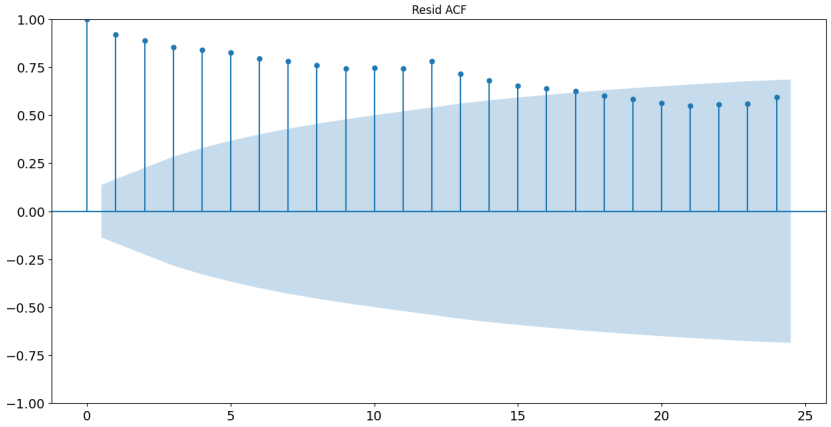


Figure 3: Exemplary ACF plot for Monthly Sales of Antidiabetic drugs in Australia

But the ACF plot not only provides insights on whether a trend or seasonal component might be present in the data, it also provides evidence on whether the time series is stationary. In Hyndman & Athanasopoulos (2021), stationarity is thereby characterized by the time independence of its statistical components; in other words, it will have no predictable patterns in the long term. This, in turn, also implies that a series containing seasonality and/or a trend is not stationary. In the case of a stationary time series (e.g., white noise), we expect the values of the ACF to be mostly insignificant, i.e., close to zero. To be more precise, the authors state that if substantially more than 5% of spikes are significant, the time series is probably not stationary. Several methods can transform a non-stationary time series into a stationary series. One common method mentioned by Hyndman & Athanasopoulos is differencing. Here, changes in the level of the time series get removed, which ultimately will result in eliminating or at least reducing the trend (and/or seasonality) in the series.

The result of first-order differencing is a series that captures the change between consecutive observations and can be written as

$$y'_t = y_t - y_{t-1}. \quad (7)$$

The authors note here that it is impossible to calculate a difference for the first observation in the series and that, consequently, the differenced time series is shortened by one entry.

In case of second-order differencing, which is sometimes needed if the already differenced data still is not stationary, we get

$$y''_t = y'_t - y'_{t-1}. \quad (8)$$

Again, the time series will contain one less entry after the differencing operation, which ultimately adds up to two fewer entries than the original series.

This can be repeated arbitrarily, but the authors point out here that the order of differencing would usually not be chosen higher than $d = 2$.

Further, seasonal differencing can be applied to eliminate seasonality from the series. Here, we take the difference between an observation and the previous observation from the same season (e.g., the same month from a year ago in the case of monthly data). We can write the result of first-order seasonal differencing like

$$y'_t = y_t - y_{t-m} \quad (9)$$

where m is equal to the periodicity our data.

Step 4: Model Selection and fitting

The next step is finding an appropriate model for our forecasting task. As stated in the book, the choice depends on several factors. Possible factors include the availability of data, the strength of the relationship between the predicted and the explanatory variables, and finally, the way the forecast is to be used. Typically, two or three models are compared with each other. Each model has its own parameters and set of assumptions that need to be defined individually based on the task and the given dataset.

The authors differentiate between two high-level groups of forecasting methods: qualitative and quantitative. Which one to choose thereby highly depends on the availability of data.

Qualitative forecasting

If no (relevant) data is available, forecasting happens in a qualitative manner. Thereby, the results are not based on guesswork but on well-structured reasoned approaches and frameworks capable of generating good forecasts, even without using historical data. The authors refer to

this as judgmental forecasting and state that it has become common practice now, e.g., in cases where new products are being launched, and there is thus no historical data available. But to leverage the advantages of this approach, such as flexibility and adaptability, practitioners need expert domain knowledge and to stay up to date on domain-relevant information. If both factors are present, judgmental forecasting can, despite its limitations of, e.g., being subjective, improve forecast accuracy significantly in case of limited data availability. Hyndman & Athanasopoulos further point out that judgmental forecasting can also complement or correct statistical approaches in cases with more available data.

Quantitative forecasting

The other high-level group of methods is quantitative forecasting. According to Hyndman & Athanasopoulos, two conditions must be satisfied to apply these. First, historical data must be available and in numerical form. Secondly, it must be reasonable to assume that some of the patterns in the past data will be relevant for predicting the future, i.e., they will (partially) continue.

If these conditions are met, there is a broad portfolio of quantitative methods, ranging from conventional Statistical methods to more advanced Machine Learning and Deep Learning (ML/DL) approaches, each having its properties, strengths, and weak spots, as illustrated in Figure 4.

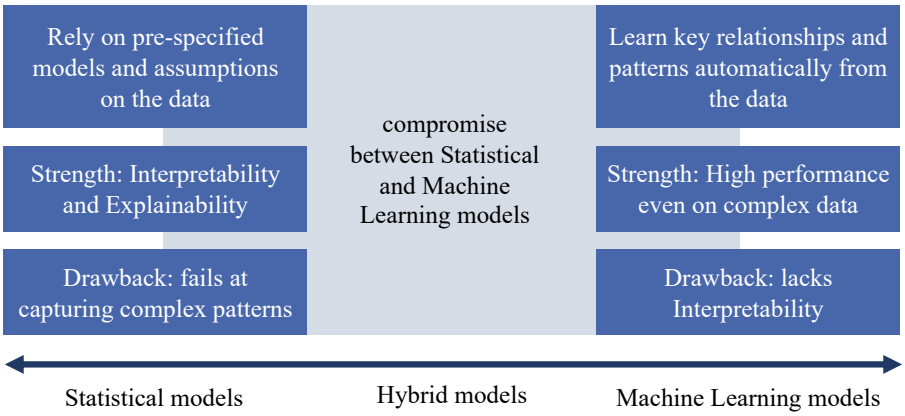


Figure 4: Spectrum of quantitative forecasting models

According to sources like Mills (2019) or Bergmeir & Benítez (2012), the main difference between the two approaches is the degree to which they rely on existing models and assumptions versus the ability to learn patterns and relationships directly from the data. Statistical models derive their forecasts by relying on theoretical assumptions about the underlying relationships in the data and modeling them using a specific set of equations or mathematical functions. This results in highly interpretable models which can be used to gain

insights into the underlying dynamics of the time series. The drawback, on the other hand, is that due to these often simplified assumptions, the models lack the ability to capture more complex patterns in the data. ML/DL methods, on the other hand, aim to learn the underlying patterns and relationships in the data automatically without relying on pre-specified equations or assumptions. While this characteristic boosts the performance of ML/DL models in extracting even complex patterns from the data, it often leads to reduced interpretability. Due to this, ML/DL models are often referred to as "black-box" models (Mills, 2019; Bergmeir & Benítez, 2012). The result is a trade-off between interpretable models on the one hand and high-performing models on the other hand. A compromise to this is hybrid models, which aim to balance both approaches, optimally combining the best of both directions into one.

Components in quantitative models

But despite the range of methods available for quantitative forecasting, some reoccurring features and components can be observed that various methods across the spectrum have included in their architecture.

The first of these components is **trend,** i.e., trend-cycle. Most models that practitioners work with have some mechanism to account for trend in the data as one of the key characteristics present in time-series data. The way of doing that can thereby differ greatly across methods. While some models like *ARIMA* aim to eliminate the trend from a series to model the data in other terms (Hyndman & Athanasopoulos, 2021), other models explicitly try to fit a trend line – or in some cases, even try to fit several piece-wise trend lines like in the *NeuralPropet* model (Triebe et al., 2021).

The next widespread component is **seasonality**. Again, there is a broad portfolio of methods to account for seasonal patterns in the data. As for the trend component, there is often the option to reduce or even remove the effects of seasonality from the data. On the other hand, we see more advanced methods often use Fourier terms to model seasonality (Triebe et al., 2021).

Another component often used in statistical models is Autoregression (**AR**). The AR component thereby is introduced to capture how past values determine the value of future predicted values. In other words, an autoregression is a regression of the variable against itself.

Thus, Hyndman & Athanasopoulos define an AR model of order p as

$$y_t = c + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \cdots + \emptyset_p y_{t-p} + \varepsilon_t \quad (10)$$

with c being a constant, i.e., the level of the series, $y_{t-i} \mid i \in \{1, \dots, p\}$ being the lagged values of y_t, $\emptyset_i$ being the respective weights for each lagged value and ε_t being the white noise error term for the current prediction at time step t. The AR component is one of the key components used in the *ARIMA* model. Still, due to its explanatory capabilities, it is also used in more novel advanced approaches like *NeuralProphet*.

The counterpart to the AR component is the Moving Averages (**MA**) component. Contrary to the AR model, the MA model uses past forecasting errors to predict future values. Thus, Hyndman & Athanasopoulos explain it as a weighted average of past forecast errors. A formal definition of the MA model of order q is

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \quad (11)$$

with $\varepsilon_{t-j} | j \in \{1, \dots, q\}$ being the observed error values from the past periods and θ_j being the respective weights for each predictor. c and ε_t are defined the same way as before, which means, in turn, that we do not observe ε_t.

In some models, we have components that allow us to include covariates. These exogenous variables affect the behavior of our target variable on top of the historical data on the target variable itself. If a model allows to include such variables, this is called multivariate models. If such components are included, it is often differentiated between past and future covariates. Whereas for past covariates, we only have historical data available (e.g., competitor activities), in the case of future covariates, both past and future values have to be known (e.g., national holidays) (Triebe et al., 2021). For this thesis, I will mainly focus on using **past covariates** but will provide an overview of the inclusion of future covariates, nevertheless, where applicable.

More recent approaches to forecasting time series often fall under the category of (deep) neural networks. The basic idea behind neural networks is to mimic the human brain's behavior when processing information. Just like our brain, a neural network is composed of multiple interconnected nodes (called neurons) organized in layers. A neural network with more than two layers (i.e., an input and an output layer) is also called a deep neural network. The intermediate layers are then further referred to as hidden layers. Each neuron receives an input that can either come from other neurons or external sources, processes this input utilizing an activation function, and produces an output, which is then passed on to other neurons or used as the network's final output. Thus, using many interconnected neurons, the network can learn complex patterns and relationships in the data that are difficult or impossible to capture using

conventional methods. By now, neural networks are effective in various applications, such as image classification, natural language processing, and time-series forecasting (Aggarwal, 2018). Thus, one feature which is often included in novel forecasting methods is **hidden layers**. Mainly, these stem from the class of recurrent neural networks, as they were explicitly designed to capture temporal relationships.

One even more advanced concept is called the **self-attention** mechanism. With the introduction of Transformer models (Vaswani et al., 2017) as a special class of neural networks, self-attention has become very prominent within the field of Natural Language Processing ever since. Attention mechanisms have thereby proven themselves the ability to pick up long-term relationships that may be challenging for recurrent networks to capture. Furthermore, it also has the advantage of increasing the interpretability of neural networks, which is a typical challenge in deep learning methods.

Another useful feature of some forecasting models is the ability to provide **quantile forecasts**. This means that a model can also provide prediction intervals instead of producing single-value predictions. This feature can be helpful to account for best-case and worst-case scenarios in the forecast (Triebe et al., 2021). Typically, this is done by predicting multiple percentiles simultaneously at each time step.

One last rather rare feature is the ability to deal with or even leverage **multi-channel** data. Often, time-series data comes with static information, e.g., about what region the data stems from or which other channels it can be attributed to. This information can be used to group the data and provide analyses and forecasts on the channel level. Still, only a few models can provide individual forecasts for each channel within the data in one go. Nevertheless, it is a desirable feature, especially if the model can leverage hidden information from the channels' interaction, as seen in the Temporal Fusion Transformer (Lim et al., 2019).

An overview of the described components and features is given in Figure 5.

Trend	MA	Self-Attention
Seasonality	Past Covariates	Multi-Channel
AR	Hidden Layers	Quantile Forecasts

Figure 5: Selection of components in quantitative models

Step 5: Application and Evaluation of Forecasting Models

Finally, after selecting the models, the forecaster can train them and produce forecasts. In a real-life scenario, all (relevant) data up to the time the forecasting period starts would be used to train the model to ensure we capture the most recent dynamics in the data. The performance of a model can then only be adequately evaluated as soon as the data for the forecasting period has become available (Hyndman & Athanasopoulos, 2021).

In experimental setups like this thesis's case, we would split the available data into a test and train set. The models would then be trained only on the train set, while the test set, often also referred to as hold-out data, would then be used to evaluate the forecasting performance. By this, we ensure that our model can generalize well on new unseen data points instead of simply memorizing the given training data (Hyndman & Athanasopoulos, 2021). According to the authors, a common rule of thumb would be to use the most recent 20% of the data as the test set and to use the remaining 80% to train the model. The least length of the test set should further ideally be as large as the required forecast horizon.

Jerez & Kristjanpoller point out that it often makes sense to have a third split of the data when it comes to ML/DL approaches, which typically include a much higher number of hyperparameters to choose from. This split is then referred to as the validation set and used to fine-tune the model configuration to improve its ability to predict out-of-sample data. In this setup, the model is then trained on the train set, comprising the least recent data, validated on the validation set. This optimizes the model's generalizability on the following periods and is ultimately tested on the test set. According to the authors, the validation set's length should be chosen so that it is diverse enough to contain relevant information that could occur similarly in the test set. Thus, ideally, the data in the validation split is realistic enough to always lead to the optimal configuration of the model to generalize onto the unseen future (Jerez & Kristjanpoller, 2020).

An illustration of the data splitting approach, optionally including a validation split, is shown in Figure 6.

Figure 6: Exemplary data splitting with optional validation split

There exist also more sophisticated validation variants, e.g., cross-validation. In this procedure the train data is split again, but this time by moving a fixed-size training window through the data. The model is then trained on each window and evaluated on subsequent (validation) time periods. By averaging over the retrieved evaluations, cross-validation allows for a more accurate evaluation of the model's generalizability (Hyndman & Athanasopoulos, 2021). Nevertheless, it might not be as relevant in cases where the amount of data is relatively low which is why its value for this thesis might be limited.

We further measure the forecasting quality of a model in terms of errors. Following Hyndman & Athanasopoulos (2021), these errors capture the difference between the observed and the predicted value, which can be written as

$$e_{T+h} = y_{T+h} - \hat{y}_{T+h|T} \quad (12)$$

where the training data is given by$\{y_1, \dots, y_T\}$, the test data is given by $\{y_{T+1}, \dots, y_{T+h}\}$ and the predicted value based on the training data is denoted as $\hat{y}_{T+i|T} | i \in \{1, \dots, h\}$[2]. The lower the error, the better the ability of the model to capture the key dynamics of the series without overfitting on noise in the data.

Commonly used errors described in the book are scale-dependent errors like the mean absolute error (MAE), the mean squared error (MSE), and the root mean squared error (RMSE), as well as percentage errors like the mean absolute percentage error (MAPE). Which error type works best depends on the forecasting task itself and especially whether we need to compare the model performance between datasets (Hyndman & Athanasopoulos, 2021).

3 Methodology

This chapter covers the methodology of the thesis which was used for the experimentation approach. Following the process structure outlined in Chapter 2.2, it starts with the problem definition and description of the data collection in Chapter 3.1. Chapter 3.2 further provides an overview of the forecasting approach that is currently used at Growblocks. After that, Chapter 3.3 continues with the EDA which further includes a definition of distinct dataset groups in order to allow for a more customized selection of models. Finally, Chapters 3.4 and 3.5 focus

[2] In case of a validation set, the train set would instead be given by $\{y_1, \dots y_{T_{train}}\}$ while the validation set would be given by $\{y_{T_{train}+1}, \dots, y_T\}$. Note that the predicted value would further be defined as $\hat{y}_{T+h|T_{train}}$, if training would happen purely based on train data set.

on the model selection. Thereby, Chapter 3.4 provides an overview of related research to get an understanding what models were used in similar use cases. Chapter 3.5 then introduces the final model selection and describe the experimentation approach that was pursued for each model.

The information presented in Chapter 3.1 and 3.2 thereby stem from several interviews conducted with Growblocks and my own experience working with their data and models.

3.1 Problem Definition and Data Collection

In the first part of this chapter, I want to provide an overview of the problem definition and data collection steps of the forecasting workflow.

Problem Definition

The first and most important questions here are about what the forecasts are used for and who uses them. According to interviews with Growblocks, the primary stakeholder in this use case are commercial leaders such as the CEO, CRO or COO. They use the forecasts to get an understanding of the development of the business performance over the next year as well as for the sake of optimizing the way the company generates revenue. This optimization could refer to, e.g., hiring plans, campaign planning, reducing friction between the different stages of the sales funnel but also increasing the efficiency within the teams at each stage. The forecasts are further used as a bottom-up counterbalance to the top-down revenue target from the C-level or board, which often is not realistic to be met by the commercial teams in charge of driving the revenue (Growblocks, personal communication, 2023).

Growblocks emphasized to use a forecasting horizon of one year, i.e., 12 time steps, as typically, companies would also plan their top-down targets for the whole year ahead.

From the interviews, I found that the specialties of this forecasting task in the case of start-up and scale-up companies are three-fold, with the first two arising from the fact that such companies have not been doing business for long.

The first challenge is that they usually have only very limited data available. Typically, companies would not spend significant efforts implementing a RevOps function before entering a stage where the commercial teams have reached a certain size so that first processes can be standardized and sales can maybe even start to be scaled. Thus, Growblocks can generally expect the data they receive to be of a certain maturity grade and usually contain a minimum of 12 months of data. It is still relatively new companies, meaning they typically will

not see datasets spanning significantly more than 8-10 years, i.e., more than 120 periods (Growblocks, personal communication, 2023).

Secondly, one specialty of the forecasting task of start-up and scale-up revenues is the instability of many time series. As these companies are still in an early stage of development, processes change constantly, and new ways of working and selling evolve. This is typically also reflected in the time series. In mature companies, it is more likely to see relatively stable revenue generation and growth, as they might already know which channels work for them and how they can serve them. In the case of start-ups and scale-ups, it is instead often more volatile and sometimes even features structural breaks, e.g., in case processes undergo far-reaching restructuring initiatives. Due to this, some interviewees pointed out that it can potentially be reasonable not to use the full series, as older data points might not be representative of the current sales funnel (Growblocks, personal communication, 2023).

The last challenge is the customer companies' heterogeneity. This might seem obvious at first, but as businesses are different, it is important to acknowledge and account for this diversity. Even though, the two challenges described above are valid for the use case in general, their intensity might differ between datasets. Not only will we see varying lengths of revenue time-series for different companies, but we will also see various dynamics among them. This can be due to external reasons, but also depends on how they operate their business internally. If companies operate their revenue engine in different ways, it will also reflect in their respective revenue time series. Thus, it might not be feasible to find a one-fits-all solution to forecast revenue of Growblocks' customers. Instead, a more granular approach is needed that allows to account for different types of customers and their specific requirements when generating the forecasts for them.

