

Time Series Forecasting with Self-Adaptive Gaussian Process Regression

Florian Franz Xaver Haselbeck

Vollständiger Abdruck der vom TUM Campus Straubing für Biotechnologie und Nachhaltigkeit der Technischen Universität München zur Erlangung eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigten Dissertation.

Vorsitz: Prof. Dr. Bastian Blombach

Prüfer*innen der Dissertation:

1. Prof. Dr. Dominik Grimm
2. Prof. Dr. rer. pol. Alexander Hübner

Die Dissertation wurde am 27.06.2023 bei der Technischen Universität München eingereicht und durch den TUM Campus Straubing für Biotechnologie und Nachhaltigkeit am 07.11.2023 angenommen.

Abstract

Time series forecasting is a growing area of research with diverse applications, such as predicting the demand for a particular product or the health status of a patient. However, this research domain faces several challenges, some of which are addressed in this thesis. A major challenge is changing data distributions, i.e., the distribution of the data used to train a model may differ from that of the samples processed during inference. This can lead to inaccurate predictions, which can be problematic for downstream tasks and even cause damage, such as financial loss in demand forecasting. Therefore, approaches are needed that are specifically designed to handle changing data distributions and quickly adapt a prediction model.

In this thesis, we first assess the applicability of time series forecasting approaches despite changing data distributions for predicting demand for small and medium-sized companies dealing with perishable goods. Sales forecasting is currently mainly used by large enterprises, whereas lack of data and know-how are common challenges for smaller companies. For perishable items, improved operational decisions due to accurate predictions can lead to waste reduction besides financial benefits. That is why we chose horticulture as an example industry. Despite having a multi-billion dollar turnover in Germany alone and being heavily affected by the disposal of unsold items, this sector has received limited attention in forecasting research. In a first-time comparative study using horticultural retail sales data, we observe promising results, with the ensemble learner XGBoost showing the best performance. However, all prediction models are limited in their ability to handle a change in the data distribution. Finally, further research is needed to verify our results in a broader study allowing for more general conclusions and to overcome several obstacles that impede the practical operation of a forecasting system in this domain.

The development of novel online approaches to address changing data distributions is the main objective of this thesis. With respect to the prediction model, we focus on Gaussian Process Regression. We present the two already published approaches **EV**ent-Triggered **A**ugmented **R**efitting of Gaussian Process Regression for **S**easonal Data (EVARS-GPR) and **EV**ent-Triggered **K**ernel **A**ddjustments in Gaussian Process modeling (ETKA). We further introduce an unpublished extension of the former called EVARS-GPR+. All three approaches rely on online methods to detect a change in the data distribution during inference. Each time a change point is detected, we adjust the prediction model with the aim of providing up-to-date predictions at all times. Despite this similarity in terms of

change point-triggered model adaptation, the three approaches differ in several ways. While EVARS-GPR and EVARS-GPR+ focus on changes visible in the output scale of seasonal data, ETKA also considers other types of changes, for instance, a shift in periodicity, as well as non-seasonal data. The former two employ ChangeFinder for change point detection, while we adopt a cumulative sum-based approach using the model's prediction uncertainty for ETKA. To allow for rapid model adaptation, EVARS-GPR reuses existing data via augmentation, while keeping the hyperparameters unchanged. EVARS-GPR+ extends this augmented adjustment with a non-augmented refitting for less significant changes in the output scale. In contrast, ETKA adapts the kernel expression of the Gaussian Process using Adjusting Kernel Search, which also accounts for other types of distributional shifts. We show broad applicability using simulated data for both EVARS-GPR and ETKA. In addition, we demonstrate good predictive performance of both approaches using real-world data. Furthermore, EVARS-GPR+ outperforms EVARS-GPR on real-world data, suggesting that the small enhancement of non-augmented refittings is beneficial. There are several points of interest for future research based on our contributions, such as combining these approaches by identifying the type of distributional shift and respond accordingly.

Beyond that, several forecasting competitions and comparative studies have not yielded an overall predominant prediction model. Thus, multiple methods need to be re-evaluated for each forecasting task while ensuring reproducibility and comparability of results. To facilitate such comparative studies, even without expert knowledge, and to provide an easily extensible tool for model developers, we present ForeTiS. ForeTiS is a time series forecasting framework in Python that covers the entire pipeline from data pre-processing over feature engineering and hyperparameter optimization to model selection. Although the pipeline is fully automated using state-of-the-art approaches, e.g., Bayesian optimization for hyperparameter search, ForeTiS is highly customizable. As an additional benefit, our framework is designed for straightforward extension and quick benchmarking of novel approaches, ensuring their accessibility. We further support users with comprehensive and hands-on online documentation, including several video tutorials and step-by-step guides.

Zusammenfassung

Zeitreihenprognosen sind ein wachsendes Forschungsgebiet mit zahlreichen Anwendungsfällen, wie der Vorhersage der Nachfrage nach einem bestimmten Produkt oder des Gesundheitszustands eines Patienten. Dieses Forschungsfeld ist mit diversen Herausforderungen konfrontiert, von denen einige in dieser Arbeit aufgegriffen werden. Eine große Herausforderung sind Änderungen in der Datenverteilung, die dazu führen können, dass die Verteilung der Trainingsdaten von der Verteilung der Stichproben während der Inferenz abweicht. Dies kann ungenaue Vorhersagen und damit Probleme in nachgelagerten Prozessen zur Folge haben, wie zum Beispiel finanzielle Verluste bei fehlerhaften Nachfrageprognosen. Daher sind Ansätze erforderlich, die speziell für den Umgang mit Änderungen in der Datenverteilung und für die schnelle Anpassung eines Prognosemodells konzipiert sind.

In dieser Arbeit wird zunächst die Anwendbarkeit von Methoden der Zeitreihenprognose trotz sich ändernder Datenverteilungen für Nachfragevorhersagen bei kleinen und mittleren Unternehmen, die mit verderblichen Waren handeln, untersucht. Absatzprognosen werden derzeit vorwiegend von großen Firmen eingesetzt, während für kleinere Betriebe unter anderem fehlende Daten und mangelnde Expertise herausfordernd sind. Bei verderblichen Gütern kann eine Verbesserung der operativen Entscheidungen durch genaue Prognosen neben finanziellen Vorteilen auch eine Reduktion der Abfallmengen bedeuten. Als Anwendungsbeispiel wurde in der vorliegenden Arbeit der Gartenbau gewählt. Obwohl diese Branche allein in Deutschland einen Umsatz in Milliardenhöhe erwirtschaftet, wurde sie bisher in der Zeitreihenforschung kaum betrachtet. In einer ersten Vergleichsstudie anhand von Einzelhandelsdaten wurden vielversprechende Ergebnisse erzielt, wobei die Ensemblemethode XGBoost die besten Prognosen liefert. Jedoch sind alle Prognosemodelle nur begrenzt in der Lage, mit Änderungen in der Datenverteilung umzugehen. Für allgemeinere Schlussfolgerungen und zur Adressierung weiterer Herausforderungen bezüglich des Betriebs eines Vorhersagesystems im Gartenbau sind weitere Forschungsarbeiten erforderlich.

Die Entwicklung neuer Methoden, die auf Änderungen in der Datenverteilung reagieren, ist das primäre Ziel dieser Arbeit. Als Prognosemodell wurde die Gaußprozess-Regression gewählt. In diesem Kontext werden die beiden bereits publizierten Methoden **Event-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data** (EVARS-GPR) und **Event-Triggered Kernel Adjustments in Gaussian Process modeling** (ETKA) vorgestellt. Außerdem wird eine bisher unveröffentlichte Erweiterung des erstgenannten Ansatzes, EVARS-GPR+, präsentiert. Alle drei Methoden basieren auf einer Online-Erkennung von

Änderungen in der Datenverteilung während der Inferenz. Wird eine Änderung erkannt, wird das Prognosemodell angepasst, um stets aktuelle Vorhersagen zu gewährleisten. Trotz dieser Ähnlichkeit hinsichtlich der Initiierung einer Modellanpassung unterscheiden sich die drei Algorithmen in mehrfacher Hinsicht. EVARS-GPR und EVARS-GPR+ sind für Änderungen in der Skala der Zielvariable saisonaler Daten konzipiert. ETKA hingegen berücksichtigt auch andere Arten von Änderungen, wie zum Beispiel eine Verschiebung der Periodizität, sowie nicht-saisonale Daten. Die ersten beiden verwenden ChangeFinder zur Erkennung von Änderungen, wohingegen ETKA einen auf der kumulativen Summe basierenden Ansatz nutzt, der die Vorhersageunsicherheit des Modells berücksichtigt. Um eine schnelle Modellanpassung zu ermöglichen, verwendet EVARS-GPR vorhandene Daten mittels Augmentierung wieder, während die Hyperparameter unverändert bleiben. EVARS-GPR+ erweitert diese augmentierte Anpassung um eine nicht-augmentierte Nachjustierung für weniger signifikante Änderungen des Wertebereichs der Zielvariable. Im Gegensatz dazu adaptiert ETKA den Kernausdruck des Gaußprozesses mit Hilfe der Adjusting Kernel Search, wodurch auch andere Arten von Verteilungsänderungen berücksichtigt werden. Anhand von simulierten Daten wird eine breite Anwendbarkeit sowohl für EVARS-GPR als auch für ETKA demonstriert. Zudem zeigen beide Methoden gute Resultate für reale Datensätze. EVARS-GPR+ übertrifft EVARS-GPR auf realen Daten, was darauf hindeutet, dass die Erweiterung um nicht-augmentierte Modellanpassungen vorteilhaft ist. Basierend auf diesen Erkenntnissen ergeben sich mehrere Ansatzpunkte für zukünftige Forschung, wie beispielsweise die Kombination dieser Methoden, um mit Hilfe einer Erkennung der Art der Verteilungsänderung entsprechend zu reagieren.

Darüber hinaus konnte in verschiedenen Studien kein führendes Prognosemodell identifiziert werden. Daher müssen für jede Prognoseaufgabe mehrere Methoden evaluiert werden, wobei die Reproduzierbarkeit und Vergleichbarkeit der Ergebnisse gewährleistet sein muss. Um solche Vergleichsstudien zu vereinfachen und Modellentwicklern ein einfach zu erweiterndes Werkzeug zur Verfügung zu stellen, wurde `ForeTiS` entwickelt. `ForeTiS` ist ein Zeitreihenvorhersage-Framework in der Programmiersprache Python, das den gesamten Prozess von der Datenvorverarbeitung über die Merkmalsbestimmung und Hyperparameteroptimierung bis hin zur Modellauswahl umfasst. Obwohl der Ablauf vollständig automatisiert ist und moderne Ansätze, wie Bayes'sche Optimierung für die Hyperparametersuche, verwendet werden, ist `ForeTiS` in hohem Maße anpassbar. Weiterhin ist das Framework leicht erweiterbar, so dass neue Methoden schnell getestet werden können und leicht zugänglich sind. Zusätzlich bietet eine umfangreiche Online-Dokumentation, einschließlich mehrerer Video-Tutorials und Schritt-für-Schritt-Anleitungen, Unterstützung.

List of Publications

Main Publications of this Thesis

PUBLICATION A:

Florian Haselbeck, Jennifer Killinger, Klaus Menrad, Thomas Hannus, and Dominik G. Grimm (2022). Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions. *Machine Learning with Applications*, 7, 100239. <https://doi.org/10.1016/j.mlwa.2021.100239>

PUBLICATION B:

Florian Haselbeck and Dominik G. Grimm (2021). EVARS-GPR: EVent-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data. *KI 2021: Advances in Artificial Intelligence*, 12873, 135–157. https://doi.org/10.1007/978-3-030-87626-5_11

PUBLICATION C:

Jan D. Hüwel*, **Florian Haselbeck***, Dominik G. Grimm and Christian Beecks (2022). Dynamically Self-adjusting Gaussian Processes for Data Stream Modelling. *KI 2022: Advances in Artificial Intelligence*, 13404, 96-114. https://doi.org/10.1007/978-3-031-15791-2_10

*Both authors contributed equally and share first authorship.

PUBLICATION D:

Josef Eiglsperger*, **Florian Haselbeck*** and Dominik G. Grimm (2023). ForeTiS: A comprehensive time series forecasting framework in Python. *Machine Learning with Applications*, 12, 100467. <https://doi.org/10.1016/j.mlwa.2023.100467>

*Both authors contributed equally and share first authorship.

Further Publications

Florian Haselbeck*, Maura John* and Dominik G. Grimm (2023). easyPheno: An easy-to-use and easy-to-extend Python framework for phenotype prediction using Bayesian optimization. *Bioinformatics Advances*, 3. <https://doi.org/10.1093/bioadv/vbad035>

**Both authors contributed equally and share first authorship.*

Maura John*, **Florian Haselbeck***, Rupashree Dass, Christoph Malisi, Patrizia Ricca, Christian Dreischer, Sebastian J. Schultheiss and Dominik G. Grimm (2022). A comparison of classical and machine learning-based phenotype prediction methods on simulated data and three plant species. *Frontiers in Plant Science*, 13. <https://doi.org/10.3389/fpls.2022.932512>

**Both authors contributed equally and share first authorship.*

Florian Haselbeck, Maura John, Yuqi Zhang, Jonathan Pirnay, Juan Pablo Fuenzalida-Werner, Rubén D. Costa and Dominik G. Grimm (2023). Superior Protein Thermophilicity Prediction With Protein Language Model Embeddings.
currently under review

Nikita Genze, Raymond Ajekwe, Zeynep Güreli, **Florian Haselbeck**, Michael Grieb, Dominik G. Grimm (2022). Deep learning-based early weed segmentation using motion blurred UAV images of sorghum fields. *Computers and Electronics in Agriculture*, 202(C), 107388. <https://doi.org/10.1016/j.compag.2022.107388>

Acknowledgments

First and foremost, I would like to sincerely thank my supervisor, Prof. Dr. Dominik Grimm, for his professional advice and for creating a working atmosphere that enabled me to become a better researcher. In particular, I am very grateful for his personal support, great dedication and outstanding supervision. Despite his high workload, he always had an open ear for me. Besides Dominik, I would like to thank all my colleagues at the GrimmLab, namely Dr. Richa Bharti, Josef Eiglsperger, Anna Fischer, Nikita Genze, Maura John, Ingrid Meindl, Dr. Sara Omranian and Jonathan Pirnay. I am grateful for the supportive and great working atmosphere, where everyone is happy about the successes of the other team members and contributes equally to the responsibilities of the professorship. In particular, I would like to thank Maura John and Josef Eiglsperger for proofreading this thesis and for a successful and enjoyable collaboration on several research papers.

I would like to thank all students who contributed to my research during their student assistant jobs, research internships and theses: Atilla Can Basaran, Vitalii Dmytrenko, Josef Eiglsperger, Katarina Golubovic, Claudia Guadarrama, Selina Kanamüller, Jennifer Killinger, Kelly Lim-Trinh, Qendrim Maxhuni, Ana Pecini, Danelle Widjaja and Yuqi Zhang. Furthermore, I would like to thank all co-authors of the research papers included as main contributions to this thesis, namely Prof. Dr. Christian Becks, Josef Eiglsperger, Prof. Dr. Dominik Grimm, Prof. Dr. Thomas Hannus, Jan Hüwel, Jennifer Killinger and Prof. Dr. Klaus Menrad. In addition, I would like to thank Prof. Dr. Thomas Hannus and Viola Stiele for supporting my research with their expertise in the field of horticulture.

I would like to thank my employer, the University of Applied Sciences Weihenstephan-Triesdorf, and the TUM Campus Straubing for hosting our research group. Moreover, I would like to thank Dr. Thomas Decker and Prof. Dr. Klaus Menrad for the administration and coordination of the research project in which I was initially involved, as well as all the people, universities and companies that contributed to the project. I would further like to acknowledge the Leibniz Supercomputing Centre for providing computing resources.

Beyond that, I would like to thank the Graduate School of the Technical University of Munich, especially the Graduate Center at the TUM Campus Straubing, for giving me the opportunity to do my PhD at this world-renowned university. I would like to thank Prof. Dr. Alexander Hübner for being the second reviewer of my thesis and Prof. Dr. Bastian Blombach for his support as chairman of the examination committee. I would

like to thank Dr. Marina Zapilko for her support in organizational matters concerning the Graduate Center. Furthermore, I would like to thank my former colleague and friend, Prof. Dr. Torsten Schön, for mentoring me throughout my PhD.

I would like to thank my friends for providing useful distractions and enjoyable events during my PhD. In addition, I would like to thank my parents Renate and Franz, my brother Christian, and my parents-in-law Gudrun and Heribert for supporting my family and me during my PhD, especially during the construction of our house and for taking care of our son Paul. Moreover, I would like to thank my son Paul for bringing a lot of joy into my life and for being a perfect encouragement even on disappointing days. I would like to thank my unborn son for the joyful anticipation of our life together and for setting me a strict deadline to finish this thesis. Finally, I would like to thank the love of my life, Lena, simply for everything.