Bringing these challenges of limited data, volatility, and heterogeneity together, Figure 7 provides a simplified overview of the different categories of customer datasets that can be seen at Growblocks.

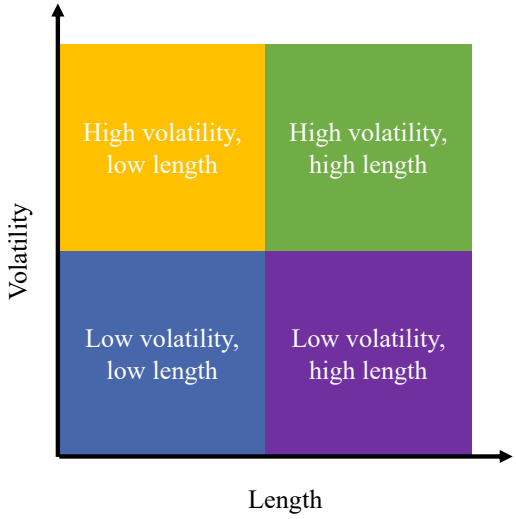


Figure 7: Categories of time series of start-ups and scale-ups in B2B SaaS

Data Collection

The collected data comprises time series on six customers of GB, sourced from designated systems used for web analytics (primarily Google Analytics) and CRM (e.g., Hubspot or Salesforce). After that, each dataset underwent a preprocessing where it was cleaned and transformed to match a standardized format. The resulting datasets include time series on the target variable (*WON_REVENUE*) and other available metrics along the funnel. The main ones to mention here are *USER_COUNT*, *CREATED_LEADS*, *CREATED_OPPORTUNITIES,* and *WON_CUSTOMERS*. All time series contain data points aggregated on a monthly level. Further, each series is grouped by regions and traffic or attribution channels.

3.2 Current Forecasting Approach

The forecasting approach currently used at GB can be categorized as a compromise between a qualitative and quantitative approach.

The aim is to model the Sales Funnel, including the respective dynamics and characteristics. As mentioned before, these can differ greatly between companies. Thus, in order to provide a model that is customized to the specific requirements and characteristics, Growblocks typically conducts an EDA to extract main characteristics and patterns. The forecasting model then serves to simulate the effect of future initiatives on the generated revenue at the end of the funnel. Figure 8 shows a schematic overview of the model architecture.

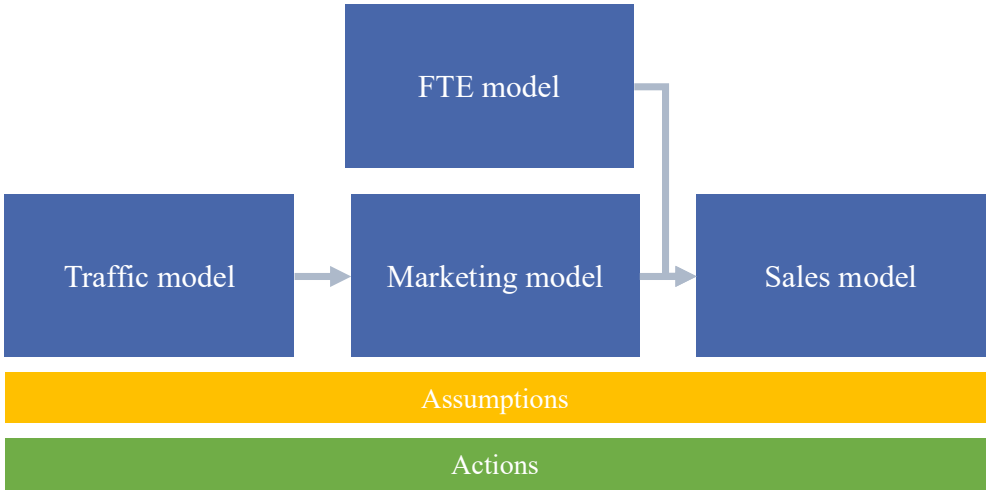


Figure 8: Current modelling scheme at Growblocks

The model contains three high-level building blocks. The first is the baseline components, comprising the Traffic, Marketing, FTE (short for full-time-equivalent), and Sales model. These components are used to define the framework and the critical dynamics of the funnel. In the Assumptions block, additional knowledge on the funnel can be encoded. On the one hand, this includes value corrections, but it can also be used to add information on, e.g., seasonal behavior.

With the baseline components and the assumptions in place, a baseline projection can be derived, i.e., the projected development of the metrics, without taking additional actions to drive revenue. The last building block is the planning engine in the Actions block and partially in the FTE model. In order to move from the baseline projection to the plan projection, we leverage this block by accounting for the effects of planned initiatives like marketing campaigns, new sales hires, and sales training.

Note that the model would typically also look at the revenue generated from the existing customer base in addition to revenue from new business. However, for the sake of this thesis, I decided to only focus on the new business side, i.e., any revenue generated from newly acquired customers.

It is further to be noted that the model can also be more granular by, e.g., differentiating between different regions or products within the flow. Nevertheless, in order to provide a high-level overview of the key mechanisms of the model, this aspect will be left out for the sake of simplicity.

In B2B SaaS, we see two main motions flowing through this system, namely the Inbound and the Outbound motion, which will be described in the next part of the chapter. For some

companies, we can additionally observe other motions, e.g., Product-Led-Growth (PLG) or Partnerships. Nevertheless, for this thesis, I will not dive deeper into those. After describing the two motions in the following, I briefly cover the knowledge encoding which happens in the Assumptions block as well as into the planning engine of the model. Finally, the chapter ends with a summary of the advantages and disadvantages of the current forecasting approach at Growblocks.

Inbound

The Inbound motion is driven through marketing initiatives and includes all activities where the customer initiates the first contact. In a blog post for the CRM software provider *Hubspot*, former CRO Mark Roberge defines it as "a modern approach that focuses on providing customers with value before pitching your product or service" (HubSpot, 2023). He describes the process to typically start with customers learning about and engaging with the business and its brand through various marketing efforts. At one point, a sales team will take over to further guide the lead through the sales process. According to the experiences of practitioners at Growblocks, Inbound leads are typically easier to convert into customers than Outbound leads, as they are usually known to have a general interest in the product already (Growblocks, personal communication, 2023).

Model-wise, the Inbound motion starts in the Traffic model. Here, user counts, i.e., the number of unique website visitors, are tracked for different traffic channels. The traffic channels thereby describe the source through which a visitor has found the website. Examples of such channels are direct traffic, paid ads, or email. Further, the model tracks each channel's goal completion rates for the respective period. Typically, these goals are activities like requesting a demo or trial, but depending on their business model, companies can also define other individual goals to keep track of. The result of the Traffic model is the number of users who have completed the respective goals.

In the next stage of the model, these users will (partially) turn into leads. This happens in the Marketing model. What percentage of the completed user count turns into leads thereby depends on the lead definition of the respective company, but in many cases, it happens to be a 1:1 mapping (Growblocks, personal communication, 2023). In the Marketing model, leads are no longer grouped by traffic channels but by their attribution channel. Often, these are defined analogously to the goals from the traffic model, meaning we might now have a group of leads created based on a demo request and another created based on a trial request.

The next step tracked in the Marketing model is the qualification of marketing-sourced leads, i.e., the conversion of created leads to marketing qualified leads (MQLs), which is typically bound to some pre-defined qualification criteria. After that, the model tracks how many of these MQLs are converted into an opportunity and will thus move into the Sales pipeline.

Finally, the number of marketing-sourced opportunities moves over to the Sales model. The Sales model is the last stage, which tracks the conversion of opportunities into customers and the associated generated revenue. Thus, at the end of the model, we get the won revenue for each attribution channel within the inbound motion.

To derive the baseline projection, Growblocks applies different heuristics. They typically calculate the average for the preceding three periods to estimate future conversion rates, (initial) user counts, and the ACV. The total outcome of conversion is typically calculated through a simple multiplication operation. Nevertheless, in case of the conversion from leads to opportunities as well as from opportunities to won customers, sales cycles are considered within the operation. This means that instead of assuming all created opportunities or customers convert within one period, we expect the conversions to be distributed across several periods where some leads or opportunities take longer to convert than others. This method is similar to an AR process where the degree is equal to the maximum length of the sales cycle (in months).

An overview of the flow within the Inbound motion is shown in Figure 9.

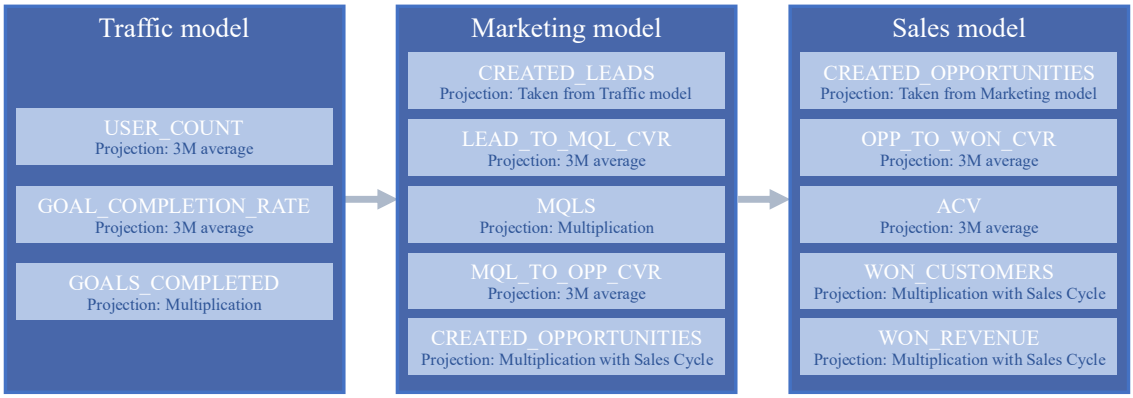


Figure 9: Flow in the Inbound motion

Outbound

The Outbound motion, on the other hand, is based on outreach which is typically done by a Sales Development team. This team keeps looking for promising leads and reaches out through, e.g., cold calling or direct email to book meetings with them. While it has been the standard way of approaching leads for a long time, Outbound is sometimes said to be dead today. However, as described in a blog post by *Cloudtask* CEO Amir Reiter, the Outbound motion

remains to have some unneglectable strengths, which still render it inevitable for B2B businesses today. For example, Outbound allows for a much higher-targeted outreach than Inbound and leverages personal interaction with the leads right from the beginning (Reiter, 2021).

Other than the Inbound motion, the Outbound motion starts in the FTE model. The model tracks the monthly number of Sales Development Representatives (SDRs) based on each SDR's start and sometimes end date. In addition to the headcount, the system tracks the respective meeting targets, i.e., the number of sales meetings each SDR should book where they, e.g., give a product demo. This number then translates into the number of created opportunities through a 1:1 mapping from the meeting targets. After that, the number of created opportunities is again taken to the Sales model, falling into the Outbound attribution channel. The remaining process is then analogous to the Inbound motion, with the result of retrieving the won revenue that can be attributed to the Outbound motion.

One can include attrition on the number of SDRs in the FTE model for the projected values. Within this number, the model accounts for potential leavers in the future by decreasing the headcounts and the resulting number of created opportunities over time. The projection of the metrics in the Sales model follows the same logic described for the Inbound motion.

Figure 10 illustrates the flow of the Outbound motion.

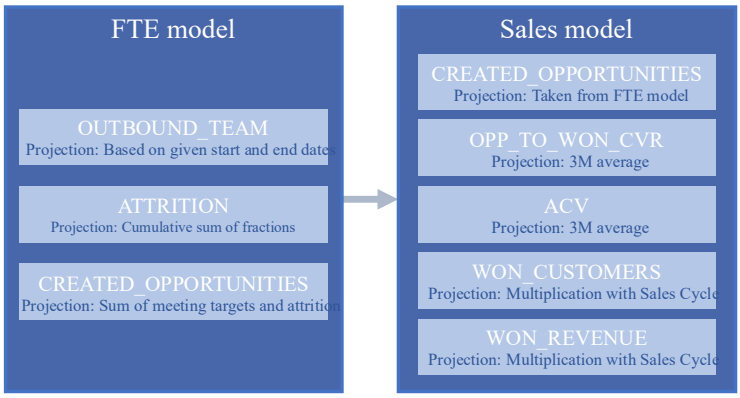


Figure 10: Flow in the Outbound motion

Knowledge encoding

Often, the basic setup of the model is insufficient to capture all critical dynamics of the funnel. Thus, the Growblocks approach includes a mechanism to encode additional information in the model. The Assumptions block can take overriding values for all available metrics in the model, i.e., if a value is available in the Assumptions block, the model will always prefer it over the calculated projection value. Typically, we would not use assumptions on the historical

values as it could distort the reporting. Still, there are some situations where it can make sense, e.g., in case we have knowledge about data quality issues in a certain period which we do not want to affect our model negatively (Growblocks, personal communication, 2023).

Apart from value corrections, the Assumptions block can be used to include information about known characteristics of the series, such as seasonality. As many projection heuristics are based on averaging immediately preceding values, we will not see recurring patterns emerge without explicitly accounting for them.

Still, it is important to note that the value of the Assumptions block depends entirely on the forecasters' knowledge, i.e., an effect needs to be known to account for them in the assumptions. This is not always trivial, especially when it comes to the task of assessing the magnitude of an effect.

Planning engine

The last key mechanism within the Growblocks approach is the planning engine. The planning engine is the heart of the forecasting model as it allows the end user to test different funnel-related actions concerning their overall impact on the resulting revenue. That way, the user can create and compare plan scenarios to figure out the optimal growth strategy for the business.

Thereby, the logic behind the module generally is to add some action that leads to an increase of one or more metrics in the funnel. As a consequence, the downstream funnel is (positively) affected with an increased revenue as the final outcome of the respective action.

The two most common types of actions are marketing and sales hiring initiatives. In the case of marketing initiatives, we aim to increase the generated Inbound Revenue. This increase can happen, e.g., through launching a marketing campaign which would typically boost the number of users who become aware of the company. Other initiatives can also target CVRs by incentivizing a conversion in some way. In both cases, the action would be added to the Actions block with the expected increase added for all affected periods.

The second common type of action is sales hiring. This time, the Outbound motion is stimulated by increasing the number of available SDRs and, thus, the number of created opportunities. In contrast to the marketing initiatives, hirings are added directly to the FTE model with the respective start date of the planned hires lying in the future. Nevertheless, we can also target other metrics in the Outbound flow, e.g., the CVR to closing a deal by organizing sales training, which would then again be added to the Actions block.

Similar to the Assumptions block, the total impact of the effect relies on assumptions made by practitioners within the company. Therefore, they should have a profound understanding of the dynamics within the company's Sales funnel to assess these effects accurately.

Strengths and Weaknesses of the Approach

Overall, from the interviews with Growblocks as well as my own work with the model, the forecasting approach shows to bear both strengths and weaknesses. The main advantage lies in its flexibility and ability to adapt to system and environment changes quickly. Another significant advantage is the explainability of the system. As the approach aims to mimic the behavior of the Sales funnel, all intermittent steps can be observed and monitored. This full-funnel visibility is beneficial if we aim to forecast the funnel output and optimize its behavior by tracking which metrics might lag. A third advantage is the option to enrich the information extracted from the data with expert knowledge. Especially in low data settings, this can be helpful in order to correct for biases in the model (Growblocks, personal communication, 2023).

However, while knowledge encoding can be perceived as an advantage of the model, on the one hand, it can also be argued to be a weakness of the model. The reason for this is that it renders the model to depend largely on the abilities and expertise of the forecasters working with the system (Growblocks, personal communication, 2023). This dependency leads to the risk that hidden patterns that are hard to uncover for humans are left out from the model, which can ultimately lower the model's forecasting performance. This is especially true when it comes to encoding volatility, as the implementation of the model requires the forecaster to understand the movements in the data enough to account for them explicitly.

Further, as the models are usually customized to the specific needs of the customer company, they are more prone to implementation mistakes than standardized solutions. This issue becomes even more severe with an increasing size of the model (Growblocks, personal communication, 2023).

3.3 Exploratory Data Analysis

In the following, I look at the six available datasets to find groups among them based on shared patterns and characteristics. After that, one exemplary dataset from each group is inspected more closely to understand the specific characteristics.

Overall Comparison and Group Definition

We start by plotting all six datasets next to each other. The result is shown in Figure 11.

Figure 11: Won revenue plot for the six available datasets

The first axis in the figure displays the time dimension, while the second axis shows the won revenue for each month. Both axes are shared across all subplots to allow for more comparability. The first thing that can be observed is that the time series differ in length. C1, as the shortest series, covers only 24 months. i.e., two years of data. In contrast, all other datasets cover more than four years, with the most extensive dataset C4 containing 101 data points ranging over more than eight years.

Looking more closely, a first feeling for some main characteristics can be won. When it comes to trend, we can observe an increasing trend in C2 and C3. For C4 and C5, there seems to be some level of trend, even though it is more subtle here. The same goes for C1, but with the limitation that it is much harder to judge here due to the short length of the series. For C6, it starts relatively flat and only picks up for the most recent two years of the data, primarily driven by an unexpectedly large spike in the end.

The most striking feature in the plots is the differences concerning the variances of each dataset. While only minor fluctuations are observable in C4 and C5 apart from a few outliers, C2, C3, and C6 show comparably strong fluctuations. C1 appears to be ranging somewhere in between. Looking further into these variances, some regularity, if not even time dependency, can be observed in the variation for C2, which seems more irregular in C3. These findings are also confirmed when looking at the boxplots, as seen in Appendix 2. In C6, assessing the degree of regularity in the fluctuations is harder. From the time plot, it seems like there might be some

time dependency in the magnitude of the spikes. The boxplot, on the other hand, suggests the spikes to be irregular, probably due to the generally very low level of the series.

Outliers can be detected predominantly in C3, C4, C5, and C6. In C3 and C4, they do not seem to be of high explanatory value and can easily be distinguished from the key patterns. On the other hand, this is much harder in the case of C3 and C6, where it is hard to assess the outliers' informational value for the prediction task from the first look.

To complement the visual inspection of the plots, Table 2 includes some summary statistics on the datasets.

Table 2: Summary statistics for each dataset

Dataset	Number of data points	Mean	Standard Variance	Maximum	75% Percentile	Median	25% Percentile	Minimum

Again, we can see some groupings emerge from looking at the statistical properties. The standard variance confirms the findings from the visual inspection. C4 and C5 show the lowest values ranging below . C1 has a medium standard variance of approx. . Finally, C2, C3, and C6 have the highest values ranging from .

Contrarily, looking at the series level, especially in terms of the mean and the median, draws a different image of having C6 as a separate category with a much lower overall level. This difference is probably because the dynamics in C6 change immensely over time, with the primary growth happening only in the last part of the series. This hypothesis is also supported by the summary statistics when taken only the most recent two years of the data (see Appendix 3). Even if it does not remove this contrast entirely, it can be seen how the level of the series is much closer to those of C2 and C3 when looking only at the most recent two years. This highlights how the grouping of the datasets depends largely on the choice of grouping criteria.

For this thesis, I chose two main criteria for the groupings: the length of a time series and the volatility within the time series. Both are critical challenges in the specific subject of this thesis, as stated in Chapter 3.1. Therefore, they should be targeted explicitly in the experiments.

This results in the following groupings: Group 1 comprises datasets C4 and C5. Both datasets contain a relatively high number of observations while at the same time showing low volatility. The prototype dataset I will focus on from this group will be C4 as it is the longer dataset of the two, which further has an even lower variance. Group 2 contains C2, C3, and C6. All three datasets cover a relatively large range of time while at the same time showing high volatility. C6 was selected as the prototype dataset here. Primarily, this was because Growblocks employees pointed it out as one dataset that was, due to its volatility, especially hard for them to forecast with the existing forecasting approach. Further, it is also the longest series in this group, which might be beneficial when building the models. Finally, Group 3 includes only C1 as the shortest available datasets. Thus, C1 will serve as the prototype for Group 3. Based on the overview of dataset categories that was derived in Chapter 3.1, Figure 12 illustrates where to locate each group within the spectrum.

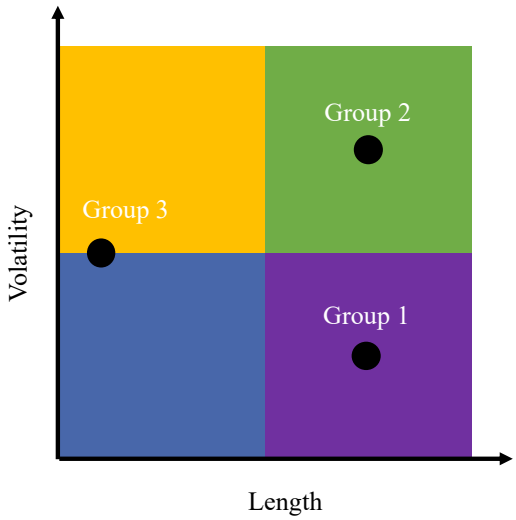


Figure 12: Dataset groups within the available spectrum of time series categories

Group-specific EDA

After deciding on the groups and the respective prototypes, the last part of this chapter is dedicated to understanding the individual dynamics of the three prototypes to find a suitable model. For this, the data is split into train and test sets, as the model definition will only happen based on the train set. The test set thereby covers the last 12 data points of the overall series, equal to the selected forecasting horizon.

Group 1

The first dataset to have a look at is C4. As a first understanding of the series could already be generated in the first part of this chapter, I will immediately dive into the decomposition of the

train series now. This was done using the *seasonal_compose* module from the *statsmodels* library by (Josef Perktold et al., 2023). The output of this is visualized in Figure 13.

Figure 13: Seasonal decomposition of C4

The topmost subplot in the decomposition plot shows the original series. Right below, the trend-cycle component can be seen. While it was still hard to tell from the original plot, it becomes very clear from this illustration that the time series includes a trend component. It mostly shows an increasing behavior even though there are three periods where the trend seems to show a decreasing behavior. Further, the data appears to include some seasonality. But looking at the scale, it seems to be minor in contrast to the remaining components. Lastly, the remainder or "residual" component is displayed in the last subplot. Here, we can recognize the two large values of the outliers from the original series.

Looking closer at the ACF for the residuals (see Figure 14), we can see an almost sinusoidal behavior which could indicate that there is still some seasonality in the remainder component. This can either be due to the reason that it has not been captured entirely in the seasonal component or, again, be an indication that it was only introduced through the decomposition. But as almost all spikes are insignificant apart from one (not taking the zeroth lag into account), this can probably be disregarded. Overall, the remainder component acts mostly like white noise, indicating that the series' key dynamics can potentially be captured with a fairly simple model.

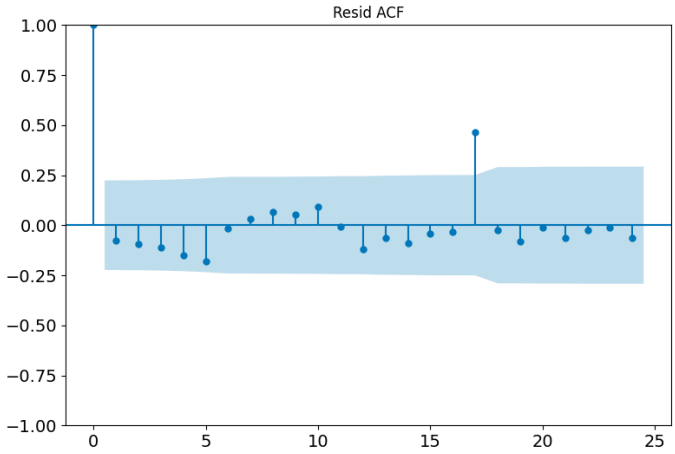


Figure 14: ACF plot for the decomposition residuals of C4

Lastly, the ACF was plotted for the overall series again to understand whether the data is stationary. The plot can be seen in Figure 15.

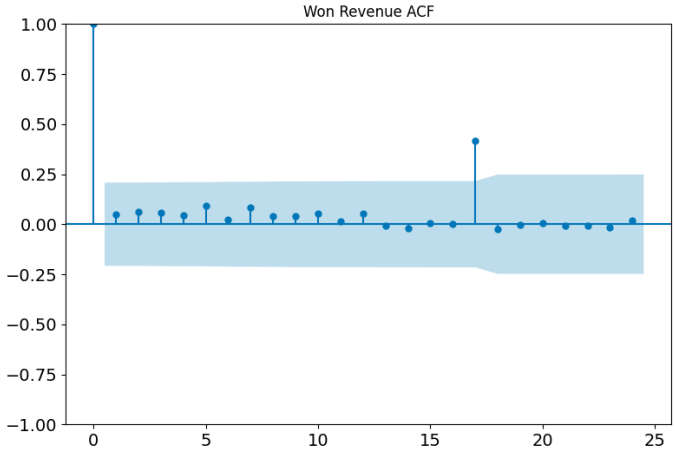


Figure 15: ACF plot for C4

Even though I would have expected some indication of non-stationary after seeing the trend component in the decomposition, the plot implies stationarity. To confirm this feeling, I further conducted an augmented Dickey-Fuller (ADF) test taken from the *statsmodels* library. The result was a test statistic of approx. -8.85 with a p-value of less than 1%, which confirms that the data is already stationary and therefore does not require further transformations. The same result could be achieved when only looking at the most recent two years of the train set.

Group 2

For Group 2, dataset C6 was taken under closer inspection. The decomposition of the train split is shown in Figure 16.

Figure 16: Seasonal decomposition of C6

The first thing that can be noted is that the very large spike at the end of 2022 is no longer included in the series, which already reduces the total variance a lot compared to the full series. Still, the trend-cycle component indicates a behavior similar to what I found in my analysis in the first part of the chapter. In the beginning, the trend stays flat and only starts increasing in the last third of the series. Telling from the scale of the trend-cycle plot compared to the overall scale, it can further be concluded that the trend does not play a predominant role in the dynamics of the series. This also goes for the seasonality component, where the scale maximum is even lower than for the trend-cycle component. The most determining component of C6 seems to be the remainder which ranges from around -50.000 up to +50.000. This makes sense as it is also in accordance with the results from my initial analysis and highlights the complexity of this dataset again. In contrast to my prototype from Group 1, we can see here how the residuals are much more scattered around zero, which is a first indication that some uncaptured information might be left in the residuals. On the other hand, when looking at the residuals ACF plot (see Figure 17), the residuals appear stationary again, which would not support this hypothesis.

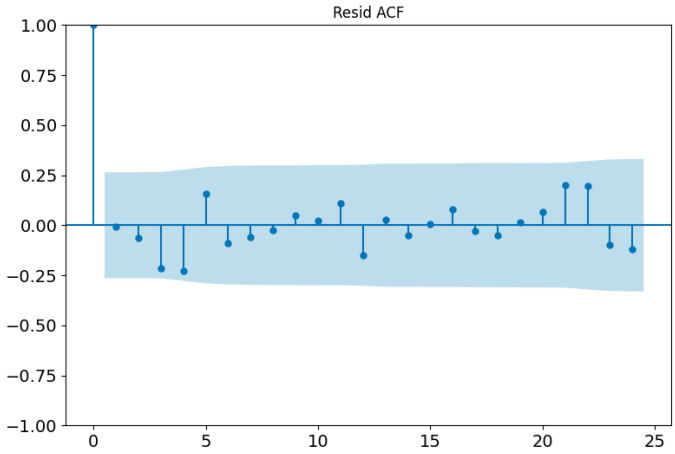


Figure 17: ACF plot for the decomposition residuals of C6

Finally, I checked the stationary of the overall train series. The resulting ACF plot can be found in Figure 18.

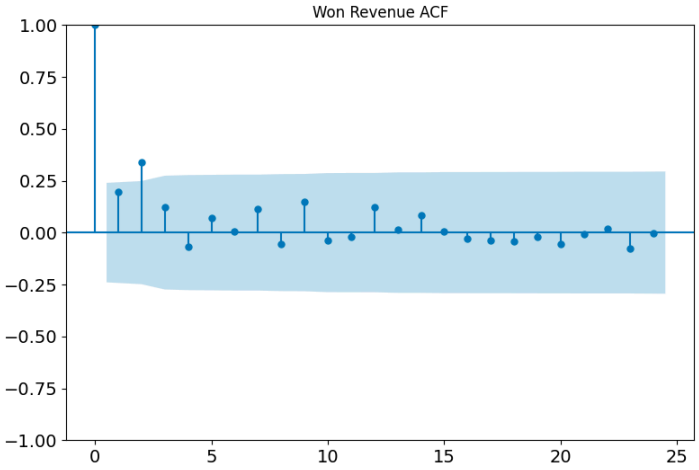


Figure 18: ACF plot for C6

Telling from the plot, we would assume the data to be non-stationary already, but the ADF states the opposite. With a p-value close to 1, the ADF test result emphasize to difference in the series to make it stationary. To render it stationary, both a second-order differencing and a combination of first-order (non-seasonal) differencing and first-order seasonal differencing work are possible options. Both versions thereby draw on components that can be seen in the decomposition. While the first variant accounts for a multiplicative trend in the data, the second variant accounts for seasonal effects. But again, considering the results for the most recent two years of the train set, the combined variant seems preferable. Rerunning the ADF test on the transformed dataset resulted in a test statistic of approximately -12 and a p-value below 1%. Figure 19 shows the plot for the transformed series.

Figure 19: C6 after applying non-seasonal and seasonal differencing

I further decided against testing a combination of second-order non-seasonal differencing and first-order seasonal differencing. The reason for this is that a less complex model is usually preferred in case of comparable outcomes, i.e., in case a simpler model already achieves to render the data stationary.

Group 3

The last dataset to inspect is C1. Here, I encountered some issues due to the limited length of the series. As I reduced the size of the available data again by splitting the data into two parts, only 12 data points were left in the train set.

This led to the result that decomposition was impossible, as the method requires data spanning at least two years. Therefore, a closer inspection of the components was not possible.

The analysis of the autocorrelation in the data was instead possible, even though with the limitation that I could only apply it up to the eleventh lag. The resulting ACF plot is shown in Figure 20.

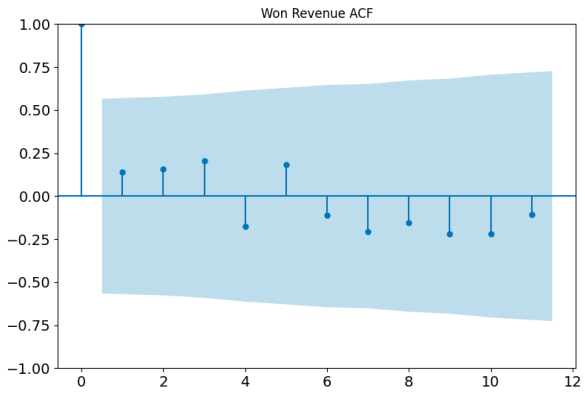


Figure 20: ACF plot for C6

Still, it can be argued that the results might not be very reliable due to the shortage of data points in the series. Revising the results from the plot with the ADF test generated a different result, indicating that the data is non-stationary and therefore required to be transformed. But due to the series' length, I could only apply normal, i.e., non-seasonal differencing. Used twice, the differencing led to significant results on the ADF test with a test statistic of approximately -6 and a p-value of below 1%. The resulting time plot is again illustrated in Figure 21.

Figure 21: C1 after applying second-order differencing

3.4 Related Research

Before starting with experimentation, it makes sense to get an overview of the state of the art and how practitioners have approached similar problems in recent years. Therefore, I conducted literature research using the *IEEE Xplore* and the *arXiv* databases.

In general, it was rather difficult to find articles and conference papers on revenue time-series forecasting of start-ups and scale-ups published in the last five years (2018-2022). After starting with a rather precise query which resulted only in a single hit across both databases, I started broadening the scope of my queries. Each query was supposed to capture one distinct aspect of the forecasting problem. The first query focused on revenue forecasting, the second on forecasting time series with volatility, and the third query targeted research papers that dealt with the problem of forecasting time series with scarce and/or limited data.

Each of those broader searches resulted in a high number of hits across both databases. The results retrieved for these broadened queries showed to be only partially applicable to answer my research question. This insight ultimately limited the search's benefit to developing a high-level intuition of how similar issues have been approached in the past.

Based on this, I decided to reduce the number of papers taken into consideration by undertaking two main actions. First, the time window was reduced to consider only papers published in the

last three years (2020-2022). Second, a qualitative approach, seeking theoretical saturation, was pursued instead of following a quantitative approach of reviewing all papers on this query. The idea here, originally coined by Glaser and Strauss in 1967, is to check only as many papers as needed to extract the key ideas on the topic under consideration and stop as soon as no further insights can be generated (Glaser and Strauss, 2017). I further introduced a threshold of a minimum of ten papers that needed to be considered before seeking theoretical convergence. This was done to ensure a minimum coverage of varying ideas and directions for each query. The decision on which papers to consider was based on how relevant I perceived a paper to be after reading through its abstract and chapter overview. By this, I was able to cut the number of reviewed papers significantly while at the same time minimizing the risk of missing out on core aspects relevant to approach my research question.

My research showed that investigations on revenue forecasting for start-ups and scale-ups in the B2B SaaS industry have been scarce so far. Still, one paper by Cao et al. from 2022 contained several parallels to the research problem covered in this thesis. In their paper, the authors pointed out the issue of usually having limited data points when attempting to forecast revenue development for start-ups and scale-ups. To solve this problem, they proposed a novel approach called *SiRe*. It was designed to fulfill multiple criteria, which also apply for this thesis, like being able to work on short revenue time series, providing long-term predictions for monthly data, and maximizing the explainability of the algorithm.

The key difference between the use case covered in the 2022 paper and the present use case is the perspective of the respective practitioner. While the present use-case starts from an inside perspective, i.e., a start-up or scale-up trying to forecast only their development, the paper's authors looked at it from an outside perspective of an investor. This investor perspective usually looks at a broad portfolio of companies simultaneously. This fact plays a vital role as the algorithm depended on the fact that the acting practitioner has access to a sufficient number of revenue time series from different companies. From these, the algorithm could learn the shared revenue (growth) distribution across all companies during training. In other words, the algorithm was designed to exploit the similarities and shared dynamics among companies in similar industry sectors and growth stages. This, in order to work correctly, required data from at least a few hundred companies. The authors also pointed out the difficulty of gathering data from private companies like start-ups and scale-ups in the paper. Coming from an inside perspective of a start-up or scale-up, this issue becomes even more severe, as their resources to

acquire access to such datasets are usually much more limited than in a comparably large investment business.

Another issue resulting from the disparity in perspective is the different goals pursued in each case. For investment professionals, it might not be critical to have predictions close to ground truth rather than seeing whether the overall increases (enough). In contrary, being close to the ground truth development might be, in fact, crucial for an inside perspective to provide reliable benchmarks for activities like target planning and budgeting.

Altogether, these differences result in a reduced transferability of the results of the paper by Cao et al. despite the high similarity of the problem statements on the first look.

Nevertheless, one valuable insight the authors gained from their experiments was that any methods that can benefit from the extraction of shared information proved to be superior to methods that lack this ability (e.g., statistical methods like ARIMA).

For the remaining broader searches, it could be seen again how time series forecasting is split into two main groups. This is traditional statistic approaches that were often used as baseline models on the one hand and ML/DL techniques which recently have become more dominant on the other hand.

With the first broadened query, I focused my research on forecasting revenue time series. This field has been targeted in industries like Air travel, Entertainment, Healthcare, and Retail. In my research, I could identify two core objectives that reoccurred in all reviewed papers for this query. The first one was about the question of which models work best for forecasting revenue development. Thereby, the portfolio of models that have been experimented with was broad – ranging from statistical methods like Vector Autoregression (VAR) models and autoregressive integrated moving average (ARIMA) models to ML/DL models like Long Short-Term Memory (LSTM) networks, Convolutional neural networks (CNN) and ensemble methods. The second reoccurring question was about which factors to train the forecasting model. As stated in Chapter 2.2, we can choose either a univariate or multivariate approach. Ultimately, this decision depends on the complexity of the data-generating process and the relevance of other (external) factors.

To tackle these questions, Vithitsoontorn and Chongstitvatana (2022) compared different modeling approaches to demand forecasting in the dairy industry. They used ARIMA as well as LSTM models. They varied the frequencies of their time-series data between using weekly and monthly data, as well as comparing the performance for univariate versus multivariate

approaches. After experimenting with data from different dairy production plants, they found no single best model for all time series. These insights highlighted that the choice of algorithm always depends on the time series in question. Meanwhile, Pundir et al. (2020) investigated the effect of including external factors when predicting future revenue generation in retail businesses. In their study, including external causal factors helped to improve the forecasting accuracy as it enhanced the algorithm's capability to uncover dependent patterns in the data. Factors they used here were e.g., a company's economic performance, the consumer price index, or the unemployment rate. Park et al. (2022) proposed an ensemble of classical and Deep Learning methods called *Forchestra,* which was meant to stabilize the predictive accuracy by combining forecasts of multiple Base Predictors. *Forchestra* outperformed multiple single-model approaches and other prominent ensemble methods in their experiments. Würfel et al. (2021) presented one relatively advanced approach, where they adopted a novel Temporal Fusion Transformer (TFT) model to predict a publisher's advertising revenue. With the TFT's attention-based architecture, the authors were able to leverage not only information on the revenue of the publisher in question but also about other customers.

Altogether, the research results for this query showed many options when attempting to forecast revenue time series with no golden go-to approach so far. On the one hand, we often see traditional models still being relatively prominent among practitioners in this field. Regardless, some situations require more advanced solutions like ML/DL or ensemble approaches. Furthermore, including external factors can be a suitable action where more context might be needed for the model to generate predictions.