Contents

1	Introduction	1
1.1	An Introduction to Time Series Forecasting	2
1.2	Current Challenges in Time Series Forecasting	5
1.3	Related Work	8
1.3.1	Demand Forecasting in Domains Similar to Horticulture	8
1.3.2	Approaches Addressing Changing Data Distributions	10
1.3.3	Time Series Forecasting Frameworks	12
1.4	Objectives of this Thesis	14
1.5	Contributions	16
1.5.1	Main Contributions	16
1.5.2	Further Contributions	18
2	Material and Methods	21
2.1	Classical Time Series Forecasting Methods	21
2.2	Frequentist Machine Learning for Time Series Forecasting	23
2.3	Probabilistic Machine Learning for Time Series Forecasting	25
2.3.1	Bayesian Regression and Bayesian Neural Networks	26
2.3.2	Gaussian Process Regression	27
2.4	Model Selection and Evaluation	34
2.4.1	Data Splits	34
2.4.2	Hyperparameter Optimization	35
2.4.3	Evaluation Metrics and Baselines	38
2.5	Change Point Detection	39
3	Results	43
3.1	Time Series Forecasting for Small and Medium-Sized Companies	43
3.1.1	Summary	44
3.1.2	Key Findings	47

3.2	Event-Triggered Augmented Refitting of Gaussian Process Regression . . .	48
3.2.1	Summary	48
3.2.2	Further Unpublished Results	50
3.2.3	Key Findings	53
3.3	Dynamically Self-Adjusting Gaussian Processes	53
3.3.1	Summary	54
3.3.2	Key Findings	55
3.4	A Comprehensive Time Series Forecasting Framework	56
3.4.1	Summary	56
3.4.2	Key Findings	58
4	Discussion	59
4.1	Time Series Forecasting for Small and Medium-Sized Companies	59
4.2	Novel Approaches for Changing Data Distributions	64
4.3	A Comprehensive Time Series Forecasting Framework	70
4.4	Further Points of Discussion	72
5	Conclusion	75
A	Publication A	77
B	Publication B	97
C	Publication C	121
D	Publication D	141
	List of Symbols	149
	List of Acronyms	159
	List of Figures	163
	List of Tables	165
	Bibliography	167

Introduction

Time series forecasting is a research area with diverse application domains, such as predicting energy demand and production, demand for a certain product, or a patient's health status. The availability of data due to increasing digitalization and the recognition of the power of accurate forecasts by various stakeholders, for instance, commercial enterprises and governments, have led to continued growth in this research area (Ahmad et al., 2020; Deb et al., 2017; Hobensack et al., 2023; Hong et al., 2019; Ingle et al., 2021; Liu & Chen, 2019; Mediavilla et al., 2022; Rajkomar et al., 2018; Sharadga et al., 2020; Yasrebi-de Kom et al., 2023; Zhang et al., 2021). Accurate predictions of future developments, e.g., the demand for a particular product, enable early interventions such as increasing or decreasing production and adjusting procurement plans. Consequently improved operational decisions are a potential competitive advantage, for instance, leading to higher revenues when meeting an increasing demand or to lower costs when responding early to a decrease in demand (Ivanov et al., 2019). As another example, accurate predictions of energy consumption and renewable energy production can support the transition to sustainable energy sources while ensuring energy security by enabling early adaptation in a smart grid (Ibrahim et al., 2020). In addition, there are use cases in healthcare where time series forecasting techniques could help improve patient treatment, for example, based on electronic health records (Tomašev et al., 2021).

However, time series forecasting faces several challenges, which we outline in Section 1.2, and some of which we address in this thesis. A major challenge in time series forecasting is changing data distributions, i.e., the data distribution from model training and inference during live operation of a forecasting system are different. With respect to the potential

benefits of time series forecasting described above, relying on a forecasting model trained on an outdated data distribution would be problematic. For example, for demand forecasting, such an inaccurate prediction model could result in financial loss due to overstocking or understocking. Moreover, in the case of smart grid applications, inaccurate forecasts could lead to security of energy supply and grid stability problems. Therefore, changing data distributions are an important yet unsolved challenge in time series forecasting (Ditzler et al., 2015; Gama, 2012; Lu et al., 2019; Rossi, 2013; Žliobaitė et al., 2016). The focus of this thesis is on self-adaptive Gaussian Process Regression (GPR), i.e., accurate and computationally efficient approaches that enable GPR to quickly adapt online to such distributional shifts. We also address further challenges in time series forecasting besides changing data distributions, which is described in more detail in Section 1.4.

In the following, we first give an introduction to time series forecasting. We then outline current challenges in the field, before reviewing related work that is relevant to the challenges we address. Finally, we describe the objectives and contributions of this thesis. After this introduction, we outline the material and methods relevant to this work in Chapter 2, including classical and machine learning (ML) time series forecasting approaches with a focus on GPR as the main prediction model in this thesis. We further describe model selection and evaluation techniques as well as change point detection (CPD) approaches. In Chapter 3, we present the results of this thesis. This chapter consists of summaries of four research papers, which are the main contributions of this thesis and are given in Section 1.5.1. We then discuss the results of these papers and some more general aspects relevant to this thesis in Chapter 4, before concluding in Chapter 5.

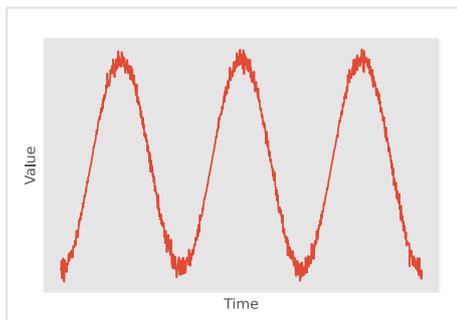
1.1 An Introduction to Time Series Forecasting

In time series forecasting, we consider a dataset $\mathcal{D} = \{(x_t, y_t) \mid t = 1, \dots, n\}$ containing $n \in \mathbb{N}$ pairs of a feature vector $x_t \in \mathbb{R}^m$ consisting of $m \in \mathbb{N}$ feature values and a target value $y_t \in \mathbb{R}$ at time step $t \in \mathbb{N}$. We assume that the target variable $\mathbf{y} \in \mathbb{R}^n$ is a time series, i.e., it can be defined as a chronologically ordered sequence of data points y_t observed at successive time steps $t = 1, \dots, n$, often recorded at constant intervals. Our goal is to predict a future value $\hat{y}_{t+h} \in \mathbb{R}$ of the target variable \mathbf{y} with a given forecast horizon $h \in \mathbb{N}$. To determine these predictions, we can use past values of the target variable \mathbf{y} itself, as well as m external features that influence the target values, denoted by $\mathbf{X} \in \mathbb{R}^{n \times m}$ for n samples. For example, if we consider ice cream sales as the target variable, weather-related and calendrical information, such as holidays or weekdays, may

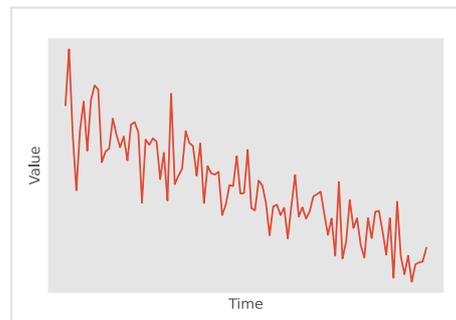
be predictive features. Besides the target variable y itself, some of the features can be a time series as well, for example daily weather data (Brockwell & Davis, 2016).

A time series may contain certain patterns that describe its behavior. A common pattern is seasonality, i.e., the time series is influenced by factors that occur with a known and fixed periodicity, such as the day of the week. Another typical component is trend, which reflects a long-term increase or decrease in the time series values, often following a linear behavior. These components can be used to model a time series. Furthermore, we may observe a cyclic behavior, i.e., the values of the time series increase and decrease with varying frequency.

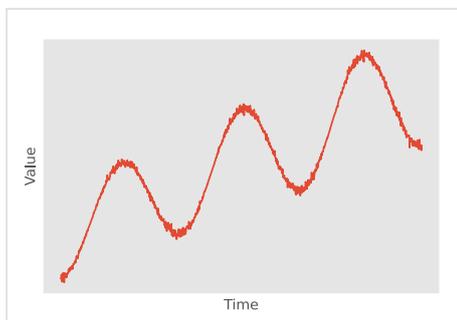
(a) Seasonal pattern



(b) Negative trend pattern



(c) Seasonal with positive trend pattern



(d) Multi-seasonal pattern

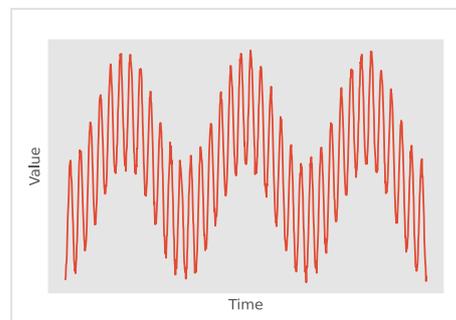


Figure 1.1: **Examples of time series patterns:** All examples include additive noise. (a) Seasonal behavior. (b) Negative trend pattern without seasonality. (c) Seasonal pattern combined with positive linear trend. (d) Multi-seasonal behavior.

Figure 1.1 visualizes some of these common time series patterns. For example, Figure 1.1a shows a seasonal behavior and Figure 1.1b a negative trend pattern. In Figures 1.1c and d, we can see combinations of patterns, i.e., seasonality with a linear increasing trend as well as a multi-seasonal behavior. An example of the latter is ice cream sales with annual in addition to weekly seasonality. When identifying such patterns, the number of

available data points should be considered. For instance, the negative trend depicted in 1.1b could also be a downward part of a long cyclical or seasonal behavior (Hyndman & Athanasopoulos, 2021).

With respect to the operation of a time series forecasting system, we can distinguish between the offline and the online phase. During the offline phase, we collect data to build our prediction model. Then, we use this prediction model to predict future values during the online phase. Usually, new samples become available in this stage that we could employ to refine our model. Several approaches can be used for time series prediction. In this thesis, we consider classical time series forecasting methods, e.g., Exponential Smoothing and Autoregressive Integrated Moving Average (ARIMA), see Section 2.1, as well as ML-based approaches described in Sections 2.2 and 2.3. Regarding the former, we include univariate models that use only the target time series y , besides multivariate approaches, which additionally leverage features X that influence the target variable.