With the second query, I removed the revenue focus and instead put it on the forecasting of volatile time series. On this topic, a lot of studies could be found – predominantly in the field of finance and stock markets. The overall picture was that LSTM models are often the preferred choice in case of volatility as they seem to be exceptionally good at capturing fluctuations in the data. One example is Örsel and Yamada (2022), which found that an LSTM model significantly outperformed their base model on the task of forecasting stocks with high volatility. Another example is Gajamannage and Park (2022), which found the LSTM model to be the most successful in their experiments in predicting development not only of stocks but also of cryptocurrencies and commodities. Emshagin et al. (2022) investigated the capabilities of the LSTM model for the short-term prediction of household electricity consumption. Again, the LSTM model turned out to be the best-performing model among the competitors. The selected competitors included a conventional ARIMA model as a base model and a Gated-

Recurrent-Unit (GRU) network which can be seen as a simplified version of the LSTM network. Furthermore, it can be observed that ARIMA often seemed to fail on time series that includes much volatility which can, e.g., be seen in Ranansinghe (2021).

The challenge of (high) volatility seems especially relevant in fields like finance and risk management where a wide selection of research explicitly devoted to forecasting the volatility itself can be found. One example is the *DeepVol* model proposed by Moreno-Pino and Zohren in 2022. Further, we see novel methods like *N-HiTS* (Challu et al., 2022) emerging to tackle the challenge of time-series forecasting under volatility.

Still, what is to be kept in mind when attempting to transfer the results of these papers onto a given use-case (from another domain) is that financial datasets are typically relatively large containing daily data ranging over several years. If this is not given – as in the present use case – the forecasting performance is likely lower as the algorithm has less information to base its prediction on.

Finally, the third focus query targeted contributions on the challenge of forecasting time series with limited and/or scarce data. The ideas in this field were more diverse than for the preceding queries, but it can be differentiated between three core ideas on a higher level.

The first and most popular one is about taking advantage of existing information on related entities. Based on this idea, Roster et al. (2022) used a Transfer Learning approach to forecast the development of new diseases in a setting with too little data to build a reliable epidemiological model. The authors found evidence that leveraging existing knowledge of related diseases can help improve predictions even if data on the target disease is limited. Nevertheless, they highlighted the importance of choosing the source disease carefully when applying this methodology as source and target domains. This means diseases must be sufficiently similar for Transfer Learning to be effective. Canaday et al. (2021) chose a different twist on the idea and instead went for a Meta-Learning approach. This approach was somewhat similar to the technique used in Cao et al. (2022), as the goal was again to learn shared patterns from a library of related time series. The issue remains the same and lies in the challenge of creating such a library with enough time series to extract meaningful information.

A second way to go about the challenge of limited data is Data Augmentation. In Javeri et al. (2021), the authors achieved significant improvements in the forecasting accuracy by leveraging a combination of Data Augmentation and Automated Machine Learning techniques. It is to be noted, though, that the datasets they used were still of intermediate length containing

over 300 data points which is much more than in the present use-case of this thesis. Thus, the transferability of the results might be limited.

Lastly, Hu et al. (2022) considered a third direction to solve the limited and/or scarce data challenge. Their proposed *AdaSource* approach was designed to boost the information value encoded in the data under consideration by leveraging related source data. This is similar to the approach of including external factors, given that those factors are known to have a causal relation to the data at hand.

A summary of the key insights for each query can be found in Table 4.

Table 3: Results of the literature review

Focus Query	Key Insights	Sources
Forecasting revenue time series of start-ups and scale-ups in B2B SaaS	▪ Little research in this field in recent years ▪ Focus so far rather on the outside perspective of investors rather than from within companies ▪ Still, some indication that forecasting might be insufficient if it is purely based on internal knowledge	▪ Cao et al. (2022)
Forecasting revenue time-series	▪ Research in various domains and industries ▪ typically pending between rather traditional statistical methods and state-of-the-art neural networks, sometimes using hybrid approaches ▪ suitable forecasting method depends on the domain, therefore finding it is not a trivial problem ▪ often beneficial to include external factors	▪ Charotia et al. (2021) ▪ Joshi et al. (2021) ▪ Kaewmanee et al. (2021) ▪ Kaunchi et al. (2021) ▪ Park et al. (2022) ▪ Pundir et al. (2020) ▪ Sharma et al. (2021) ▪ Syavasya & Muddana (2021) ▪ Venkataramani et al. (2019) ▪ Vithitsoontorn & Chongstitvatana (2022) ▪ Würfel et al. (2021) ▪ Zhang et al. (2020) ▪ Zháo & Jayadi (2021)
Forecasting time series with volatility	▪ in case of volatility, we often see LSTM be the preferred option ▪ many research papers, especially in the field of finance and stock markets	▪ Challu et al. (2022) ▪ Emshagin et al. (2022) ▪ Fjellström (2022) ▪ Gajamannage & Park, (2022)

	▪ the emergence of novel techniques like N-Hits to deal with volatility	▪ Lu & Yi (2022) ▪ Ma et al. (2021) ▪ Moreno-Pino & Zohren (2022) ▪ Muhammad et al. (2022) ▪ Orsel & Yamada (2022) ▪ Ranasinghe (2021)
Forecasting time series with limited/scarce data	▪ often trying to leverage existing knowledge through e.g., Transfer-Learning ▪ alternatively, trying to mimic new data points through Data Augmentation ▪ sometimes we can enrich information through e.g., feature boosting	▪ Canaday et al. (2021) ▪ Hu et al. (2022) ▪ Javeri et al. (2021) ▪ Jayasani et al. (2021) ▪ Jin et al. (2022) ▪ Kalla et al. (2020) ▪ Nastasescu & Cercel (2022) ▪ Puri et al. (2022) ▪ Ren et al. (2022) ▪ Roster et al. (2022)

3.5 Model Selection and Experiments

This chapter discusses the approach used for the experiments, including an overview of the three models used: *SARIMAX*, *NeuralProphet,* and *Temporal Fusion Transformer*. The main objective behind the selection of models was to cover the full range of conventional statistical methods to state-of-the-art ML/DL methods. The exact choice of models was based on my own experience from my studies, recent literature, and recommendations from my supervisor Daniel Hain[3].

Based on the collection of common components in quantitative forecasting models introduced in Chapter 2, Table 5 provides an overview of the three models and the components included for each.

Table 4: Selected methods and their key components

	SARIMAX	NeuralProphet	Temporal Fusion Transformer
Trend	X	X	
Seasonality	X	X	
AR	X	X	
MA	X		

[3] Daniel Hain's expertise in the field of time-series forecasting stems from teaching the subject of Machine Learning at the Copenhagen Business School as well as having supervised several projects in that area before.

Past Covariates	X	X	X
Hidden Layers		X	X
Self-Attention			X
Multi-Channel		X	X
Quantile Forecasts		X	X

The table highlights again how the selected models are distributed across the spectrum from statistical methods to ML/DL approaches. While there is a clear difference between the components used in the SARIMAX compared to the Temporal Fusion Transformer, the NeuralProphet is a hybrid approach that sits somewhere in between.

As discussed in Chapter 3.1, the ultimate goal of the experiments was to forecast the won revenue for the given time series across the following 12 periods. All models were trained in two ways: first, using the full series available on the prototype series and then again only using the most recent two years of the data. Further, I started using a heuristic approach to get a first feeling for the algorithms. Then I moved on to an optimization approach in the second step to see how fine-tuning the model configurations improves the results.

3.5.1 SARIMAX

The first model is SARIMAX, an extension of the Autoregressive Integrated Moving Averages (ARIMA) model, which was brought to public attention by Box & Jenkins in 1970 and has become a popular choice for all sorts of forecasting problems by now.

Model Overview

Hyndman & Athanasopoulos (2021) describes ARIMA as a statistical model which aims to describe the autocorrelations in the data. It combines three main components: an autoregressive component (AR), an integration (I), and a moving average (MA).

The first component discussed by the authors is the I component. This component is strongly related to the stationarity of the data and typically the first component to estimate in the model. The I component adjusts for the non-stationarity in the series to transform it into a state where the AR and MA components can be estimated for it (Hyndman & Athanasopoulos, 2021). In other words, the degree of the I component is equal to the order of differencing d that is needed to render the series stationary and is thus particularly used to account for a trend in the data. Again, it should be noted that the authors emphasize not choosing $d > 2$.

After the data is known to be stationary, it can be modeled in the means of AR and MA terms. Following the definitions in Chapter 2, p is used to express how many lags shall be considered within the AR process to predict future time steps. Further, q denotes the number of past

forecasting errors retrieved from the MA process. By combining these three components, a non-seasonal ARIMA model can be retrieved, which again can be defined as

$$y'_t = c + \emptyset_1 y'_{t-1} + \cdots + \emptyset_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (13)$$

following the same notation as in Chapter 2 with the only difference that this time y'_t is the (multiply) differenced series instead of the original one. A shorter notation for this is $ARIMA(p, d, q)$.

Each component can further be included on a seasonal level to account for seasonal patterns in the data. In that case, Hyndman & Athanasopoulos speak of a SARIMA model with the shortened notation $SARIMA(p, d, q)x(P, D, Q)_m$. The uppercase letters thereby represent the respective degrees on a seasonal level, and m is equal to the number of observations per year (e.g., twelve in the case of monthly data).

Finally, the implementation from the *statsmodels* library that I have used for my experiments extends the model by another feature: the option to include exogenous regressors, i.e., future covariates. This extended variant is called SARIMAX, where the 'X' refers to the exogenous regression component.

Experimentation approach

While the SARIMAX initialization function from the *statsmodels* library mentioned above contains several more hyperparameters (see Appendix 4), I initially focused on two main hyperparameters: *order* and *seasonal_order*. The model expects both variables to be tuple inputs representing the (p, d, q) or respectively $(P, D, Q)_m$ configuration. The fourth value in the expected input for *seasonal_order* is associated with the frequency m of the data which therefore needed to be set to 12. Thus, six hyperparameters remained to be selected – p,$d, q, P, D,$ and Q.

For the heuristics approach, the parameters associated with the I component were set based on the findings from the EDA in Chapter 3.3. To find the optimal configuration for the AR and MA components, I used a grid search approach, similar to what is done in the *AutoArima* method that is often used in R.[4] Here, I simply looped through all possible combinations of p, q, P and Q within $\{0,1,2,3\}$ for each parameter using the selected values of d and D. This resulted in a portfolio of 256 potential models for the heuristics approach. The final heuristics-

[4] The darts library in python contains an implementation of the *AutoARIMA* that is meant to replicate the functionality of the R equivalent. But after having spent some time on experimenting with this, it turned out not to be as useful as the original in R and was therefore discarded in the following experiments.

based model was chosen based on the AICc, a variant of Akaike's Information Criterion (AIC). The AIC is a standard metric for comparing the quality of statistical models on a given dataset. It rewards goodness of fit while at the same time trying to avoid overfitting by penalizing higher complexity. Compared to the original version, the AICc has been developed to correct for biases that sometimes occur for shorter time series when using the AIC (Hyndman & Athanasopoulos, 2021).

After having found the optimal heuristics model, it was trained again on the train set and finally tested based on its projection performance on the test set.

For the optimization approach, instead of keeping d and D fixed while optimizing the remaining parameters, they were included in the grid search as well. Thereby, the value of d was varied within the set of $\{0,1,2\}$, while the value of D was only varied in a binary logic within $\{0,1\}$. Overall, this increased the number of potential models from 256 to 1280 for each optimization experiment. One issue that arose here was that I could no longer use the AICc to select the optimal model. This was due to the reason that the value of the AICc is dependent on the length of the evaluated time series. In other words, time series with a higher degree of differencing produce a lower AICc on average, not because of a better fit, but due to their shorter length.

For this reason, I instead used a validation split within the train set, which was chosen to cover the last 12 periods. The final model was then selected based on the MSE that each model achieved when tested on the validation set. After choosing the best model within the optimization approach, this model was finally trained on the full train series.

Due to the already shortened two years set, I decided to only apply the optimization approach to the full series datasets.

3.5.2 NeuralProphet

The second selected model is NeuralProphet (NP), an evolution of Facebook's *Prophet* (Taylor & Letham, 2017). It is a hybrid ranging somewhere between statistical and ML/DL models. Introduced in a 2021 paper by Triebe et al., the key idea behind the NP was to find a balance between interpretability and performance. While Facebook's Prophet already met the interpretability aspect, Triebe et al. improved the performance by including neural layers in the NP architecture.

Model Overview

NP is composed of six modules, where each has its own inputs and modeling processes. The overall model formula can be written as

$$\hat{y}_t = T(t) + S(t) + E(t) + F(t) + A(t) + L(t), \quad (14)$$

where

$T(t)$ = Trend at time t

$S(t)$ = Seasonal effects at time t

$E(t)$ = Event and holiday effects at time t

$F(t)$ = Regression effects at time t for future known values exogenous variables

$A(t)$ = Auto-regression effects at time t based on past observations

$L(t)$ = Regression effects at time t for lagged observations of exogenous variables.

The authors note that the model can be configured individually, i.e., each module can be switched on or off according to what is needed. In case all modules are switched off, only a static offset parameter for the series level is fitted and used as the trend component. In its default mode, the model includes only trend and seasonality.

For the trend component, NP uses a classical approach of approximating an offset value m and a growth rate k with the difference that NP fits a piecewise trend. Triebe et al. explain that this allows the trend component to change over time, ultimately enabling the model to deal with time series that show a non-linear trend. The growth rate can thereby only change a finite number of times and is kept constant between these change points.

Analogous to its predecessor, NP uses Fourier terms to model seasonality. According to the authors, this has the advantage that Fourier terms produce smooth and interpretable functions which are stable to fit to data. On the other hand, they also point out the potential issue that Fourier terms rely on the assumptions that the seasonal shapes are fixed and will not change over time.

NP supports modeling additive and multiplicative seasonal patterns and automatically chooses from daily, weekly, and yearly seasonality depending on the data frequency and length. The authors note here that each type of seasonality will only be activated if at least two full periods of data are available (e.g., yearly seasonality can only be activated if the time series covers at least two years). In the case of a multiplicative configuration, the seasonal effect(s) will first

be scaled by the trend component for the respective time step before being added to the overall prediction.

The next component of the NP is an AR module. The main logic behind this remains the same as introduced in Chapter 2 and used in the ARIMA model but with one key difference. The traditional AR model is designed to produce one-step forecasts, i.e., using a forecasting horizon of $h = 1$. If we need a multi-step ahead forecast now with $h > 1$, h models must be fitted. Each is then associated with one forecast step and builds upon the preceding models by taking their one-step prediction as one input to predict the next step. By adopting the architecture of an existing modified version of *AR-Net* from (Triebe et al., 2019), NP can produce all h forecasts with one model. Accordingly, the output is h values corresponding to the AR effect for each forecasting step.

The main parameter to determine here is the number of past values to include when making each prediction. In the paper, this parameter is referred to as p. The choice depends on the approximate length of relevant context in past observations. Triebe et al. highlight that it is often hard to determine p accurately in practice. However, a common rule of thumb would be to set it to twice the innermost periodicity or twice the forecasting horizon.

Another parameter to tune in this model is the number of hidden layers. By default, the *AR-Net* does not contain any hidden layers but can be added if the data contains non-linear patterns. The trade-off here lies between a higher forecasting accuracy vs. a loss in interpretability. While we can quantify the contribution of each past value to each prediction directly in the default model, the deep architecture only allows us to observe the relative importance of each past observation on all predictions.

The next component of the NP model is lagged regressors, which encode information on past covariates. If lagged regressors are used, the authors explain NP to build a separate Lagged Regressor module for each of the m given covariates. From a functional perspective, each module is identical to the AR module, producing h predicted effects based on the last p observations from the respective covariate. These values are then added as additive components to the overall forecasts. All considerations regarding relevant parameters for this module are analogous to the AR module.

Further, in addition to past covariates, NP also allows to include the weighted effect of future covariates in the overall prediction. As for the seasonal component, the effect can either be configured to be additive or multiplicative.

The last module introduced by the authors targets the effect of special events and holidays. These may occur sporadically and are modeled analogous to future covariates, with the difference that these series only contain binary values on whether an event occurs at a specific time step. Apart from providing user-defined events, we can also include country-specific holidays for the Events and Holidays module. In this case, a country name can be provided to the model, automatically retrieving the national holidays and adding them to the set of events.

These components are combined to retrieve the model depicted at the beginning of this section. The paper's authors further introduce several more (optional) features included in the NP model, like automated data imputation or data normalization, which will not be explored further due to the limited scope of this thesis.

One additional feature included after the model's first introduction is quantile forecasts (GitHub, 2023). From the documentation website of the *neural_prophet* repository, which was published along with the introduction paper by Triebe on git can be taken that there are multiple ways to introduce uncertainty to the forecasts generated by NP models now (Triebe, 2021). Nevertheless, the feature seems rather new, and good documentation on how to work with it is still rare. For this reason, as well as due to the limited scope of this thesis, I decided not to dive deeper into the feature at this point.

A second feature of a later evolution of the model is global modeling which enables the model to deal with multi-channel data (GitHub, 2023). In this variant, the trend and the seasonal component can be configured to be estimated on a global level (default) or on a local level. If a local configuration is chosen, the respective component will be fitted across all data points within the same channel, i.e., containing the same value in the required ID field (Triebe, 2021).

Experimentation approach

For the experiments with the NP model, I used the *neural_prophet* repository mentioned above. The range of available hyperparameters here is much broader than for the SARIMAX model (see Appendix 5). Nevertheless, the NP model already includes some mechanisms to automate the selection of some parameters, such as those related to seasonality. Then, as the forecasting horizon was one year, i.e., twelve periods, I set the associated parameter *n_forecasts* to 12. From the remaining hyperparameters, I chose to focus on the number of lags (*n_lags*) and the number of hidden layers (*num_hidden_layers*), as recommended in Triebe et al. (2021).

I further relied on the recommendation of the authors for then choosing the number of lags in the heuristic approach. Thus, I aimed to use lags worth twice the forecasting horizon. This

worked out for the full-series training approach, where I set it to $p = 24$. In the case of the shortened train set, I could only set it to $p = 12$, however. For the Group 3 prototype, I could only set the number of lags to $p = 6$ due to the short length of the series. The number of hidden layers was set to $l = 0$ for all models in the heuristics approach.

One action that needed to be taken before training the model was to render the data compatible with the framework. This step was done by transforming it into a *DataFrame* object featuring a column 'ds', which stored the timestamps of the and a column 'y', which contained the values for the target variable at each time step. I relied on the default setup using an Adam optimizer and an auto-selected batch size, learning rate, and number of epochs for the training. The only training variable set individually was the training loss, for which I chose the MSE.

The resulting model was then trained on the train series. After that, a projection and some additional analytical plots could be generated from that, which I will draw on briefly in Chapter 4.

I introduced a validation split within the train set for the optimization approach. Using the predefined *split_df* function from the package mentioned above, I configured the validation set to cover the last 12 consecutive periods of the train set. By default, the function further included as many preceding lags as needed to produce the respective predictions on the validation set.

Due to the reduced length of the shortened series, I only applied the optimization approach for the full series data. This decision was also emphasized by the finding that the full series seemed to lead to a better performance of the resulting models than the shortened series. Further, the Group 3 prototype was omitted from the optimization approach as the train set was too short to split the dataset a second time.

I used a grid search again to find the optimal model configuration, optimizing for the validation loss in terms of MSE. Here, I looped through all available combinations with p ranging between zero and the respective maximum number of lags, i.e., 24, and l varying within the set of $\{0,1,2\}$. This approach resulted in 73 trained variants for each prototype of the two remaining groups. The training setup was similar to the one used in the heuristics approach.

After retrieving the best-performing model, it was trained again on the whole train set without splitting it prior to training. As the series length was already rather short, I decided on this approach to leverage as much of the available information as possible. Especially as the validation split covered the most recent part of the series, I could have otherwise risked losing meaningful information needed to forecast the following time steps.

On top of that, I started doing some multivariate experiments, following the same approach as the univariate experiments. Here, I used the time series on the number of created leads and the number of created opportunities as lagged regressors. The main idea behind these additional experiments was to win a first impression of the usefulness of including additional variables for the overall forecasting performance.

Initially, I further planned on executing some preliminary multi-channel experiments that needed to be discarded later in the process due to this project's limited time and scope.

3.5.3 Temporal Fusion Transformer

The last model I will use is the Temporal Fusion Transformer (TFT), a novel attention-based Deep Learning model introduced by Lim et al. in 2020. It was optimized for producing interpretable multi-horizon, i.e., multi-step forecasts while maintaining high performance. Here, the authors built upon previous advancements in the field of time-series forecasting, where practitioners have relied on recurrent neural network (RNN) architectures for a while before then, also starting to explore attention-based methods for the task of multi-step forecasting.

Model Overview

Even though we see some similarities in the TFT compared to the NP, TFT stands out, e.g., due to its capability of handling multiple time series simultaneously. By this, the TFT can leverage shared characteristics of the (related) time series to produce more accurate forecasts. In the context of revenue time series in B2B SaaS companies, this could, for example, be leveraged if the revenue time series is split by, e.g., regions or channels. This is done using static covariate encoders, a key building block of the TFT. At runtime, these allow the TFT to leverage information from static covariates by turning them into context vectors fed into different locations of the model (Lim et al., 2020).

Let there now be I unique entities in a given time series dataset where each entity i is associated with a set of static covariates $s_i \in \mathbb{R}^{m_s}$, as well as inputs $\mathcal{X}_{i,t} \in \mathbb{R}^{m_x}$ and target values $y_{i,t} \in \mathbb{R}$ for all time steps $t \in [0, T_i]$. $\mathcal{X}_{i,t}$ is further subdivided into observed inputs $z_{i,t} \in \mathbb{R}^{(m_z)}$ which are only known for the past and known inputs $x_{i,t} \in \mathbb{R}^{(m_x)}$ which are known both for the past and the future.

Then, the authors give the overall prediction formula of the TFT for entity i as

$$\hat{y}_i(q, t, \tau) = f_q\left(\tau, y_{i,t-p:t}, z_{i,t-p:t}, x_{i,t-p:t+\tau}, s_i\right), \tag{15}$$

where $\hat{y}_i(q, t, \tau)$ is a quantile forecast for the q^{th} sample quantile for the τ-step-ahead horizon prediction at time t and $f_q(.)$ is a prediction model. It is to be noted that TFT outputs predictions for all time steps $\tau \in \{1, \dots, \tau_{max}\}$ where τ_{max} is the desired time horizon of the forecast. Further, p indicates the length of the look-back window i.e., only past information within the time window $\{t - p, \dots, t\}$ will be considered for the prediction generation. Already from this prediction formula, it becomes apparent that the approach to calculating the final prediction differs greatly from the two other models. Instead of additively combining the different components of the model intuitively, the prediction generation happens in an Encoder-Decoder manner which thus feels more like a black box. For this introduction, I will not dive into the mathematical details of the TFT architecture now but rather provide a high-level overview of the main components. Nevertheless, a full schematic overview of the architecture can be found in Appendix 6.

Apart from the TFT feature of leveraging multi-channeled data, it can further be derived from the prediction formula that the model includes the option to account for covariates. This applies to past and future covariates encoded in $z_{i,t}$ and $x_{i,t}$ respectively. One of the main building blocks of the TFT mentioned in their paper in that context is variable selection networks. These are supposed to filter out unnecessary information from the covariates to prevent them from potentially decreasing the model's performance. Furthermore, this component boosts the interpretability of the TFT by providing insights into the relevance of each input for the final prediction.

The core building block of the TFT that is responsible for capturing the characteristics in the given series, like trend and seasonality, is referred to as temporal processing. This block is two-fold as it is used to simultaneously learn both long- and short-term dependencies from the input data. Lim et al. uses a recurrent sequence-to-sequence layer for short-term temporal relationships – or, more specifically, an LSTM layer. This layer ought to capture dependencies in the direct neighborhood of each observation. Long-term relationships, on the other hand, shall be captured within a modified multi-head attention block. The configuration of this block is also the key driver of the enhanced explainability of the TFT model, as the attention-heads work together in what can be understood as an ensemble manner. The result is a combined matrix containing information on the relevance of each observation for the overall prediction.

Like the NP model, the TFT uses quantile forecasts to provide a range of potential outcomes for the target variable on top of point forecasts. The authors achieved this by enabling the model

to predict multiple percentiles simultaneously at each time step, as seen in the prediction formula at the beginning of this chapter.

However, another useful feature of the TFT is gating. According to the authors, this is used to equip the model with the ability to automatically adjust its modeling complexity based on the complexity of the given dataset. Lim et al. proposes a Gated Residual Network (GRN) to implement this. This model identifies whether non-linear processing is needed to model the given data properly and, if not, suppresses any parts of the architecture that are irrelevant in this case.

Experimentation approach

For the experiments with the TFT, I relied on the implementation in the *darts* library.[5] Even though being the most complex model in my selection, the TFT model comes with a relatively low number of main input variables (see Appendix 7). The *output_chunk_length* is thereby equal to the forecasting horizon and was again set to 12. For the setting of most of the hyperparameters, I again relied on the authors' recommendation. Ultimately, I focused on tuning the *input_chunk_length,* which is equal to the number of lags p to include for the prediction generation, and the *hidden_size,* which the authors state to be the main hyper-parameter of the model.

Again, I started with a heuristic approach, following the same approach as the NP in choosing the value of p. For the *hidden_size*, I started by using the default value of 16.

In its default training setup, the TFT implementation in *darts* uses a *QuantileRegression* likelihood instead of a conventional loss function to produce a probabilistic model that outputs quantile forecasts. Further, it uses an Adam optimizer, a batch size of 32 and 100 epochs.

The first experiment failed across all series, producing forecasts that gave the impression that the model did not pick up any of the characteristics. Thus, I continued doing more general tests to find out whether this was due to the model's configuration. Here, I gradually changed the value of various hyperparameters to see whether any significant changes could be prompted. Minor improvements could be achieved by choosing a large size of hidden states and increasing the number of training epochs. Nevertheless, even then, the model performance stayed very low, so I ultimately decided against putting more effort into optimizing the TFT as the other models seemed to be much more promising while at the same time being much faster to train.

[5] Initially, I intended to use the implementation in the *pytorch_forecasting* library, but ultimately could not manage to make it work due to some conflicting dependencies in the installed packages.

Nevertheless, I could gain some insights to analyze and formulate assumptions on why the model performed that badly through my experimentation which I will elaborate on in Chapter 4.

4 Results

This chapter talks about the results of my experimentation. This includes the presentation of results and their analysis in the context of the research objective of this thesis. The chapter starts with an overview of the overall results across all datasets, mainly in terms of the RMSE. After that, it dives into the individual results for the prototypes of each dataset group. This includes a visual inspection of each model regarding their feature-capturing capabilities for the respective dataset, a comparison to the forecasting performance of the Growblocks model, and an overall assessment of which model might be most suitable for each dataset group.

The Growblocks model should thereby not be regarded as a hard benchmark. Rather, it was supposed to help understand the model's strengths and drawbacks to find the optimal way to complement the Growblocks approach utilizing quantitative methods. One argument for this was that the test set contained less than 12 forecasted periods for two out of three datasets. This especially plays a role as the Growblocks model had thus more recent information available to base its forecast on.

4.1 Overall Analysis

As described in Chapter 3.5, I split the experimentation into two approaches: the heuristics approach on the one hand and the optimization approach on the other. Appendices 8 and 9 show the final results for each approach in terms of the achieved RMSE. Generally, it can be said that the optimization approach failed in the case of the SARIMAX model, while it has proved to be very useful in the case of the NP model. For the TFT model, I only applied the heuristics approach for the reasons stated in Chapter 3.5.

In the following, I will discuss only the best results across both approaches that could be achieved for each prototype with each model, displayed in Table 6.

Table 5: Experimentation results

		C4 *Group 1 Prototype*		C6 *Group 2 Prototype*		C1 *Group 3 Prototype*	
		Configuration	RMSE	Configuration	RMSE	Configuration	RMSE
SARIMAX	Full series	$(1,0,1)x(0,0,0$	2.49E+03	$(2,1,1)x(0,1,1)$	1.87E+05	$(3,2,1)x(3,0,3)$	8.50E+04
	2 years	$(1,0,1)x(0,0,0$	1.95E+03	$(0,1,0)x(2,1,1)$	1.83E+05	-	-
	Full series	$p = 3, l = 1$	7.50E+03	$p = 1, l = 0$	1.65E+05	$p = 0, l = 0$	1.30E+05

NeuralProphet (univariate)	2 years	$p = 12, l = 0$	1.93E+04	$p = 12, l = 0$	2.87E+05	-	-
NeuralProphet (multivariate)	Full series	$p = 5, l = 2$	8.62E+03	$p = 24, l = 0$	1.59E+05	-	-
	2 years	$p = 12, l = 0$	2.81E+04	$p = 12, l = 0$	3.31E+05	-	-
Temporal Fusion Transformer	Full series	$p = 24,$ $hs = 128,$ $epochs = 10k$	3.22E+03	$p = 24,$ $hs = 128,$ $epochs = 10k$	1.80E+05	-	-
	2 years	$p = 24,$ $hs = 128,$ $epochs = 1k$	8.53E+03	$p = 24,$ $hs = 128,$ $epochs = 1k$	1.90E+05	-	-

The table lists each prototype-model combination's best-performing configuration and the respective RMSE error. I chose the RMSE here as I will only be comparing model performances individually for each dataset and thus have no need for a scale-independent error. Thereby, the notation of the configurations follows the definitions presented in Chapter 3.5. This means, in the case of the SARIMAX model, it follows the format$(p, d, q)x(P, D, Q)$. Further, for the NP and the TFT p refers to the number of lags used as input, l defines the number of hidden layers in the NP. Finally, hs is the size of each hidden layers within the TFT and $epochs$ defines the number of epochs the TFT was trained for.

To evaluate results, I will now look into both a dataset perspective and a model perspective. This is meant to provide an overview of which method performed best for each dataset group and how each selected model performed on the forecasting task in general.

Dataset perspective

For Group 1, which consisted of relatively simple datasets, the best results in terms of RMSE could be achieved with the SARIMAX model when trained with a heuristic approach on the shortened version of the dataset. Group 2, which contained more complex series, worked best with the NP model when optimized on the full series. Finally, for C1, i.e., Group 3, it was harder to find a suitable model. This challenge was mainly due to its very short length, which, e.g., did only allow for a very basic model in the NP and further prevented me from fitting a TFT model on the data. From the remaining two models, the NP could achieve a slightly better result with a pure trend-fitting approach.

Model perspective

From a model perspective, SARIMAX achieved a decent performance across all datasets, which again highlights its relevance even in times of advanceded approaches like ML/DL. Thereby, training it only on the most recent two years of the dataset showed to be more promising in the experiments, leading to a better performance than training it on the full-series

data. The NP often did not lead immediately to good results when following the heuristics approach. Still, when fine-tuned, it often showed its superiority over the SARIMAX, especially when it came to capturing more complex patterns. Here, the full series led to better results which might, to some degree, also be because I could not apply my optimization approach to the two-year data. Lastly, the TFT did not lead to promising results in most cases. This might mainly be due to the shortage of available data points within each series. This hypothesis is supported by (Cristina, 2022) but also mentioned in (Wolf et al., 2019) which state that Transformer models often show to be rather "data-hungry" models and thus cannot unfold her full potential when applied on small datasets. Even though a competitive result could be achieved for the most extensive dataset within my selection (C4), the model failed even with a significant increase in training time for shorter datasets.[6] This is also why the full series will likely always be preferred over the two-year set.

4.2 Analysis for Group 1

The first individual analysis focuses on C4 as the selected prototype of dataset Group 1. Group 1 showed a relatively stable development of the target variable, with the main volatility captured within a few outliers. Further, C4 also stuck out by being the longest dataset within the original six available datasets. The six forecasts generated for C4 are illustrated in Figure 22.

[6] This applies also for dataset C6, even though purely judging by the RMSE would suggest the opposite. This is due to the reason that the test set of C6 contains many zeros, leading to a wrong impression for any forecasts that default to zero and close to zero values in case of underfitting (see also Chapter 4.3).

Figure 22: Forecasts for C4

From a RMSE perspective, the SARIMAX model with the configuration $(1,0,1)x(0,0,0)$ trained on the two-year data (top right in Figure 22) led to the best result.

Evaluation of feature-extraction

Even though the RMSE can already provide a good indication of the fit of a model, it is reasonable to complement it with an analysis of how well each model performs in capturing key patterns and characteristics within the respective data.

Thus, regarding the performance of the SARIMAX from that perspective, the full-series and the two-year variant seemed to underfit the data. The result was an oversimplified model, consisting of a single piece-wise fitted line that follows the approximate level of the series. This makes sense when looking at the model's configuration, which was the same in both cases, as neither a trend component nor any seasonal effects were included. Instead, the data generation process was purely modeled within a combination of a first-level AR and a first-level MA process. After all, the fit on the test set was slightly better for the shortened data than for the full series. A possible reason for this could be that there are more large spikes in the full-series dataset, which could have led the model to falsely assign a higher relevance to them than in the version trained on the two-year series.

In the NP model, I observed an opposite tendency to the SARIMAX. Instead of underfitting the variation in the data, the NP tended to overstate it for its prediction. This could be due to assigning too much meaning to the few large spikes in the data. Overall, the full-series data led to better results for both the univariate and the multivariate NP, not only in terms of RMSE but also in terms of feature extraction. Interestingly, in the case of the C4 dataset, the model

performance of the NP decreased by adding past covariates. A plot of the parameters determined for the best model variant across the NP experiments with C4, which is the univariate NP trained on the full series, is visualized in Appendix 10. Especially the seasonality subplot in this emphasizes again how the model overstated effects related to the series' volatility, ultimately resulting in the strong fluctuations discussed above.

The last model to discuss is the TFT model. After the first few experiments with C4 failed, some further experiments with the model provided evidence that the model might eventually need a longer training time to pick up on the dynamics in the data due to the low number of available data points in the train set. Thus, after increasing the maximum number of epochs to 10.000 for the full-series data, the model achieved competitive results. Not only did it achieve a lower RMSE than the NP model, but the dynamics reflected in the forecast also appeared to be a good compromise between what could be seen in the SARIMAX and the NP, ultimately being very close to the actual dynamics of the series.

Comparison to the Growblocks model

Another thing to consider when evaluating the suitability of the three quantitative methods for modeling C4 is comparing it to the performance of the Growblocks model. In the case of C4, the Growblocks forecast spanned the same time horizon, which was used in my experiments; thus, performances could easily be compared across methods. Figure 23 shows the plot for the forecast generated by the Growblocks model.

Figure 23: Growblocks model forecast for C4

The forecast retrieved from the Growblocks model achieved an RMSE of approx. 4.04E+10. Thus, in my experiments, I could achieve better results with both the SARIMAX model across both dataset variants and the TFT model when trained long enough on the full-series data. Further, while both quantitative models were closer to matching the actual level of the test series, especially the TFT also showed to be better at estimating the dynamics inherent to the test period than the forecast generated by the Growblocks model.

Overall assessment

Overall, for datasets from Group 1 like C4, which show mostly low volatility and comparably simple to predict patterns, quantitative models seem helpful in dealing with revenue time series and forecasting their future development.

SARIMAX was the most suitable option from my experiments for modeling such datasets. Even though being a relatively basic model, the model proved to be sufficient to provide a benchmark for the future development of the series. Further, SARIMAX stands out by being very easy and fast to train and additionally provides highly interpretable outcomes. The main drawback of using the SARIMAX is that it might produce oversimplified models and can only generate point forecasts rather than providing quantile forecasts on top. When using the SARIMAX, using only the most recent two years of the available data makes sense to avoid introducing too much noise.

Suppose the respective dataset is large enough, and there might be an increased importance on modeling the (remaining) volatility close to the ground truth. In that case, one can further try applying a TFT with a high number of training epochs on the full series data.

4.3 Analysis for Group 2

The second analysis focuses on C6 as the prototype of dataset Group 2. Group 2 contained high volatility and, in the case of C6, irregular behavior featuring large outliers, which were especially hard to predict. It further showed a changing dynamic over time, with spikes becoming more prominent and frequent, especially in the last third of the series.

Figure 24 depicts the forecasts generated for C6 across all models.

Figure 24: Forecasts for C6

The best-performing model in terms of RMSE was the multivariate NP when trained on the full-series data using five lags and two hidden layers (third row left in Figure 24). The covariates included were the number of created leads (*CREATED_LEADS*) and the number of created opportunities (*CREATED_OPPORTUNITIES*) covering the same period as the target variable.

Evaluation of feature-extraction

Again, I want to look at the models' performances in capturing the dynamics within the data. Generally, I could see that all models seemed to perform better with the full series than with the two-year data in this aspect. In the case of the SARIMAX, the two-year model missed out on accounting for the growing magnitude of the spikes and instead stayed mainly at the zero line. The NP, in contrary, overstated the series' growth in the case of the shortened series. In

the case of the two-year TFT model, the same pattern of not picking up on dynamics could be observed as in Chapter 4.2.

On the other hand, the full-series models did a better job of picking up the features. The full-series SARIMAX forecast featured the characteristic change in the behavior of the series over time, even though the trend picked up a little too strongly in the end.

The full-series NP performed better on this, estimating a more conservative trend, but in the case of the univariate version, it missed out on increasing the magnitude of the spikes over time. This only happened in the multivariate variant. Here, the behavior looked already very similar to the ground truth within the forecasting horizon, even though the magnitude of the forecasted spikes still differed from their actual height in the test set. Nevertheless, some overfitting could be noted based on the training data's high fit versus the test set's reduced fit. Again, the resulting model's parameters are visualized in Appendix 10. This time, it contained the lag relevance not only for the target variable but also for the respective covariates included in the multivariate model. This can be very helpful in determining the actual relevance of each additional variable included in the model.