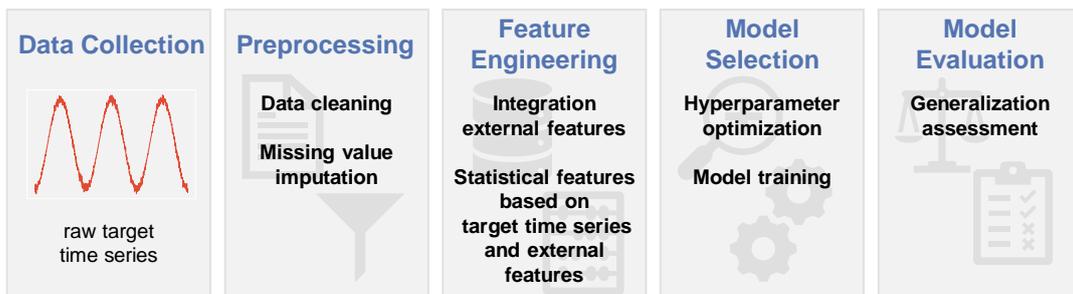


Figure 1.2: **A common procedure in a time series forecasting project:** After collecting the raw target time series, the data must be cleaned and missing values may need to be imputed. Then, in the case of a multivariate prediction model, feature engineering must be performed. Using this dataset, model selection, including hyperparameter optimization and model training, can be performed to finally select a model with the best hyperparameter configuration based on the results on validation data, see Section 2.4 for more details. Finally, we can evaluate the generalization ability of the model on unseen data.

We visualize a typical procedure in a time series forecasting project in Figure 1.2. First, we need to collect historical data on the target time series. Then, we may need to clean up the data, e.g., handle outliers and missing values. For instance, we may want to impute missing values by employing methods such as mean and k-Nearest Neighbor (k-NN) imputation. In the next step, we can add features that influence the target variable to enable the application of multivariate prediction models. For external features, we may also need to collect historical observations, e.g., when integrating weather data. We can further infer

additional features, i.a., statistical information about these external factors and the target variable. For example, past sales of a certain product may be relevant to future demand. Some of these features may also require a pre-processing step, e.g., due to outliers or missing values. Depending on the number of features and samples, we may additionally include dimensionality reduction approaches, such as Principal Component Analysis (PCA). With the resulting dataset, we can perform the hyperparameter optimization and training for each prediction model we consider in a study. Using these results, we can select the best-performing model with its best-fitting hyperparameters on the basis of validation data. For more details on this step, see Section 2.4. Finally, we can evaluate the generalization ability of the model on unseen data. In time series forecasting, this can be combined with simulating the live operation of a forecasting system with continuously incoming new data and potential model updates (Hyndman & Athanasopoulos, 2021; Meisenbacher et al., 2022).

1.2 Current Challenges in Time Series Forecasting

Time series forecasting faces several challenges, some of which are outlined in the following. A major issue in time series forecasting, and the focus of this thesis, is changing data distributions. In ML, we usually aim to build a well-generalizing model based on a dataset that reflects the entire system behavior. Therefore, it is common to train a prediction model on data collected offline and use it to predict a certain outcome based on unseen data, assuming that the data distribution does not change. However, this assumption is often not true for time series forecasting. Sudden changes in system behavior are a common problem that can be caused by external or internal influences. For example, operational or strategic decisions may affect sales of a certain product. In addition, external factors, such as the outage of a major consumer in the case of energy demand forecasting or the recent SARS-CoV-2 pandemic affecting sales in various industries, may be influential. Some of these factors can be considered as features when building a forecasting system, but probably not all of them, e.g., due to lack of historical data, unavailability of a continuous data source, or when an influence is unknown in the first place. A change in the generative data distribution can lead to an outdated prediction model. Continuing to use such a prediction model during live operation of a forecasting system and deriving operational decisions based on it could cause damage, for instance, financial loss in case of a misestimation of demand. We can distinguish between different types of data distribution changes, although there are varying definitions in the literature. For example, concept drift can be defined as

a change in the joint probability distribution of the features and the target variable, and data or covariate drift refers to a change in the distribution of a model's inputs, i.e., the features. Since in both cases an adaptation of the prediction model is required, we refer to these phenomena as changing data distributions or distributional shifts. Detecting and adapting to such a change in system behavior during the online phase is difficult. With respect to detection, we are in a trade-off between preventing false alarms, e.g., in case of outliers, and ensuring a quick reaction of the forecasting system. Furthermore, if we react early, we only have a few samples of the new data distribution available to adapt the prediction model. Therefore, handling changes in the data distribution, especially during the online phase, requires specific approaches that take these circumstances into account (Ditzler et al., 2015; Gama, 2012; Lu et al., 2019; Rossi, 2013; Žliobaitė et al., 2016).

An application of time series forecasting with great potential is demand prediction. Accurate predictions of future demand enable early operational decisions to meet increases in demand or reduce costs in the event of a decrease. Currently, demand forecasting is mainly used by large enterprises such as Amazon and Walmart (Böse et al., 2017; Fildes et al., 2022; Seaman & Bowman, 2022). However, especially for small and medium-sized companies, which often have less financial flexibility, the accurate adjustment of procurement and production can be a significant competitive advantage. Thus, time series forecasting is an opportunity for such companies, but there are often challenges, e.g., a lack of know-how and budget. Moreover, many state-of-the-art approaches such as DeepAR require large amounts of data that may not be available (Benidis et al., 2022; Salinas et al., 2020). It is particularly important to be able to adapt procurement and production quickly and early when working with perishable goods that can only be kept in stock for a short period of time. For such products, the disposal of unsold items in overstock situations results in financial loss as well as environmental damage due to wasted resources in production and transportation (Duan et al., 2012). The challenge of changing data distributions described above can be especially problematic for smaller companies with fewer financial resources. Financial losses due to inaccurate forecasts may be harder for these companies to deal with, in particular if storing unsold items is not an option due to perishability. A domain that is highly affected by these issues is horticulture. Most horticultural businesses are small and medium-sized, and this sector deals with perishable goods such as ornamental plants (Bundesministerium für Ernährung und Landwirtschaft, 2021). Furthermore, demand for horticultural products tends to be highly seasonal, subject to sudden changes, and influenced by several external factors, for instance, holidays and weather. These properties make manual forecasting difficult, and in addition, both external and internal factors can

cause changing data distributions. Even though the horticultural industry generates billions in annual sales in Germany alone (Zentralverband Gartenbau e.V., 2022), there are no scientific publications using time series forecasting approaches to predict horticultural sales. Thus, horticulture is a good example to evaluate time series forecasting for sales prediction in small and medium-sized companies that deal with perishable products and for which changing data distributions are particularly challenging.

As outlined in Section 1.1, multiple classical as well as ML-based approaches for time series forecasting exist. However, several forecasting competitions and comparative studies have not resulted in an overall predominant prediction method (Bojer & Meldgaard, 2021; Makridakis & Hibon, 2000; Makridakis et al., 2018, 2022). Thus, a re-evaluation of various prediction models is required for every new time series forecasting task. For this purpose, an easy-to-use and easy-to-extend comprehensive time series forecasting framework would be beneficial for both end users and model developers. Besides simplifying the application of state-of-the-art time series forecasting approaches, such a framework can help to ensure reproducibility and comparability of results through a unified pipeline.

The example described above in Figure 1.1b, where a negative trend pattern could be incorrectly identified due to a lack of data, shows that the amount of data collected offline can affect the quality of the initial prediction model. When considering the data collection process for a time series forecasting project, an additional challenge is the need for historical observations of the system behavior. If not already recorded, it may even take years to collect a meaningful number of samples in the case of a time series with annual seasonality. Consequently, the start of a time series forecasting project would also be delayed (Hyndman & Athanasopoulos, 2021; Shumway & Stoffer, 2000).

A well-known and often neglected problem in demand forecasting is that typically sales figures are collected, which can be used to only approximate demand. However, the true demand is not known because it may have been higher, for example, in out-of-stock situations. If stock quantities are recorded, they could be used to better estimate the real demand, e.g., by including them in the forecasting system or by imputing out-of-stock situations, but still without knowing the exact demand figures (Fildes et al., 2022). In this thesis, we use the terms sales and demand forecasting as synonyms, since stock recordings to distinguish between both are not available for the data we employ.

1.3 Related Work

In the following, we present related work with respect to the current challenges that are addressed in this thesis. Predicting horticultural demand, as outlined, is a good example to assess the challenges and applicability of time series forecasting for small and medium-sized companies dealing with perishable products. Furthermore, horticultural sales are affected by various internal and external factors that can cause a change of the data distribution, making this domain suitable to assess this challenge on real-world data. Since there are no scientific publications on horticultural demand forecasting, domains with similar characteristics could provide some guidance. Therefore, we first provide an overview of time series forecasting applications for predicting food and tourism demand. We then describe several approaches specifically designed for changing data distributions, the main challenge addressed in this thesis. Finally, we provide information on currently available time series forecasting frameworks that allow the use of multiple prediction models while ensuring reproducibility and comparability of results.