Finally, for the full-series TFT, the model did not seem to pick up on the patterns properly. It remained unclear whether this was because C6 is shorter than C4 and would thus need an even higher training time. Another explanation could be that the model would have needed a different configuration to capture the more complex relationships inherent to C6. The reason for the yet low RMSE in this experiment was because of the many zero values in the test set of C6. As the TFT seems to start building from zero during training, many test values are already met by default, even though the model has barely started fitting the data. Thus, by calculating the RMSE for a forecast only consisting of zeros, we would achieve approximately the same error as the two-year TFT. Anyways, the tiny size of the spikes within the projection of the full-series model, even in contrast to the ones present in the training data, implied that the model was strongly underfitting the data.

Comparison to the Growblocks model

Again, I want to include the prediction generated by the Growblocks approach in the discussion. It is to be noted that Growblocks generated the plan only for April 2022 onwards, which thus covers only 9 out of 12 periods. Figure 25 visualizes the forecast in a time plot.

Figure 25: Growblocks model forecast for C6

The Growblocks approach achieved an RMSE of 5.64E+5 for its forecast. In comparison, the two-year SARIMAX achieved an RMSE of 2.23E+5, the univariate full-series NP 1.88E+5, and the multivariate variant 1.68E+5 for the respective nine periods. From a dynamics perspective, the model suggested a strongly increasing trend which – even though it is close to the two prominent peaks within the forecasting horizon – is not in line with the remaining test data points. Thus, the model failed to account for the volatility which was already present in the data prior to the forecasting horizon.

Overall assessment

Taking these observations together, it makes sense to seek the aid of quantitative models when attempting to forecast complex datasets like C6, which feature a lot of volatility and similar challenging patterns.

I found the NeuralProphet most suitable within the selected models for this task. It keeps the computational expense at a reasonable level while at the same time being equipped to deal with the difficult dynamics present in the dataset. One drawback of the model is its increased complexity compared to the SARIMAX. The NP model includes many more hyperparameters and features, which can be very useful but make it harder to determine what is needed. A second issue in that context that I discovered in my experiments is that the model created by the NP depends largely on the initialization. Thus, the same configuration can lead to strongly differing forecasts if the initialization is not kept constant on every run.

To allow the model to capture the characteristics adequately, it makes sense to leverage all available data, i.e., the full-series data. Further, one should spend some time finding a

hyperparameter configuration that maximizes the model's generalizability onto the validation set.

4.4 Analysis for Group 3

The third and last analysis will focus on C1. Being the only dataset within Group 3, C1 stuck out due to its short length of only 24 entries, i.e., only 12 periods of training data. Apart from this, its dynamics appeared to lie somewhere between C1 and C2, showing a medium level of volatility and not seeming to behave particularly irregularly within the available time.

As stated at the beginning of this chapter, finding a quantitative model that could deal with such little data was challenging. Ultimately, two forecasts were generated, which are visualized in Figure 26.

Figure 26: Forecasts for C1

Regarding RMSE, the forecast retrieved from the NP (lower plot in Figure 26) was superior to that retrieved from SARIMAX. The model was trained on the full available training data and was configured to neither include an AR component nor any hidden layers due to the small size of the dataset.

Evaluation of feature-extraction

Looking at the plots, both models did not perform very well in picking up on the dynamics in the data.

Even though the SARIMAX accounted for some volatility in the data, the model's overall fit seemed somewhat random, featuring an increasing trend for the test period that fairly overshot the actual values.

The NP, on the other hand, did not account for any volatility in the data but also seemed to feature a similar trend to the SARIMAX. The parameters plot confirmed this impression, showing that the NP has only fitted a trend component (see Appendix 11). Upon closer inspection, it becomes apparent that additional components would have required more training data to be available. With all other components in the NP disabled, the NP was forced to model the data only in terms of a trend component and consequently failed to account for any characteristics that this could not explain.

Comparison to the Growblocks model

For the performance comparison of the Growblocks forecast with the ones retrieved from my experiments, only five projected values for 2022 were available for C4. Again, Figure 27 illustrates the forecast from the Growblocks model.

Figure 27: Growblocks forecast for C1

Across these five periods, the Growblocks approach achieved an RMSE of 5.31E+4. In my experiments, the SARIMAX reached an error of 1.14E+5 for the same time horizon; for the NP model, it was 1.74E+5. Due to the shortage of forecasted periods, it was relatively hard to provide a reliable evaluation of the feature-extraction capabilities of the respective Growblocks forecast. Still, within the available time window, it seemed okay, especially in comparison to the results of the quantitative models. The level was met mostly, even though the trend behavior could be a little stronger. Further, the forecast included some variation, even though the actual variation in the test set could be observed to be much stronger.

Overall assessment

After all, the results from my experimentation do not indicate much potential for using quantitative methods for modeling such short datasets, as seen in C1. Even though the SARIMAX proved to generate value already with a relatively short series covering only two years in the other experiments, it could not sufficiently capture the dynamics from this even shorter series covering only one year of training data. On the other hand, the NP and the TFT showed to be even more prone to a lack of data and are thus even less suitable for time series like C1.

5 Conclusion

The last chapter of this thesis is devoted to subsuming the results and discussing them in the context of recent research. First, I summarize the thesis and elaborate on the main findings regarding my two initial research questions in Chapter 5.1. In Chapter 5.2, I discuss how the results of this thesis contribute to scientific discussions about time-series forecasting and its application in practice. Based on this, Chapter 5.3 provides actionable recommendations on how Growblocks can use the results and integrate them into their existing system. Finally, after discussing some of the limitations of this thesis in Chapter 5.3, Chapter 5.4 concludes by setting an outlook for future research to continue where this thesis leaves off.

5.1 Summary

Forecasting is vital to any successful business nowadays. Thereby, companies use forecasting in various fields like Supply Chain Management, Human Resources Planning, and activities related to Financial Planning. One important task in this context is the definition of revenue targets. In modern B2B businesses, RevOps departments have evolved to maximize the companies' revenue potential. Growblocks is a company that has specialized in RevOps software and services, supporting their customers in forecasting revenue to come up with realistic, executable revenue plans. Thereby, Growblocks gets to work with time series data which mainly stem from start-up and scale-up businesses operating in B2B SaaS. Thus, the company is regularly confronted with various datasets, differing in their size and in the level of volatility. Currently, Growblocks uses a forecasting approach that, despite leveraging historical data, strongly relies on expert knowledge to generate predictions associated with several potential risks and biases.

In this thesis, I wanted to explore how quantitative forecasting methods can support revenue planning in providing realistic growth targets for start-up and scale-up companies in B2B SaaS.

To answer this, I focused on two interrelated research questions, Q1 and Q2. Q1 dealt with the question of whether quantitative methods can generate realistic forecasts for short, volatile revenue time series. On top of that, Q2 went a step further by asking what methods are preferred for different dataset types when attempting this.

To answer this, I conducted experiments using three quantitative models from the range of statistical methods to ML/DL methods to generate forecasts for three exemplary time series from different dataset groups. These groups differed primarily in terms of two criteria: length and volatility. Group 1 included datasets with a medium to high number of data points and a low level of volatility apart from a few outliers. Group 2 datasets were of similar length but were much more volatile and irregular. Lastly, Group 3 consisted of a single dataset with only one year of training data for the experiments. Apart from this main characteristic, the volatility of the Group 3 series seemed to be on a medium level ranging somewhere between Group 1 and 2.

The chosen models were SARIMAX, NeuralProphet, and Temporal Fusion Transformer. For each model, I conducted experiments on two variants of the training set: the full available training series on the one hand and then only the most recent two years of it on the other hand. By this, I aimed to determine whether the amount of available information or its recency is more important to retrieve a good forecast from each model.

Ultimately, I found that there is no absolute answer to Q1. Instead, it depends on the time series itself. The main factors that play a role here are the series' length and complexity. If the observed dynamics in the time series are rather simple, a relatively short time series of 2 years can already be sufficient to allow quantitative methods to capture the key components accurately. But the more complex a series appears, the greater the need for more historical data to enable the models to pick up on the patterns. If this applies, Q1 can be answered with yes, and quantitative models can greatly support forecasting revenue time series. In contrast, if a time series is very short compared to its complexity, Q1 must be answered with no, as quantitative methods did then not provide promising results. Further, regardless of the complexity of a time series, quantitative methods do not appear to be helpful in cases where only one year of data is available to train the models. Consequently, this illustrates a second case where Q1 must be answered with no.

To elaborate on the answer for Q2, I will again look at each of the three dataset groups individually. For Group 1 datasets, I found the SARIMAX to be the most suitable model trained on only the most recent two years of the data. Further, if the respective customer has been in

business for long already and can thus provide a comparably long time series, one can try modeling the data with the TFT, which showed to be potentially fruitful in this case.

Group 2 datasets yielded the best results when a multivariate NeuralProphet was fine-tuned on them. The model thereby leveraged the full series to capture meaningful patterns and relationships to account for more complex dynamics.

Finally, for Group 3 datasets, none of the quantitative models was able to generate a good forecast. Thus, the best working forecasting method, in this case, might be the current Growblocks approach as a hybrid of quantitative and qualitative methods, as it offers the option to include expert knowledge on the dynamics that is hard to extract purely from the available data.

Overall, it seems to be a good idea to start with a SARIMAX model whenever there is no intuition about which model to start with. From that, one can move on to testing more advanced models if the SARIMAX is insufficient in capturing complex patterns.

5.2 Contribution

My results contribute to research and discussion around time-series forecasting in multiple ways.

First, from a scientific perspective, the thesis contributes to the overall discussion around time series forecasting. More specifically, the results provide new insights into applying forecasting methods for short time series, revenue time series, and volatile time series. Moreover, this thesis is nested at the intersection of those three areas. It thus achieves to be even more specific with regards to the interaction of them and their associated challenges. The main contribution is evaluating the selected models' forecasting performance on the different datasets. These datasets represent multiple combinations of the challenges mentioned above, i.e., the dataset length and volatility. Thus, the thesis provides a comprehensive overview across two dimensions which is the models on the one hand and the data on the other.

Apart from that, from a business perspective, the findings provide insights for start-up and scale-up companies in B2B SaaS on how they can forecast their revenue. By comparing multiple models on their suitability not only for one but for multiple datasets with different patterns and specialties, the thesis supports businesses in choosing the suitable model and approach for their specific needs and characteristics. Further, the thesis serves as a high-level guide on how they can not only select the right model but also design their forecasting workflow end-to-end.

For Growblocks specifically, the results can be used not only in favor of their revenue forecasting but also to improve their existing offering for their customers. As providing accurate forecasts is one key aspect of Growblocks' value proposition, it is highly relevant to provide insights on potential improvements of these forecasts. Moreover, as Growblocks is regularly faced with different customer types, categorizing them into multiple groups of customers can be a significant advantage to render the forecasting more customized and thus drive an even higher accuracy for each group.

5.3 Recommendations for Growblocks

The question that remains open is how Growblocks should incorporate the insights won in this thesis into the Growblocks approach.

In the current setup, Growblocks starts by inspecting their customers' datasets within an EDA to identify characteristics and patterns in the series. After that, they set up the forecasting model for the respective customer based on their findings from the EDA and existing knowledge on the company and industry.

As the Growblocks approach still bears some undeniable advantages over a purely quantitative approach, like the models' full-funnel visibility and flexibility, I would not recommend replacing it with another model but rather complementing it. This approach is also emphasized in Bunn & Wright (1991) exploring the relationship and interaction between qualitative and quantitative forecasting methods. The authors' findings ultimately suggest that combining both can lead to improved forecasting accuracy.

Therefore, to begin with, Growblocks should implement a categorization step at the end of the EDA. Here, each dataset is put into one dataset group based on well-defined criteria. The grouping derived in this thesis serves as a starting point for this. Still, it might need to be refined as soon as more datasets become available and other determining criteria emerge.

After the dataset is placed within the suitable group, the remaining process follows the respective group-specific path.

In the case of Group 1 and 2 datasets, quantitative models showed to allow for more accurate forecasts than the Growblocks model. Thus, Growblocks can use quantitative forecasts as benchmark predictions to validate or correct the forecast retrieved from its model. To achieve this, Growblocks first needs to select a suitable model from the class of SARIMAX models in the case of Group 1 datasets and from the class of (multivariate) NP in Group 2 datasets. In the next step, the model is used to generate a forecast that Growblocks can compare its own

prediction against. Especially in the case of the NP, it makes sense to look at quantile forecasting again here to get a better estimate of the range of possible outcomes. If the Growblocks forecast shows to miss this prediction, they should rediscuss the existing assumptions and consider adjusting the Growblocks model configuration. Otherwise, if it does seem to match the quantitative forecast, it can be viewed as validated, which implies a higher chance of actually meeting the targets.

Additionally, as the improved forecasting performance of quantitative methods applies not only to the achieved RMSE but also concerning the feature-extraction capabilities, the models can further be used to support understanding the underlying patterns and relationships in the data. This, in turn, can then be used to fine-tune the configuration of the Growblocks model.

This combined approach can ultimately render the forecasting more accurate for both Group 1 and Group 2 datasets.

One potential risk here is the occurrence of existential changes like structural breaks, which neither the Growblocks approach nor quantitative methods might be able to predict. Thus, the strategy's success relies on the assumption that the observed dynamics within the dataset will continue in the future.

On the other hand, based on the results from my experiments, Growblocks should refrain from using quantitative methods in the case of Group 3 datasets. Instead, they should stick to the existing forecasting model and leveraging their background knowledge of the respective business and market for these kinds of concise time series. If done accurately, this shows to yield better forecasts than quantitative methods in that situation. Regardless, this state is only temporary and quantitative methods might become a viable option again as soon as the respective series grow long enough.

One general risk might be that there will be time series that fit neither dataset group. In this case, I recommend Growblocks start with a SARIMAX forecasting model. If the resulting model seems too simple to capture the dynamics within the train set accurately, they should move to a more advanced model like the NP. Given the model can achieve a good fit on the train data, it can be used as benchmark prediction, as described in the case of Groups 1 and 2. Regardless, this option should be considered cautiously, as the approach is not validated through experiments.

Overall, Growblocks is advised to invest strongly in exploiting quantitative forecasting in the future. By this, they can not only improve their existing offering of supporting other companies

in driving their revenue growth but also get the chance to unlock the next level of their own revenue potential.

5.4 Limitations

This thesis includes several limitations.

Firstly, by definition, the pursued approach assumes that all revenue time series of start-up and scale-up companies in B2B SaaS fall into one of the three predefined groups. This is strongly simplified as the categorization was derived from a tiny and thus non-representative sample of six datasets. In reality, the set of all available time series within this field might be much more diverse. It would therefore require a different grouping which might include a different number of groups and is defined based on other criteria. Following this, the results must be considered cautiously and might not be generalizable for any early-stage revenue time series.

If we now assume the group definition to be correct, one still needs to consider the results within each group with caution as they have been only retrieved based on one exemplary dataset. This was due to the time limitation of this thesis and ultimately affected the generalizability of results again. Thus, it is not secured after all that the specific outcome of my experiments was due to the reasons highlighted in my analysis rather than any other reasons. The experiments would need to be repeated on (at least) the remaining datasets within each group to validate the insights.

Another limitation relates to the selection of hyperparameters used in the optimization approach during my experiments. As seen in Chapter 3.5, there is a broad range of available hyperparameters for each of the selected models. The subset I selected for fine-tuning the models is only one of many available options. Thus, it cannot be ruled out that a different selection of hyperparameters would lead to other results than the ones presented in Chapter 4.

This might be even more true in the case of advanced features of the models. One important example is the multi-channel capabilities that are part of the NP architecture but play an exceptionally big role in the TFT. Potentially, the feature could have helped mitigate the issue of having only very few data points available, as it would have implied using all available splits of the time series instead of the one aggregated version used in my experiments. But due to the limited scope of this thesis and the unpromising results retrieved in the first preliminary experiments with the TFT, I did not start investigating whether this feature renders any significant performance improvements.

Next, the experimentation approach was limited to only three selected forecasting models while many more exist. Especially in ML/DL, there is a broad palette of novel models pursuing various approaches to capture meaningful patterns within time-series data. Some examples of these are DeepAR (Salinas et al., 2017), N-BEATS (Oreshkin et al., 2019) and N-Hits (Challu et al., 2022). Again, I was obliged to limit my selection to have enough time to deal in depth with each selected model. Nevertheless, this could have prevented me from finding an even more suitable model for the forecasting task.

One last limitation on the modeling side regards including external factors in the multivariate experiments. So far, the multivariate experimentation approach has focused only on covariates that are part of the Sales funnel. Still, no covariates from outside that could potentially affect the system, and thus the target variable, such as market factors, etc., were added to the model. From the articles reviewed in Chapter 3.4, like Pundir et al. (2020), this approach was already successful in other use cases and might have a similar potential for this use case.

But even if all these factors had been considered, there remains one high-level limitation around the question of how much complexity the model is supposed to account for. Generally, time series like those on business revenues are subject to a highly dynamic market ecosystem and are thus affected by various factors. This ultimately renders it extremely unlikely to be able to account for all sorts of influences and disturbances. Therefore, it is reasonable to discuss whether a relatively basic model that only accounts for key reoccurring patterns like seasonality and trend cycles should be preferred over a very complex model that will probably never be able to account for all sorts of causal relationships and effects correctly.

5.5 Outlook

Based on the limitations described in the preceding chapter, some potential directions for future research emerge.

Firstly, it would be important to revise the results of this thesis based on a more extensive selection of datasets to allow for more generalizability. Not only would this allow for a more well-founded definition of groups, but in case model selection and fitting would further be applied for all datasets within every group, it could also support achieving a higher generalization about the model selection within each group.

Another direction could be to dive deeper into one of the models used in this thesis. Especially with the NP and the TFT, there exist a lot of configurations and features that have not been focused on yet. While it was essential to get an overall intuition about the suitability of each

model in the first step covered in this thesis, further research can build upon that by digging deeper into the realizable forecasting potential of each model encoded in their different features and components.

Apart from diving deeper into the already given models, future work could also focus on broadening the portfolio by testing other models that have not been focused on yet. Even though the model selection in this thesis gives a first intuition about which models across the spectrum are to be preferred in what situation, many more models could work even better in the given use case.

One more direction for research could be the exploration of external factors. As markets are highly complex, dynamic ecosystems, various factors could play a role in forecasting the revenue growth of a company operating within a particular market. Future work could thus focus on collecting potential influence factors and evaluating them based on their information value for the forecasting task. This could help render models less naïve and have them account for relevant causal factors outside the Sales funnel.

Finally, relating to the last, high-level limitation mentioned in Chapter 5.4, future research could attempt to zoom out again to investigate the qualitative question of how much complexity is desirable for determining revenue targets in businesses. By coming up with a more pinpointed definition of this, research would ultimately allow for a more targeted selection of models that is alignment with the actual needs of practitioners and businesses.