1.3.1 Demand Forecasting in Domains Similar to Horticulture

Similar to horticulture, food companies often deal with perishable items such as baked goods or agricultural products, e.g., vegetables and fruits. In addition to perishability, the demand for many of these products is often also seasonal (Tsoumakas, 2019). In contrast to horticulture, several comparative studies have been published evaluating classical forecasting and ML-based approaches for predicting food demand. In addition to the market size, food enterprises are often bigger than horticultural companies, which may explain why they receive more attention in forecasting research. For more details regarding the subsequently mentioned prediction methods, see Sections 2.1, 2.2, and 2.3. Shukla and Jharkharia (2011) evaluate the applicability of time series forecasting methods to predict sales of onions in an Indian market using the univariate classical model ARIMA. With the seasonal extension of ARIMA, Seasonal Autoregressive Integrated Moving Average (SARIMA), Sankaran (2014) achieves lower percentage errors when trying to forecast sales figures of onions in wholesale. In contrast to these studies, Arunraj and Ahrens (2015) and Arunraj et al. (2016) consider external factors in a Seasonal Autoregressive Integrated Moving Average with exogenous factors (SARIMAX) model to forecast retail sales of bananas in a German supermarket. The external factors are modeled using multiple linear regression and quantile regression, respectively, and are combined with the regular SARIMA model. In their study, the SARIMA model combined with quantile regression outperforms

a Multilayer Perceptron (MLP). Žliobaitė et al. (2012) take into account that some time series are difficult or even impossible to predict because they follow a random pattern. Thus, they develop a two-step approach. They first categorize time series into predictable and random. Then, they use a moving average for the latter and train a linear regression, k-NN, or regression tree to predict the former. In a more recent comparative study, Priyadarshi et al. (2019) include a recurrent neural network with Long Short-Term Memory (LSTM) cells, a classical ML-based technique, namely Support Vector Regression (SVR), and the ensemble methods Random Forest and XGBoost. The authors aim to predict daily sales of tomatoes, potatoes, and onions and observe that the LSTM network and SVR perform best. This result is in contrast to (Turgut & Erdem, 2022), where XGBoost performs better than an LSTM network and SARIMA in predicting vegetable and fruit sales.

While fruits and vegetables usually have a shelf life of a few days or weeks, bakery products, such as rolls and pretzels, often have to be sold within a few hours. Typically, large bakeries deliver to stores on a daily basis, but often articles need to be prepared for sale in the store, e.g., frozen items need to be baked. Thus, besides daily sales forecasts, intraday demand is highly relevant for scheduling this preparation (Huber, 2019). Huber et al. (2017) propose a decision support system employing Autoregressive Integrated Moving Average with eXogenous factors (ARIMAX) to forecast sales in a bakery chain. The authors conclude that predicting sales in clusters based on intraday patterns containing substitutional products provides similar results to item-by-item forecasting with less computational cost. In another publication focusing on sales on special days such as holidays, Huber and Stuckenschmidt (2020) observe advantages for the neural network-based approaches MLP and LSTM for time series with a history of more than 150 days. For an extension of their work with the goal of retrieving daily and intraday demand forecasts, they employ also an LSTM network. In addition to internal sales data, they include external factors such as calendar and weather information, as well as features characterizing the sales location. On a dataset of 14 products, nine stores, and 987 days, an LSTM network consistently performs best (Huber, 2019; Huber & Stuckenschmidt, 2021). Yang and Sutrisno (2018) utilize early morning sales to predict demand for bakery products for the remainder of the day, concluding that an MLP performs best. In an earlier study, Doganis et al. (2006) use a radial basis function neural network with a genetic algorithm to predict a dairy company's fresh milk sales. Liu and Ichise (2017) combine an LSTM network with an autoencoder, i.e., a second neural network, to predict sales of beverages as an example of a product for which meteorological data strongly influence demand. Their method outperforms an LSTM network and gradient boosted trees, among others.

Both in tourism and horticulture, demand is characterized by strong seasonality with sudden changes and influenced by several external factors such as weather and holidays. Thus, tourism is a domain that could provide insights for horticultural sales forecasting. Athanasopoulos et al. (2011) describe a forecasting competition with 366 monthly, 427 quarterly and 518 annual time series restricted to the tourism sector. In this competition, including univariate approaches and methods using explanatory features, the former deliver a better performance. In a more recent review paper, Jiao and Chen (2019) provide an overview of methodological improvements in tourism forecasting from 2008 to 2017, reviewing 72 studies published during this period. The authors also identify that classical time series forecasting approaches are still widely used, while they observe a trend towards incorporating external features, e.g., online search engine data. Furthermore, ML-based approaches became more popular, often in an ensembled form. Several other review papers on tourism demand forecasting exist, i.a., (Witt & Witt, 1995), (Song & Li, 2008), (Peng et al., 2014), (Wu et al., 2017), and (Song et al., 2019). Song et al. (2019) present a comprehensive review based on 211 papers published between 1968 and 2018, focusing on the most influential publications in terms of citations. The authors also observe a trend towards hybrid models combining different approaches and an increasing use of ML-based approaches, especially due to the increase of online data. However, they also conclude that there is no overall predominant prediction method.

1.3.2 Approaches Addressing Changing Data Distributions

Since changing data distributions are a major challenge in time series forecasting, approaches are needed that specifically address this problem. A common and straightforward solution is to periodically refit a prediction model with sequentially incoming new data samples. An initial prediction model is first trained and optimized using historical data. Then, as new samples become available during the online phase, this prediction model is periodically retrained at regular intervals, either by keeping the model's hyperparameters fixed or by even performing a hyperparameter optimization. For this retraining, old samples may be discarded in order to focus on newer ones and to lower the runtime. However, periodical refitting can be computationally exhaustive depending on the retraining frequency, especially in the case of unnecessary model updates when the data distribution has not changed. On the other hand, less frequent refittings would save computational resources but could lead to extended periods of using an outdated prediction model during the live operation of a forecasting system (Huber, 2019; Hyndman & Athanasopoulos, 2021).

With respect to other methods for handling changing data distributions, we focus on approaches that use GPR, as this is the prediction model that is used in the contributions of this thesis addressing this challenge. We have chosen GPR, which is described in more detail in Section 2.3.2, because of its many advantages such as its inherent provision of prediction uncertainties and its good performance in a first comparative study, see Section 3.1. Periodical refitting as described above is also possible for GPR, either on a certain window of current samples or on the whole dataset, and with fixed or even optimized hyperparameters (Perez-Cruz et al., 2013). Garnett et al. (2009) introduce an additional hyperparameter in the covariance function of a Gaussian Process (GP) model to account for change points, i.e., time steps at which the data distribution changes. However, their approach assumes a single change point within a pre-defined window size. Saatçi et al. (2010) use GPs within Bayesian Online Change Point Detection (BOCPD), primarily to accurately detect change points online. A common technique for finding a kernel, one of the most important components of a GP model, is the Compositional Kernel Search (CKS), see Section 2.3.2. This search algorithm has been extended to Automatic Bayesian Covariance Discovery (ABCD) in order to account for change points. Change points are modeled by multiplying a kernel expression by a sigmoidal function, which results in transitioning between parts of the entire kernel expression at pre-defined time steps (Duvenaud, 2014; Lloyd et al., 2014). Thus, change points are not determined online. The approach of Liu et al. (2015) requires the definition of certain steady states associated with a specific prediction model. During the online phase, the forecasting system can then automatically switch between them or train a new one if no pre-trained model is declared reliable for a particular data point. This approach also requires *a priori* knowledge to define steady states and train individual models for them, which is possible for some technical processes. The authors later extended this work by using fuzzy c-means clustering to partition data samples for training local GPR models. For inference, these local predictors are included in an ensemble weighted by their probability to obtain the final prediction value (Liu & Gao, 2015). Jin et al. (2015) follow a similar idea, but propose a new procedure for clustering the data samples and use online adaptation of the local predictors previously trained on individual data clusters. Moving-Window GPR is another approach that combines a periodical adaptation using the latest samples within a fixed window with a dual updating and pre-processing strategy. With respect to the dual updating, Moving-Window GPR updates the mean and variance used to normalize the data, as well as a bias term based on past model errors to correct current predictions. Furthermore, the authors use a Savitzky-Golay filter (Savitzky & Golay, 1964) for noise reduction on the

covariates and the target variable (Ni et al., 2012). GP-non-Bayesian clustering (GP-NBC) is designed for real-time CPD and regression, with emphasis on computational efficiency and robustness to changing data distributions. For computational efficiency, GP-NBC separates the tasks of CPD, prediction, and model reidentification. In this algorithm, a GP model predicts the next value before being updated with the current sample and stored in a queue of a certain size. For CPD, a likelihood ratio test is performed, buffering a number of samples equal to the size of the model queue. A new candidate GP model is then trained with this data. If a significant deviation of the likelihood ratio is detected, a change point is declared and a new predictor is initialized. For model reuse, a likelihood ratio test is again performed to determine whether a new predictor is significantly different from previous models (Grande et al., 2017). The INstant TEmporal structure Learning (INTEL) algorithm first trains a template GPR model based on data collected offline. Using this template, an ensemble is created by varying hyperparameters based on assumptions about changes that can occur during live operation. For inference, the individual predictions are combined based on the likelihood of a new observation given each model. A drawback of INTEL is that potentially occurring changes must be assumed *a priori* (Liu et al., 2020).