References

6sense. (2023, May 6). *How to Build a B2B Sales Funnel: The 4 B2B Sales Funnel Stages | 6sense*. https://6sense.com/blog/b2b-sales-funnel/

Aggarwal, C. C. (2018). *Neural networks and deep learning: A Textbook*. Springer. http://repository.psa.edu.my/handle/123456789/2034

Babai, M. Z., Ali, M. M., Boylan, J. E., & Syntetos, A. A. (2013). Forecasting and inventory performance in a two-stage supply chain with ARIMA(0,1,1) demand: Theory and empirical analysis. *International Journal of Production Economics*, *143*(2), 463–471. https://doi.org/10.1016/j.ijpe.2011.09.004

Baz, J., Granger, N. M., Harvey, C. R., Le Roux, N., & Rattray, S. (2015). *Dissecting Investment Strategies in the Cross Section and Time Series.* https://doi.org/10.2139/ssrn.2695101

Bergmeir, C., & Benítez, J. M. (2012). On the use of cross-validation for time series predictor evaluation. *Information Sciences*, *191*, 192–213. https://doi.org/10.1016/j.ins.2011.12.028

Box, G. E. P., & Jenkins, G. M. (1970). *Time series analysis;: Forecasting and control*. *Holden-Day series in time series analysis*. Holden-Day.

Bulla, D. N., & Scott, P. M. (1987). Manpower Requirements Forecasting: A Case Example. In *Strategic Human Resource Planning Applications* (pp. 145–155). Springer, Boston, MA. https://doi.org/10.1007/978-1-4613-1875-0_12

Bunn, D., & Wright, G. (1991). Interaction of Judgemental and Statistical Forecasting Methods: Issues and Analysis. *Management Science*, *37*(5), 501–518. http://www.jstor.org/stable/2632457

Canaday, D., Pomerance, A., & Girvan, M. (2021, October 7). *A Meta-learning Approach to Reservoir Computing: Time Series Prediction with Limited Data*. https://arxiv.org/pdf/2110.03722

Cao, L., Horn, S., Ehrenheim, V. von, Anselmo Stahl, R., & Landgren, H. (2022). *Simulation-Informed Revenue Extrapolation with Confidence Estimate for Scaleup Companies Using Scarce Time-Series Data*. https://arxiv.org/pdf/2208.10375 https://doi.org/10.1145/3511808.3557110

Challu, C., Olivares, K. G., Oreshkin, B. N., Garza, F., Mergenthaler-Canseco, M., & Dubrawski, A. (2022, January 30). *N-HiTS: Neural Hierarchical Interpolation for Time Series Forecasting*. https://arxiv.org/pdf/2201.12886

Charotia, H., Garg, A., Dhama, G., & Maheshwari, N. (2021, December 15). *Optimal Latent Space Forecasting for Large Collections of Short Time Series Using Temporal Matrix Factorization*. https://arxiv.org/pdf/2112.08052

Cleveland, R. B., Cleveland, W. S., McRae, J. E., & Terpenning, I. (1990). STL: A seasonal-trend decomposition procedure based on loess, *6*(1), 3–33.

Coad, A., & Hölzl, W. (2012). Firm Growth: Empirical Analysis. *Handbook on the Economics and Theory of the Firm*, 324–338. https://doi.org/10.4337/9781781002407.00035

Cristina, S. (2022, October 19). Inferencing the Transformer Model. *Machine Learning Mastery*. https://machinelearningmastery.com/inferencing-the-transformer-model/

Cryer, J. D., & Chan, K. (2008). *Time series analysis: With applications in R* (Second edition, corrected at third printing). *Springer texts in statistics*. Springer.

Dagum, E. B., & Bianconcini, S. (2016). *Seasonal Adjustment Methods and Real Time Trend-Cycle Estimation*. *SpringerLink Bücher*. Springer. https://doi.org/10.1007/978-3-319-31822-6

Dechow, P. M., Hutton, A. P., & Sloan, R. G. (2000). The Relation between Analysts' Forecasts of Long-Term Earnings Growth and Stock Price Performance Following Equity Offerings *Contemporary Accounting Research*, *17*(1), 1–32. https://doi.org/10.1111/j.1911-3846.2000.tb00908.x

Digitopia, Inc. (2023, April 15). *What is RevOps? The Ultimate Guide to Revenue Operations*. https://www.digitopia.agency/revenue-operations-revops-starter-guide

Emshagin, S., Halim, W. K., & Kashef, R. (2022, December 16). *Short-term Prediction of Household Electricity Consumption Using Customized LSTM and GRU Models*. https://arxiv.org/pdf/2212.08757

Ensafi, Y., Amin, S. H., Zhang, G., & Shah, B. (2022). Time-series forecasting of seasonal items sales using machine learning – A comparative analysis. *International Journal of Information Management Data Insights*, *2*(1), 100058. https://doi.org/10.1016/j.jjimei.2022.100058

FasterCapital. (2023, April 2). *The Importance of AchievingYour Annual Revenue Target - FasterCapital*. https://fastercapital.com/content/The-Importance-of-AchievingYour-Annual-Revenue-Target.html

Fjellström, C. (2022, January 20). *Long Short-Term Memory Neural Network for Financial Time Series*. https://arxiv.org/pdf/2201.08218

Franses, P. H., van Dijk, D., & Opschoor, A. (2014). *Time series models for business and economic forecasting* (Second edition). *Economics, statistics and mathematical economics*. Cambridge University Press.

Gajamannage, K., & Park, Y [Yonggi]. (2022, May 10). *Real-time Forecasting of Time Series in Financial Markets Using Sequentially Trained Many-to-one LSTMs*. https://arxiv.org/pdf/2205.04678

GitHub. (2023, April 22). *Releases · ourownstory/neural_prophet*. https://github.com/ourownstory/neural_prophet/releases

Growblocks. (2023, May 9). *RevOps Platform - Manage your revenue engine*. https://growblocks.com/

Hu, C., Wang, J., Wu, D., Liu, X., & Dudek, G. (2022). Accurate Communication Traffic Forecasting with Multi-Source Adaptive Feature Boosting. In *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*.

HubSpot. (2023, April 11). *What Is Inbound Marketing? | HubSpot*. https://www.hubspot.com/inbound-marketing

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: principles and practice, 3rd edition.* OTexts. OTexts.com/fpp3

Javeri, I. Y., Toutiaee, M., Arpinar, I. B., Miller, J. A., & Miller, T. W. (2021). Improving Neural Networks for Time-Series Forecasting using Data Augmentation and AutoML. In *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*.

Jayasani, C., Dammalage, P., Sarathchandra, S., Godaliyadda, R., Ekanayake, P., Herath, V., Ekanayake, J., & Dharmaratne, S. (2021). Limited Data Forecasting for Dengue Propagation. In *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*.

Jerez, T., & Kristjanpoller, W. (2020). Effects of the validation set on stock returns forecasting. *Expert Systems with Applications*, *150*, 113271. https://doi.org/10.1016/j.eswa.2020.113271

Jin, X., Park, Y [Youngsuk], Maddix, D. C., Wang, H., & Wang, Y. (2022). Domain Adaptation for Time Series Forecasting via Attention Sharing. In *Proceedings of the 39 th International Conference on Machine Learning*. https://arxiv.org/pdf/2102.06828

Josef Perktold, Skipper Seabold, Kevin Sheppard, ChadFulton, Kerby Shedden, jbrockmendel, j-grana6, Peter Quackenbush, Vincent Arel-Bundock, Wes McKinney, Ian Langmore, Bart Baker, Ralf Gommers, yogabonito, s-scherrer, Evgeny Zhurko, Matthew Brett, Enrico Giampieri, Yichuan Liu, . . . Yaroslav Halchenko. (2023). *statsmodels/statsmodels: Release 0.14.0*. Zenodo. https://zenodo.org/record/7899735#.ZFYk5OxByhY https://doi.org/10.5281/zenodo.7899735

Joshi, V., Jha, K., Jain, M., & Kulkarni, S. (2021). Tourism Footfall Forecasting and Recommendation System. In *2021 International Conference on Communication information and Computing Technology (ICCICT)*.

Kaewmanee, P., Muangprathub, J., & Sae-Jie, W. (2021). Forecasting Tourist Arrivals with Keyword Search using Time Series. In *2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*.

Kalla, R., Murikinjeri, S., & Abbaiah, R. (2020). An Improved Demand Forecasting with Limited Historical Sales Data. In *2020 International Conference on Computer Science and Management Technology (ICCSMT)*.

Kaunchi, P., Jadhav, T., Dandawate, Y., & Marathe, P. (2021). Future Sales Prediction For Indian Products Using Convolutional Neural Network-Long Short Term Memory. In *2021 2nd Global Conference for Advancement in Technology (GCAT)*.

Lim, B., Arik, S. O., Loeff, N., & Pfister, T. (2019, December 19). *Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting*. https://arxiv.org/pdf/1912.09363

Lindgren, B. W. (1993). *Statistical theory* (Fourth edition) [Routledge]. https://permalink.obvsg.at/

Lu, J., & Yi, S. (2022). Reducing Overestimating and Underestimating Volatility via the Augmented Blending-ARCH Model. *Applied Economics and Finance*, *9*(2), 48. https://doi.org/10.11114/aef.v9i2.5507

Ma, Z., Wang, C., & Zhang, Z. (2021). Deep Learning Algorithms for Automotive Spare Parts Demand Forecasting. In *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*.

Makridakis, S. G., Wheelwright, S. C., & MacGee, V. E. (1983). *Forecasting: Methods and applications* (2. ed.). Wiley. http://www.loc.gov/catdir/enhancements/fy0607/82023858-d.html

Mills, T. C. (2019). An Introduction to Forecasting With Univariate Models. In T. C. Mills (Ed.), *Applied time series analysis: A practical guide to modeling and forecasting* (pp. 121–130). Academic Press. https://doi.org/10.1016/B978-0-12-813117-6.00007-7

Moreno-Pino, F., & Zohren, S. (2022, September 23). *DeepVol: Volatility Forecasting from High-Frequency Data with Dilated Causal Convolutions*. https://arxiv.org/pdf/2210.04797

Muhammad, N., Shah, S. W., & Khan, G. M. (2022). Evolving computationally efficient prediction model for Stock Volatility using CGPANN. In *2022 2nd International Conference on Artificial Intelligence (ICAI)*.

Nastasescu, G.-S., & Cercel, D.-C. (2022). Conditional Wasserstein GAN for Energy Load Forecasting in Large Buildings. In *2022 International Joint Conference on Neural Networks (IJCNN)*.

OECD. (2023, February 11). *Agricultural output - Meat consumption - OECD Data*. https://data.oecd.org/agroutput/meat-consumption.htm

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019, May 24). *N-BEATS: Neural basis expansion analysis for interpretable time series forecasting*. https://arxiv.org/pdf/1905.10437

Orsel, O. E., & Yamada, S. S. (2022, January 31). *Comparative Study of Machine Learning Models for Stock Price Prediction*. https://arxiv.org/pdf/2202.03156

Park, Y.-J., Kim, D., Odermatt, F., Lee, J., & Kim, K.-M. (2022). A Large-Scale Ensemble Learning Framework for Demand Forecasting. In *2022 IEEE International Conference on Data Mining (ICDM)* (pp. 378–387). https://doi.org/10.1109/ICDM54844.2022.00048

Pundir, A. K., Ganapathy, L., Maheshwari, P [Pratik], & Kumar, M. N. (2020). Machine Learning for Revenue Forecasting: A Case Study in Retail business. In *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 201–207). https://doi.org/10.1109/IEMCON51383.2020.9284819

Puri, C., Kooijman, G., Vanrumste, B., & Luca, S. (2022). Forecasting time series in healthcare with Gaussian processes and Dynamic Time Warping based subset selection. *IEEE Journal of Biomedical and Health Informatics*, *PP*(12), 6126–6137. https://doi.org/10.1109/JBHI.2022.3214343

Ranasinghe, S. (2021). Univariate Time Series Forecasting Under High Volatility: A Case Study of Sri Lanka Stock Price Index. In *2021 10th International Conference on Information and Automation for Sustainability (ICIAfS)*.

Reiter, A. (2021, January 29). Inbound vs Outbound Sales: What's the Difference. *CloudTask*. https://cloudtask.com/blog/what-is-the-difference-between-inbound-and-outbound-sales

Ren, X., Zhang, X., & Zhao, C. (2022). Migration Data-based Graph Neural Network for Disease Forecasting. In *2022 8th International Conference on Big Data and Information Analytics (BigDIA)*.

Robinson, C. (1965). Some Principles of Forecasting in Business. *The Journal of Industrial Economics*, *14*(1), 1. https://doi.org/10.2307/2097647

Roster, K., Connaughton, C., & Rodrigues, F. A. (2022). *Forecasting new diseases in low-data settings using transfer learning*. https://arxiv.org/pdf/2204.05059 https://doi.org/10.1016/j.chaos.2022.112306

Salinas, D., Flunkert, V., & Gasthaus, J. (2017, April 13). *DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks*. https://arxiv.org/pdf/1704.04110

Sharma, A. K., Kiran, M., Pauline Sherly Jeba, P., Maheshwari, P [Prateek], & Divakar, V. (2021). Demand Forecasting Using Coupling Of Machine Learning And Time Series Models For The Automotive After Market Sector. In *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*.

Shynkevich, Y., McGinnity, T. M., Coleman, S. A., Belatreche, A., & Li, Y [Yuhua] (2017). Forecasting price movements using technical indicators: Investigating the impact of varying input window length. *Neurocomputing*, *264*, 71–88. https://doi.org/10.1016/j.neucom.2016.11.095

Statistics Denmark. (2023, May 6). *Exchange rates*. https://www.dst.dk/en/Statistik/emner/oekonomi/valutakurser-renter-og-vaerdipapirer/valutakurser

Syavasya, C., & Muddana, A. L. (2021). Machine learning based Time series prediction using Holt-Winters Exponential Smoothing with Multiplicative Seasonality. In *2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT)*.

Taylor, S. J., & Letham, B. (2017). Forecasting at scale. *PeerJ Preprints.* Advance online publication. https://doi.org/10.7287/peerj.preprints.3190v2

Triebe, O. (Ed.). (2021). *Core Module Documentation - NeuralProphet documentation*. https://neuralprophet.com/code/forecaster.html

Triebe, O., Hewamalage, H., Pilyugina, P., Laptev, N., Bergmeir, C., & Rajagopal, R. (2021, November 29). *NeuralProphet: Explainable Forecasting at Scale*. https://arxiv.org/pdf/2111.15397

Triebe, O., Laptev, N., & Rajagopal, R. (2019, November 27). *AR-Net: A simple Auto-Regressive Neural Network for time-series*. https://arxiv.org/pdf/1911.12436

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention Is All You Need*. https://arxiv.org/pdf/1706.03762

Venkataramani, S., Ramesh, A., S, S. S., Jain, A. K., Gudur, G. K., & Vijayaraghavan, V. (2019). A Dynamically Adaptive Movie Occupancy Forecasting System with Feature Optimization. In *2019 International Conference on Data Mining Workshops (ICDMW)*.

Vithitsoontorn, C., & Chongstitvatana, P. (2022). Demand Forecasting in Production Planning for Dairy Products Using Machine Learning and Statistical Method. In *2022 International Electrical Engineering Congress (iEECON)*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., Platen, P. v., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., . . . Rush, A. M. (2019, October 9). *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. https://arxiv.org/pdf/1910.03771

World Bank Climate Change Knowledge Portal. (2023, May 4). *Denmark Climatology*. https://climateknowledgeportal.worldbank.org/country/denmark/climate-data-historical

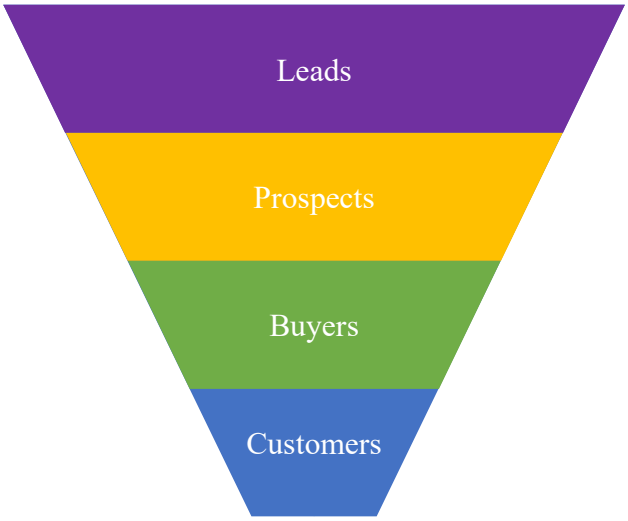
Würfel, M., Han, Q., & Kaiser, M. (2021). Online Advertising Revenue Forecasting: An Interpretable Deep Learning Approach. In *2021 IEEE International Conference on Big Data (Big Data)*.

Zhang, S., Zhang, H., Zhu, Y., Sun, J [Jinhu], Yuan, D., Li, H., Zhang, J., Bo, L., Li, Y [Yunling], & Sun, J [Jing] (2020). Optimizing Social Economy Prediction based on Integration of Time Series Models. In *2020 35th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*.

Zháo, M. A., & Jayadi, R. (2021). Forecasting Daily Visitors and Menu Demands in an Indonesian Chain Restaurant using Support Vector Regression Machine. In *2021 International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*.

Appendix

Appendix 1: Sales Funnel in B2B (6Sense, 2021)



Appendix 2: Boxplot for Won Revenue across datasets

Appendix 3: Dataset overview (2 years only)

Dataset	Number of data points	Mean	Standard Variance	Maximum	75% Percentile	Median	25% Percentile	Minimum
C1	24	48,365	40,136	159,383	65,245	41,010	19,590	2,320
C2	24	150,913	121,301	479,714	204,532	109,724	64,156	7,721
C3	24	184,563	110,570	591,171	207,848	164,434	118,276	45,989
C4	24	7,728	2,207	11,801	9,565	7,858	5,789	4,171
C5	24	26,931	8,375	45,651	30,591	25,690	22,149	12,798
C6	24	88,649	128,617	550,902	151,848	43,529	0	0

Appendix 4: SARIMAX input parameters (Pektold et al., 2023)

Input parameter	Data Type	Description	
endog	*array_like*	The observed time-series process y.	
exog	*array_like*	(Optional) Array of exogenous regressors, shaped nobs x k.	None

order	*iterable*	(Optional) The (p,d,q) order of the model for the number of AR parameters, differences, and MA parameters. d must be an integer indicating the integration order of the process, while p and q may either be an integers indicating the AR and MA orders (so that all lags up to those orders are included) or else iterables giving specific AR and / or MA lags to include.	(1,0,0)
seasonal_order	*iterable*	(Optional) The (P,D,Q,s) order of the seasonal component of the model for the AR parameters, differences, MA parameters, and periodicity. D must be an integer indicating the integration order of the process, while P and Q may either be an integers indicating the AR and MA orders (so that all lags up to those orders are included) or else iterables giving specific AR and / or MA lags to include. s is an integer giving the periodicity (number of periods in season), often it is 4 for quarterly data or 12 for monthly data.	(0,0,0,0)
trend	*str* or *iterable*	(Optional) Parameter controlling the deterministic trend polynomial $A(t)$. Can be specified as a string where 'c' indicates a constant (i.e. a degree zero component of the trend polynomial), 't' indicates a linear trend with time, and 'ct' is both. Can also be specified as an iterable defining the non-zero polynomial exponents to include, in increasing order. For example, [1,1,0,1] denotes $a + bt + ct^3$.	None
measurement_error	*bool*	(Optional) Whether or not to assume the endogenous observations endog were measured with error.	False
time_varying_regression	*bool*	(Optional) Used when an explanatory variables, exog, are provided to select whether or not coefficients on the exogenous regressors are allowed to vary over time.	False
mle_regression	*bool*	(Optional) Whether or not to use estimate the regression coefficients for the exogenous variables as part of maximum likelihood estimation or through the Kalman filter (i.e. recursive least squares). If *time_varying_regression* is True, this must be set to False.	True
simple_differencing	*bool*	(Optional) Whether or not to use partially conditional maximum likelihood estimation. If True, differencing is performed prior to estimation, which discards the first	False

		$sD + d$ initial rows but results in a smaller state-space formulation. See the Notes section for important details about interpreting results when this option is used. If False, the full SARIMAX model is put in state-space form so that all datapoints can be used in estimation.	
enforce_stationarity	*bool*	(Optional) Whether or not to transform the AR parameters to enforce stationarity in the autoregressive component of the model.	True
enforce_invertibility	*bool*	(Optional) Whether or not to transform the MA parameters to enforce invertibility in the moving average component of the model.	True
hamilton_representation	*bool*	(Optional) Whether or not to use the Hamilton representation of an ARMA process (if True) or the Harvey representation (if False).	False
concentrate_scale	*bool*	(Optional) Whether or not to concentrate the scale (variance of the error term) out of the likelihood. This reduces the number of parameters estimated by maximum likelihood by one, but standard errors will then not be available for the scale parameter.	False
trend_offset	*int*	(Optional) The offset at which to start time trend values. Default is 1, so that if *trend*='t' the trend is equal to 1, 2, …, nobs. Typically is only set when the model created by extending a previous dataset.	1
use_exact_diffuse	*bool*	(Optional) Whether or not to use exact diffuse initialization for non-stationary states.	False
dates		*description missing*	None
freq		*description missing*	None
missing		*description missing*	'none'
validate_specification		*description missing*	True
*****kwargs***		(Optional) Keyword arguments may be used to provide default values for state space matrices or for Kalman filtering options.	

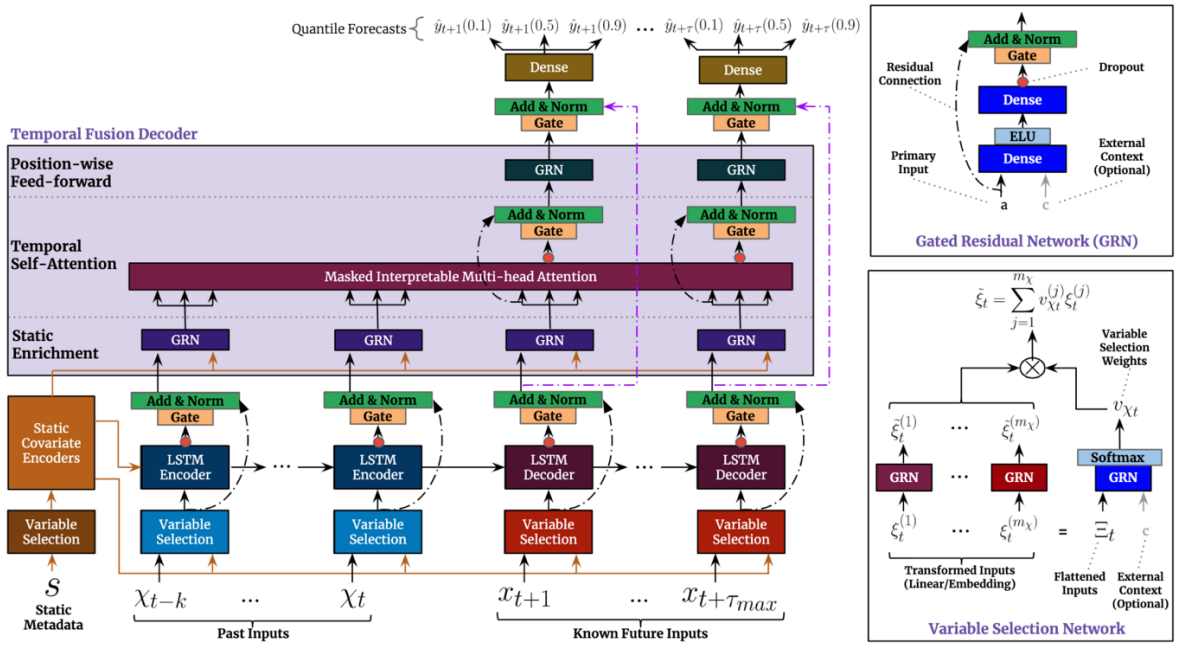
Appendix 5: NeuralProphet input parameters (Triebe, 2023)

Input parameter	Data Type	Description	Default Value
growth	str	Set use of trend growth type.	'linear'
changepoints	*list[str], list[np.datetimes], np.array[np.datetimes]*	(Optional) Manually set dates at which to include potential changepoints.	None

n_changepoints	*int*	Number of potential trend changepoints to include.	10
changepoints_range	*float*	Proportion of history in which trend changepoints will be estimated.	0.8
trend_reg	*float*	Parameter modulating the flexibility of the automatic changepoint selection.	0
trend_reg_threshold	*bool*	(Optional) Allowance for trend to change without regularization.	False
trend_global_local	*str*	Modelling strategy of the trend when multiple time series are present.	'global'
yearly_seasonality	*bool*	Fit yearly seasonality.	'auto'
weekly_seasonality	*bool*	Fit monthly seasonality.	'auto'
daily_seasonality	*bool*	Fit daily seasonality.	'auto'
seasonality_mode	*str*	Specifies mode of seasonality.	'additive'
seasonality_reg	*float*	(Optional) Parameter modulating the strength of the seasonality model.	0
season_global_local	*str*	Modelling strategy of the seasonality when multiple time series are present.	'global'
n_lags	*int*	Previous time series steps to include in auto-regression. Aka AR-order.	0
ar_reg	*float*	(Optional) How much sparsity to induce in the AR-coefficients.	None
ar_layers	*list[int]*	(Optional) array of hidden layer dimensions of the AR-Net. Specifies number of hidden layers (number of entries) and layer dimension (list entry).	[]
n_forecasts	*int*	Number of steps ahead of prediction time step to forecast.	1
lagged_reg_layers	*list[int]*	(Optional) array of hidden layer dimensions of the Covar-Net. Specifies number of hidden layers (number of entries) and layer dimension (list entry).	[]
learning_rate	*float*	(Optional) Maximum learning rate setting for 1cycle policy scheduler.	None
epochs	*int*	(Optional) Number of epochs (complete iterations over dataset) to train model.	None
batch_size	*int*	(Optional) Number of samples per mini-batch. If not provided, batch_size is approximated based on dataset size. For manual values, try ~8-1024. For best results also leave epochs to None.	None
newer_samples_weight	*float*	Sets factor by which the model fit is skewed towards more recent observations. Controls the factor by which final samples are weighted more compared to initial samples. Applies a positional weighting to each sample's loss value.	2

newer_samples_start	*float*	Sets beginning of 'newer' samples as fraction of training data. Throughout the range of 'newer' samples, the weight is increased from 1.0/*newer_samples_weight* initially to 1.0 at the end, in a monotonously increasing function (cosine from pi to 2*pi).	0.0
loss_func	*str, torch.nn.functional.loss*	Type of loss to use.	'AdamW'
collect_metrics	*list[str], dict, bool*	Set metrics to compute.	True
quantiles	*list*	A list of float values between (0, 1) which indicate the set of quantiles to be estimated.	[]
impute_missing	*bool*	Whether to automatically impute missing dates/values.	True
impute_linear	*int*	Maximal number of missing dates/values to be imputed linearly.	10
impute_rolling	*int*	Maximal number of missing dates/values to be imputed using rolling average.	10
drop_missing	*bool*	Whether to automatically drop missing samples from the data.	False
normalize	*str*	Type of normalization to apply to the time series.	'auto'
global_normalization	*bool*	Activation of global normalization.	False
global_time_normalization	*bool*	Specifies global time normalization.	True
unknown_data_normalization	*bool*	Specifies unknown data normalization.	False
accelerator	*str*	(Optional) Name of accelerator from pytorch_lightning.accelerators to use for training. Use "auto" to automatically select an available accelerator. Provide None to deactivate the use of accelerators.	None
trainer_config	*dict*	Dictionary of additional trainer configuration parameters.	{}
prediction_frequency	*dict*	(Optional) periodic interval in which forecasts should be made. More than one item only allowed for {"daily-hour": x, "weekly-day": y"} to forecast on a specific hour of a specific day of week.	None

Appendix 6: Temporal Fusion Transformer architecture (Lim et al., 2020)



Appendix 7: Temporal Fusion Transformer input parameters (Unit8 SA, 2023)

Input parameter	Data Type	Description	
input_chunk_length	*int*	Encoder length; number of past time steps that are fed to the forecasting module at prediction time.	
output_chunk_length	*int*	Decoder length; number of future time steps that are fed to the forecasting module at prediction time.	
hidden_size	*int, list[int]*	Hidden state size of the TFT. It is the main hyper-parameter and common across the internal TFT architecture.	16
lstm_layers	*int*	Number of layers for the Long Short Term Memory (LSTM) Encoder and Decoder (1 is a good default).	1
num_attention_heads	*int*	Number of attention heads (4 is a good default)	4
full_attention	*bool*	If True, applies multi-head attention query on past (encoder) and future (decoder) parts. Otherwise, only queries on future part.	False
feed_forward	*str*	A feedforward network is a fully-connected layer with an activation. TFT Can be one of the glu variant's FeedForward Network (FFN)[2]. The glu variant's FeedForward Network are a series of FFNs designed to	'GatedResidualNetwork'

		work better with Transformer based models.	
dropout	*float*	Fraction of neurons affected by dropout. This is compatible with Monte Carlo dropout at inference time for model uncertainty estimation (enabled with mc_dropout=True at prediction time).	0.1
hidden_continuous_size	*int*	Default for hidden size for processing continuous variables	8
categorical_embedding_sizes	*[Dict[str, Union[int, Tuple[int, int]]]]*	A dictionary used to construct embeddings for categorical static covariates. The keys are the column names of the categorical static covariates. Each value is either a single integer or a tuple of integers.	None
add_relative_index	*bool*	Whether to add positional values to future covariates. Defaults to False. This allows to use the TFTModel without having to pass future_covariates to fit() and train(). It gives a value to the position of each step from input and output chunk relative to the prediction point. The values are normalized with input_chunk_length.	False
loss_fn	*nn.Module*	PyTorch loss function used for training. By default, the TFT model is probabilistic and uses a likelihood instead (QuantileRegression). To make the model deterministic, you can set the ` likelihood` to None and give a loss_fn argument.	None
likelihood	*Likelihood*	The likelihood model to be used for probabilistic forecasts. By default, the TFT uses a QuantileRegression likelihood.	None
norm_type	*str, nn.Module*	The type of LayerNorm variant to use. Default: LayerNorm. Available options are [“LayerNorm”, “RMSNorm”, “LayerNormNoBias”], or provide a custom nn.Module.	‘LayerNorm’
use_static_covariates	*bool*	Whether the model should use static covariate information in case the input series passed to fit() contain static covariates. If True, and static covariates are available at fitting time, will enforce that all target series have the same static covariate	True

		dimensionality in fit() and ``predict()`.	
***kwargs*		Optional arguments to initialize the pytorch_lightning.Module, pytorch_lightning.Trainer, and Darts' TorchForecastingModel.	

Appendix 8: Experimentation results from the heuristics approach

		C4 *Group 1 Prototype*		C6 *Group 2 Prototype*		C1 *Group 3 Prototype*	
		Configuration	RMSE	Configuration	RMSE	Configuration	RMSE
SARIMAX	Full series	(1, 0, 1)x(0, 0, 0)	2.49E+03	(2, 1, 1)x(0, 1, 1)	1.87E+05	(1, 1, 0)x(0, 0, 0)	1.87E+05
	2 years	(1, 0, 1)x(0, 0, 0)	1.95E+03	(0, 1, 0)x(2, 1, 1)	1.83E+05	-	-
NeuralProphet (univariate)	Full series	$p = 24, l = 0$	2.78E+04	$p = 24, l = 0$	2.57E+05	$p = 0, l = 0$	1.30E+05
	2 years	p=12, l=0	1.93E+04	p=12, l=0	2.87E+05	-	-
NeuralProphet (multivariate)	Full series	$p = 24, l = 0$	1.14E+04	p=24, l=0	1.59E+05	-	-
	2 years	p=12, l=0	2.81E+04	p=12, l=0	3.31E+05	-	-
Temporal Fusion Transformer	Full series	$p = 24$, $hs = 128$, $epochs = 10k$	3.22E+03	p=24, hs=128, $epochs$=10k	1.80E+05	-	-
	2 years	$p = 24$, $hs = 128$, $epochs = 1k$	8.53E+03	$p = 24$, $hs = 128$, $epochs = 1k$	1.90E+05	-	-

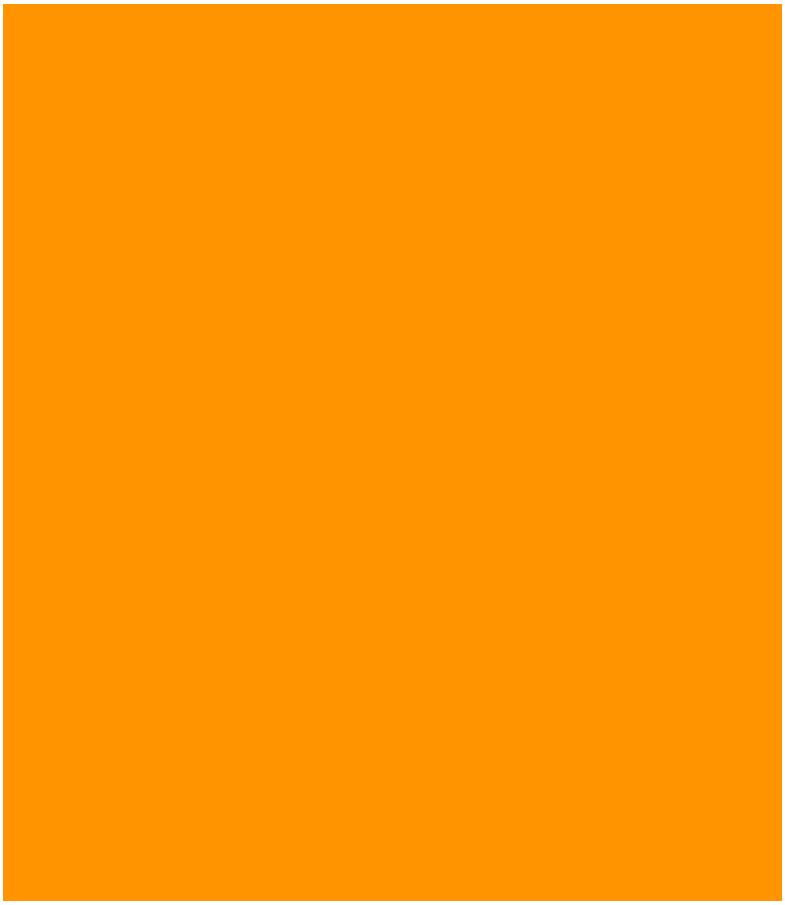
Experiments highlighted in green were taken over to results table in Chapter 4.

Appendix 9: Experimentation results from the optimization approach

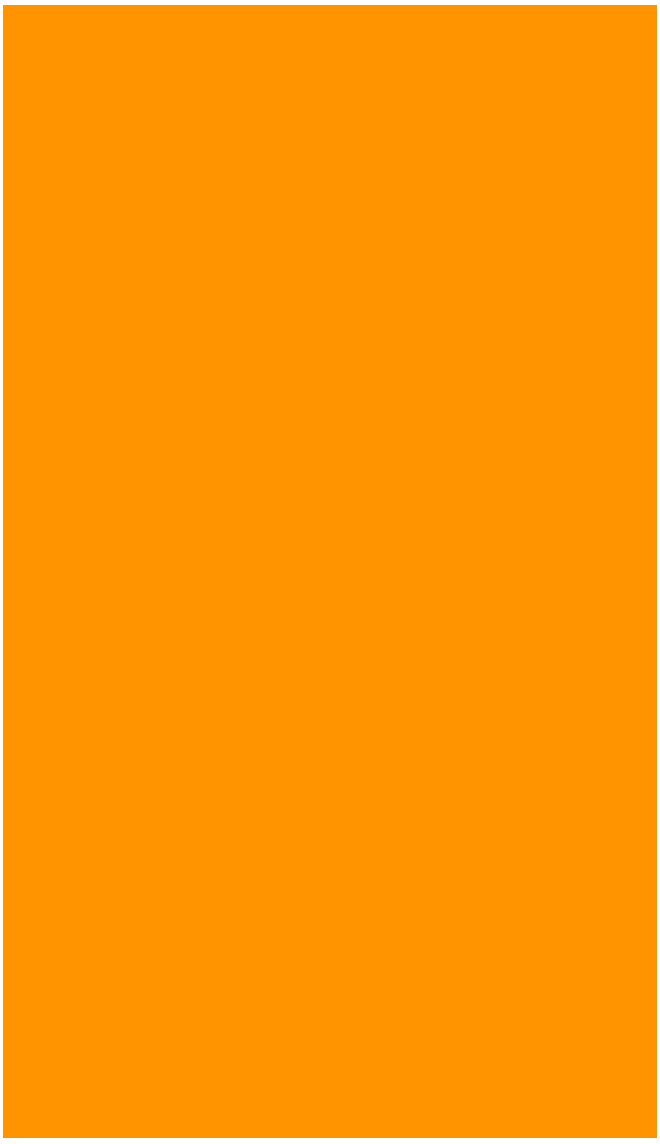
		C4 *Group 1 Prototype*		C6 *Group 2 Prototype*		C1 *Group 3 Prototype*	
		Configuration	RMSE	Configuration	RMSE	Configuration	RMSE
SARIMAX	Full series	(1, 0, 1)x(0, 0, 0)	2.49E+03	(2, 1, 1)x(0, 1, 1)	1.87E+05	-	-
	2 years	-	-	-	-	-	-
NeuralProphet (univariate)	Full series	$p = 24, l = 0$	2.78E+04	$p = 24, l = 0$	2.57E+05	-	-
	2 years	-	-	-	-	-	-
NeuralProphet (multivariate)	Full series	$p = 24, l = 0$	1.14E+04	p=24, l=0	1.59E+05	-	-
	2 years	-	-	-	-	-	-
Temporal Fusion Transformer	Full series	-	-	-	-	-	-
	2 years	-	-	-	-	-	-

Experiments highlighted in green were taken over to results table in Chapter 4.

Appendix 10: NeuralProphet parameters for C4 (using full-series data)



Appendix 11: NeuralProphet parameters for C6 (using full-series data)



Appendix 12: Parameters plot for C1