1.3.3 Time Series Forecasting Frameworks

Forecasting competitions and comparative studies have not yielded an overall predominant method, thus requiring the re-evaluation of different prediction models for each time series forecasting task (Bojer & Meldgaard, 2021; Makridakis & Hibon, 2000; Makridakis et al., 2018, 2022). To ensure reproducibility and comparability of results, a user-friendly and automated framework that covers the entire time series forecasting pipeline from data pre-processing over feature engineering to hyperparameter optimization, model selection, and results analysis would be beneficial. Furthermore, a framework should allow for straightforward extension, besides already integrating state-of-the-art prediction methods, to support the development and benchmarking of new forecasting approaches. Meisenbacher et al. (2022) conclude in a recent review paper that no such framework currently exists, most of the available ones covering only parts of the time series forecasting pipeline. In this thesis, we focus on Python as one of the most widely-used programming languages, especially in data science. Several Python libraries are available for ML in general. Frameworks such as `statsmodels` (Seabold & Perktold, 2010), `scikit-learn` (Pedregosa et al., 2011), `PyTorch` (Paszke et al., 2019), and `TensorFlow` (Abadi et al., 2015) are intended for specific types of prediction models, for instance neural networks with respect to the latter two. However, these libraries are not specifically designed for time series forecasting, but

provide important functionality for parts of the pipeline. Beyond that, there are several AutoML frameworks that fully automate the entire ML pipeline, i.a., `auto-sklearn` (Feurer et al., 2015), and `AutoKeras` (Jin et al., 2023). Furthermore, `AutoGluon` (Erickson et al., 2020) and `HyperTS` (Zhang et al., 2022) also include time series prediction capabilities. However, AutoML libraries usually provide a rather high-level interface, making them a useful tool for end users without providing insights into what is actually happening in the background, and often offering restricted customization options. Thus, such frameworks may be less interesting for model developers due to limited extensibility and for research in general, where it is essential to understand the background procedures (Alsharif et al., 2022). There are several publications that present Python libraries designed for specific parts of the time series forecasting pipeline, such as (Bauer et al., 2020), (Züfle & Kounev, 2020), and (Martínez et al., 2019), for which Meisenbacher et al. (2022) give an overview. `Darts` (Herzen et al., 2022) and `sktime` (Löning et al., 2019) are two powerful frameworks providing various time series prediction models. However, both are limited with respect to hyperparameter optimization. For example, they do not integrate state-of-the-art Bayesian optimization, see Section 2.4. This observation also applies to `Merlion` (Bhatnagar et al., 2021), which additionally provides a graphical user interface. `pyWATTS` (Heidrich et al., 2021) has recently been released with the goal of covering the entire time series forecasting pipeline. Beyond that, `PyCaret` (Moez, 2023) integrates time series forecasting capabilities since one of its recent releases, but without including a neural network specific library, for instance, `PyTorch`. These libraries have in common that a user needs to write several lines of code to perform a comparative study, which may require expert knowledge. Moreover, they do not provide a command line interface, which could be helpful for a straightforward setup and deployment on multiple machines in combination with containerization, e.g., via `Docker` (Merkel, 2014). In summary, there are many libraries available for time series forecasting, with a large community maintaining them and developing new functionality. However, there is currently a lack of a comprehensive framework that allows for straightforward, customizable, and reproducible application on multiple machines, including various already integrated prediction models of different types, providing state-of-the-art fully-automated hyperparameter optimization, and allowing for rapid integration and benchmarking of novel approaches.

1.4 Objectives of this Thesis

Focusing on changing data distributions, this thesis intends to address some of the challenges described in Section 1.2, which leads to the objectives outlined subsequently.

Objective 1: Assess the potential and limitations of time series forecasting approaches for predicting sales for small and medium-sized companies dealing with perishable goods in the presence of changing data distributions. As mentioned above, small and medium-sized companies dealing with perishable products could gain a competitive advantage by leveraging accurate demand forecasting. However, changes in the data distribution resulting in inaccurate predictions are a risk, especially for less financially strong companies. In this thesis, we aim to (i) evaluate the potential and limitations of time series forecasting approaches in predicting demand for perishable products of such companies, and (ii) assess potential limitations due to changing data distributions using real-world data. As an example, we select the horticultural sector, a domain with annual sales in the billions in Germany alone, which is affected by the disposal of unsold items but has received only limited attention in forecasting research. Furthermore, horticultural sales show scientifically interesting characteristics such as strong seasonality and many influencing external factors. These factors, as well as operational and strategic decisions, can further lead to a shift in the data distribution. For a first proof of concept, we use retail sales data, the level of trade that is closest to the end customer and thus most likely to be influenced by external factors such as weather and holidays. We assess both classical time series forecasting as well as frequentist and probabilistic ML-based approaches, which are outlined in Chapter 2, for this prediction task. Based on the results of this study, we can decide whether the time series forecasting methods used have the potential to predict horticultural sales accurately enough. In addition, this study allows for the assessment of limitations due to distributional shifts that may be present in the data, i.e., whether the employed prediction models are able to handle such changes in a real-world setting. Our findings can then be transferred to a broader comparative study using data from companies across the entire value chain, and provide guidance for the development of approaches addressing changes in the data distribution.

Objective 2: Develop novel computationally efficient methods that can quickly adapt to changing data distributions during live operation of a forecasting system. Changing data distributions are a challenge for time series forecasting systems, especially during live operation. The use of outdated prediction models trained on a previous data

distribution can cause damage, e.g., financial loss. Therefore, methods that quickly adapt to such changes are essential. We aim to develop novel online approaches for this purpose, focusing on computational efficiency and fast adaptation to a changed data distribution to allow for integration into a real-world forecasting system. The primary objective is to provide up-to-date predictions at all times while considering a potential trade-off with computational efficiency. In order to assess the applicability and limitations, we evaluate our newly developed approaches on simulated data with pre-defined characteristics and on real-world data from different domains. Regarding the prediction model, we focus on GPR, see Section 2.3.2, a probabilistic approach with good performance in several studies and providing prediction uncertainties that are potentially useful for subsequent applications. Based on the assumptions for the design of an algorithm addressing this challenge, Objective 2 can be further divided into two subgoals:

Objective 2A: Develop a novel computationally efficient online method that can quickly adapt to a change in the output scale of seasonal data. Seasonality, i.e., periodic system behavior, is a pattern that many time series show. This property can be advantageous for detecting a change in the data distribution, since we can compare current data points to previous seasons. Furthermore, a distributional shift may be visible in the scale of the target variable, leading to higher or lower values, but not change the system behavior entirely. We aim to exploit these two properties to develop a novel and computationally efficient online method that allows for rapid adaptation of a prediction model while focusing on changes in the output scale of seasonal data. These assumptions can be beneficial for the algorithm development and are not a major limitation due to their commonness.

Objective 2B: Develop a novel and computationally efficient method that can quickly adapt online to various types of distributional shifts. Nevertheless, other types of distributional shifts may require different detection and adaptation mechanisms, e.g., a new kernel search in case of GPR. Therefore, we aim to develop a novel approach that accounts for different types of distributional shifts and adapts a prediction model accordingly during the online phase. Besides not being limited to output scale changes, this method should also be able to handle non-seasonal time series. Thus, by making fewer assumptions, we aim to achieve a broader applicability, which may be at the cost of computational efficiency as well as speed and accurateness of model adaptation compared to the more specific scenario of Objective 2A.

Objective 3: Enable reproducibility and comparability of results and facilitate accessibility of novel approaches through a fully-automated, user-friendly and easy-to-extend time series forecasting framework. Several forecasting competitions and comparative studies have shown the need to re-evaluate different prediction models for each new time series forecasting task. However, as outlined in Section 1.3.3, there is currently no comprehensive, easy-to-use, and easy-to-extend framework available to perform such studies. A key issue in this context is to ensure reproducible and comparable results. Hence, we aim to address this challenge by developing a novel time series forecasting framework that covers the entire workflow and employs state-of-the-art approaches. For ease of use, the pipeline should be fully automated but allow for customization. Furthermore, it should pre-integrate various classical forecasting and ML-based prediction models, while allowing for rapid integration and benchmarking of novel approaches. Hence, we also want to facilitate the accessibility of newly developed methods, for instance, ours with respect to Objective 2. In this way, we aim to provide a powerful resource for both end users and model developers. The framework should be available with a command line interface and as a programming library, both of which require only a single line of code to run a comparative study. In addition, users need to be supported with comprehensive and hands-on online documentation.

1.5 Contributions

The aforementioned objectives of this thesis are addressed in four publications, for which detailed summaries and author contributions are given in Chapter 3. During my PhD, I further contributed to the four research papers that are outlined in a second subsection below. Since these are not main contributions of this thesis, the topics and author contributions are briefly described without detailed summaries.

1.5.1 Main Contributions

One of the objectives of this thesis is to assess the applicability of time series forecasting methods to predict the sales of small and medium-sized horticultural companies in the presence of distributional shifts. For this purpose, we use horticultural retail sales data in a first-time comparison of classical time series forecasting and ML-based approaches to predict sales for products such as potted plants, cut flowers, and shrubs. Our comparative study further allows to evaluate prediction results with and without changes in the data distribution based on a real-world example. This research resulted in the following publication, which

we describe in more detail in Section 3.1:

PUBLICATION A:

Florian Haselbeck, Jennifer Killinger, Klaus Menrad, Thomas Hannus, and Dominik G. Grimm (2022). Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions. *Machine Learning with Applications*, 7, 100239. <https://doi.org/10.1016/j.mlwa.2021.100239>

With respect to the general problem of changing data distributions, we observe that all prediction models are limited in handling this challenge despite periodical model refittings. We therefore address this issue in two further research papers. In a first publication, described in detail in Section 3.2, we focus on changes in the output scale and seasonal data, see Objective 2A. In the resulting paper given below, we apply CPD methods to trigger an augmented refitting of a GPR model. By augmenting existing data, we allow for a quick adjustment of the current prediction model.

PUBLICATION B:

Florian Haselbeck and Dominik G. Grimm (2021). EVARS-GPR: EVent-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data. *KI 2021: Advances in Artificial Intelligence*, 12873, 135–157. https://doi.org/10.1007/978-3-030-87626-5_11

To ensure an up-to-date model delivering useful predictions at all times for other data distribution changes as well, e.g., a shift of the periodicity, we extended our work addressing Objective 2B. This goal requires searching for a new kernel expression of the underlying GPR model, for which we use the efficient Adjusting Kernel Search (AKS), see Section 2.3.2. To trigger this model adjustment, we also employ an online CPD method. For more details on the following publication, we refer to Section 3.3.

PUBLICATION C:

Jan D. Hüwel*, **Florian Haselbeck***, Dominik G. Grimm and Christian Beecks (2022). Dynamically Self-adjusting Gaussian Processes for Data Stream Modelling. *KI 2022: Advances in Artificial Intelligence*, 13404, 96–114. https://doi.org/10.1007/978-3-031-15791-2_10

*Both authors contributed equally and share first authorship.

A further problem in this field is the need for a comprehensive, easily extensible, and user-friendly time series forecasting framework. This is required to ensure reproducible and comparable results as well as to facilitate accessibility of novel methods, such as the ones

presented in the two publications given above. Therefore, we have developed ForeTiS, an easy-to-use Python framework that fully automates the entire time series forecasting pipeline. Our framework applies state-of-the-art methods, e.g., Bayesian optimization for hyperparameter tuning, and allows for rapid integration and benchmarking of new forecasting approaches. ForeTiS is published in the following research paper, see Section 3.4 for more details.

PUBLICATION D:

Josef Eiglsperger*, **Florian Haselbeck*** and Dominik G. Grimm (2023). ForeTiS: A comprehensive time series forecasting framework in Python. *Machine Learning with Applications*, 12, 100467. <https://doi.org/10.1016/j.mlwa.2023.100467>

*Both authors contributed equally and share first authorship.

1.5.2 Further Contributions

The four above-described main contributions of this thesis are related to three further research papers in which I have been involved. The relationships to the main publications, the topics addressed, and the individual author contributions are briefly outlined below, without detailed summaries of the content.

The time series forecasting framework ForeTiS presented in PUBLICATION D is based on the design and implementation of easyPheno, which has been developed for phenotype prediction, a field of research in bioinformatics. Despite this different application domain, the design principles, e.g., to simplify the integration of new prediction models and several parts of the code, i.a., the fully-automated hyperparameter optimization, have been transferable.

Florian Haselbeck*, Maura John* and Dominik G. Grimm (2023). easyPheno: An easy-to-use and easy-to-extend Python framework for phenotype prediction using Bayesian optimization. *Bioinformatics Advances*, 3. <https://doi.org/10.1093/bioadv/vbad035>

*Both authors contributed equally and share first authorship.

Florian Haselbeck, Maura John, and Dominik G. Grimm designed the software package. Maura John implemented the data pre-processing and the synthetic data generation, with contributions from Dominik G. Grimm. Florian Haselbeck wrote the code for the fully-automated hyperparameter optimization, the prediction models, including their easy-to-extend structure, the results analysis, and the surrounding parts for the pipeline automation.

Florian Haselbeck and Maura John created the online documentation as well as the initial draft of the manuscript, which was reviewed and edited by Dominik G. Grimm.

We employed `easyPheno` to conduct a comparative study of classical and ML-based phenotype prediction models, both on simulated and real-world data. This research resulted in the following publication:

Maura John*, **Florian Haselbeck***, Rupashree Dass, Christoph Malisi, Patrizia Ricca, Christian Dreischer, Sebastian J. Schultheiss and Dominik G. Grimm (2022). A comparison of classical and machine learning-based phenotype prediction methods on simulated data and three plant species. *Frontiers in Plant Science*, 13. <https://doi.org/10.3389/fpls.2022.932512>

**Both authors contributed equally and share first authorship.*

Maura John, Florian Haselbeck, and Dominik G. Grimm designed the study. Rupashree Dass, Christoph Malisi, Patrizia Ricca, Christian Dreischer, and Sebastian J. Schultheiss provided and prepared two real-world breeding datasets. Maura John prepared the data of the model organism *Arabidopsis thaliana* and generated the synthetic datasets. Florian Haselbeck and Maura John performed all experiments and comparative as well as statistical analyses. Maura John, Florian Haselbeck, and Dominik G. Grimm analyzed and interpreted the results. Maura John, Florian Haselbeck, and Dominik G. Grimm wrote the manuscript, with contributions from the further authors.

Beyond that, the code base of `ForeTiS` and `easyPheno` was further transferred to another bioinformatics problem, namely the prediction of protein thermophilicity. This enabled us to quickly develop and benchmark new approaches for this domain, including a protein language model-based classifier, which outperforms the current state-of-the-art.

Florian Haselbeck, Maura John, Yuqi Zhang, Jonathan Pirnay, Juan Pablo Fuenzalida-Werner, Rubén D. Costa and Dominik G. Grimm (2023). Superior Protein Thermophilicity Prediction With Protein Language Model Embeddings.

currently under review

Florian Haselbeck, Maura John, and Dominik G. Grimm designed the study and the architecture of the novel prediction model. Maura John and Yuqi Zhang collected, reviewed, and prepared the data. Florian Haselbeck implemented the analysis pipeline, with contributions from Jonathan Pirnay for the transformer-based models. Florian Haselbeck and Maura John conducted all experiments. Florian Haselbeck, Maura John, and Dominik G. Grimm analyzed and interpreted the results. Florian Haselbeck, Maura John, and Dominik Grimm

created the manuscript, with support from Jonathan Pirnay, Yuqi Zhang, Juan Pablo Fuenzalida-Werner, and Rubén D. Costa.

I further contributed to the following publication, which is topic-wise not related to the main contributions of this thesis, by running parts of the experiments, supporting with results analysis, as well as reviewing and editing the manuscript. In this work, we leverage a deep learning (DL)-based approach to identify weeds in sorghum fields using images collected by a drone.

Nikita Genze, Raymond Ajekwe, Zeynep Güreli, **Florian Haselbeck**, Michael Grieb, Dominik G. Grimm (2022). Deep learning-based early weed segmentation using motion blurred UAV images of sorghum fields. *Computers and Electronics in Agriculture*, 202(C), 107388. <https://doi.org/10.1016/j.compag.2022.107388>

Material and Methods

In the following, we present the material and methods relevant to this thesis. First, we outline classical time series forecasting methods, before explaining frequentist and probabilistic machine learning (ML) approaches that can be applied for time series prediction. In Section 2.3, we focus on Gaussian Process Regression (GPR), which is a key component for two of the main contributions of this thesis. Section 2.4 describes model selection and hyperparameter optimization techniques. Finally, we introduce change point detection (CPD) methods that are important for this work.

2.1 Classical Time Series Forecasting Methods

Recalling the problem formulation that is given in the introduction, we consider a dataset $\mathcal{D} = \{(\mathbf{x}_t, y_t) \mid t = 1, \dots, n\}$ containing $n \in \mathbb{N}$ pairs of a feature vector $\mathbf{x}_t \in \mathbb{R}^m$ consisting of $m \in \mathbb{N}$ feature values and a target value $y_t \in \mathbb{R}$ at time step $t \in \mathbb{N}$. Classical forecasting methods aim to predict future sequences by using chronologically ordered time series data, of which they usually retrieve statistical information for projection. While univariate approaches use only the time series of the target variable $\mathbf{y} \in \mathbb{R}^n$ containing n target values y_t observed at successive time steps t , multivariate methods additionally consider the m external features collected in $\mathbf{X} \in \mathbb{R}^{n \times m}$ for the whole dataset with n samples. Two common univariate classical forecasting models are Exponential Smoothing and Autoregressive Integrated Moving Average (ARIMA). In its basic form, Exponential Smoothing predicts a weighted average of past time series values with exponentially decaying weights to place more emphasis on more recent observations (Brown, 1956). This can be extended with a trend and seasonal component to additionally capture such

properties, which can be modeled in an additive or multiplicative way (Holt, 1957; Winters, 1960). Using the additive formulation, the predicted value $\hat{y}_{t+h} \in \mathbb{R}$ at time step $t+h$ with the forecast horizon $h \in \mathbb{N}$ can be determined as

$$\hat{y}_{t+h} = l_t + (\rho + \rho^2 + \dots + \rho^h)v_t + c_{t-n_{seas} + \lfloor (h-1) \bmod n_{seas} \rfloor + 1} \quad (2.1)$$

with the floor function $\lfloor x \rfloor$ giving the largest integer value less than or equal to x , the seasonal length $n_{seas} \in \mathbb{N}$, and the damping parameter $\rho \in]0, 1]$ to model a damped trend, for which $\rho = 1$ is equal to a linear trend. The level $l_t \in \mathbb{R}$, the trend component $v_t \in \mathbb{R}$, and the seasonal part $c_t \in \mathbb{R}$ at time step t are further defined as

$$\begin{aligned} l_t &= \alpha(y_t - c_{t-n_{seas}}) + (1 - \alpha)(l_{t-1} + \rho v_{t-1}) \\ v_t &= \beta(l_t - l_{t-1}) + (1 - \beta)\rho v_{t-1} \\ c_t &= \gamma(y_t - l_{t-1} - v_{t-1}) + (1 - \gamma)c_{t-n_{seas}} \end{aligned} \quad (2.2)$$

with $\alpha \in [0, 1]$, $\beta \in [0, 1]$, and $\gamma \in [0, 1]$ being the smoothing parameters of the level, trend and seasonal component, respectively (Gardner, 2006; Hyndman & Athanasopoulos, 2021).

In contrast to Exponential Smoothing, which attempts to approximate the individual components of a time series, the key idea of ARIMA is to model autocorrelations, i.e., the correlation between a series and a lagged version of itself. An ARIMA model consists of autoregressive (AR) and moving average (MA) terms. The AR part determines predictions using a linear combination of past observations of the target variable, similar to a multiple regression model but employing lagged target values instead of independent features. The MA component uses a linear combination of past error terms for forecasting. An assumption of these models is a stationary time series. Therefore, the so-called integrated component was introduced because differencing, i.e., calculating the difference between consecutive observations, can help stabilize the mean (Box & Jenkins, 1970). Using the d times differenced time series $\mathbf{y}^d \in \mathbb{R}^n$ leads to the formulation of an $\text{ARIMA}(p_{AR}, d, q_{MA})$ model with the differencing degree $d \in \mathbb{N}$, the intercept $\mu \in \mathbb{R}$, the white noise error term $\varepsilon_{wn,t} \in \mathbb{R}$, the order of the AR and MA part $p_{AR} \in \mathbb{N}$ and $q_{MA} \in \mathbb{N}$, and the model parameters $\lambda_i \in \mathbb{R}$ and $\chi_i \in \mathbb{R}$:

$$\mathbf{y}_t^d = \mu + \underbrace{\sum_{i=1}^{p_{AR}} \lambda_i \mathbf{y}_{t-i}^d}_{\text{AR}(p_{AR})} + \underbrace{\sum_{i=1}^{q_{MA}} \chi_i \varepsilon_{wn,t-i}}_{\text{MA}(q_{MA})} + \varepsilon_{wn,t} \quad (2.3)$$

A widely-used extension of ARIMA accounting for seasonal behavior is Seasonal Autoregressive Integrated Moving Average (SARIMA). SARIMA additionally introduces seasonal AR and MA components and a seasonal differencing step for stationarity. The formulation of these parts is similar to the non-seasonal ones but includes a backshift of a seasonal cycle n_{seas} . The parameters of the seasonal components are typically given in uppercase notation: $P_{AR} \in \mathbb{N}$, $D \in \mathbb{N}$, and $Q_{MA} \in \mathbb{N}$ (Hyndman & Athanasopoulos, 2021; Shumway & Stoffer, 2000). To include external factors \mathbf{X} , SARIMA has been further extended to its multivariate version Seasonal Autoregressive Integrated Moving Average with eXogenous factors (SARIMAX). Typically, this is achieved by an additional linear regression model using the external factors, see Section 2.2, allowing the usually uncorrelated error term to be autocorrelated. This additive error term is modeled using SARIMA, leading to the full SARIMAX model (Hyndman & Athanasopoulos, 2021; Shumway & Stoffer, 2000).

2.2 Frequentist Machine Learning for Time Series Forecasting

The problem of predicting a continuous target value y_{t+h} of a time series based on features can be considered as a regression task with respect to ML. From a frequentist viewpoint, probabilities are seen as long-run frequencies of random and repeatable events. Regarding the likelihood function $p(\mathcal{D} | \mathbf{w})$, we consider a model's parameters \mathbf{w} to be fixed. Hence, in frequentist ML, we try to determine point estimates \mathbf{w}^* of the parameters that best explain the observed data \mathcal{D} (Bishop, 2006; Murphy, 2022). In the following, we outline frequentist ML-based methods for time series forecasting that are relevant to this thesis, namely regularized linear regression, the ensemble learner XGBoost, and neural network-based approaches including a Multilayer Perceptron (MLP) and a Long Short-Term Memory (LSTM) network.

A common approach in this context is linear regression, which models the relationship between the target values \mathbf{y} and the feature matrix \mathbf{X} as

$$\mathbf{y} = \mathbf{X}^\dagger \mathbf{w} + \boldsymbol{\varepsilon}, \quad (2.4)$$

with the weights $\mathbf{w} = (w_0, w_1, \dots, w_m)^T \in \mathbb{R}^{m+1}$ including the intercept $w_0 \in \mathbb{R}$ and the model coefficients $w_k \in \mathbb{R}$, $\mathbf{X}^\dagger \in \mathbb{R}^{n \times m+1}$ containing a vector of ones and the feature matrix \mathbf{X} , and the unobserved random error $\boldsymbol{\varepsilon} \in \mathbb{R}^n$. For regularization, a penalty term $\Omega(\mathbf{w}) \in \mathbb{R}$ is usually included when fitting a linear regression model. With $\Omega(\mathbf{w})$ weighted

by $\kappa \in \mathbb{R}_{>0}$, we can estimate the weights \boldsymbol{w} by minimizing the deviation between the target values \boldsymbol{y} and the predicted values $\mathbf{X}^\dagger \boldsymbol{w}$ (James et al., 2017; Yan & Su, 2009):

$$\operatorname{argmin}_{\boldsymbol{w}} \frac{1}{2} \|\boldsymbol{y} - \mathbf{X}^\dagger \boldsymbol{w}\|_2^2 + \kappa \Omega(\boldsymbol{w}) \quad (2.5)$$

Depending on the regularization term $\Omega(\boldsymbol{w})$, we can distinguish between Least Absolute Shrinkage and Selection Operator (LASSO), Ridge, and Elastic Net Regression. LASSO uses the L1-norm $\|\boldsymbol{w}\|_1$, i.e., the sum of the absolute values of the weights, for penalization. This sparsity constraint forces the weights of unimportant features toward zero, which is why this is often considered an automatic feature selection (Tibshirani, 1996). Ridge Regression instead employs the L2-norm $\|\boldsymbol{w}\|_2^2$, for which squared instead of absolute values of the weights are used. This regularizer is known to distribute the influence among correlated features (Hoerl & Kennard, 1970). For Elastic Net, we use $\Omega(\boldsymbol{w}) = \zeta \|\boldsymbol{w}\|_2^2 + (1 - \zeta) \|\boldsymbol{w}\|_1$, i.e., a weighted sum of the L1-norm and the L2-norm controlled by the hyperparameter $\zeta \in [0, 1]$, combining both above-mentioned effects (Zou & Hastie, 2005).

Gradient boosted trees have performed well in several forecasting competitions, which makes these approaches highly relevant to this thesis (Bojer & Meldgaard, 2021). XGBoost is a scalable and runtime-efficient implementation of gradient boosting. With this technique, weak learners, often decision trees, are added sequentially with a focus on the errors of the current ensemble. Thereby, the model's bias with respect to the bias-variance trade-off gets reduced. This process is guided by a gradient descent algorithm over a loss function. Besides computationally efficient procedures and the ability to work with missing values, XGBoost adds measures to prevent overfitting. XGBoost uses a regularized objective and so-called shrinkage, which scales newly added weights after each step of tree boosting. Furthermore, XGBoost employs feature and row subsampling, i.e., randomly selecting features and samples to construct an individual weak learner instead of using all of them (Chen & Guestrin, 2016; James et al., 2017).

Beyond that, neural network-based approaches can be applied for time series forecasting. Using MLPs, we can model complex and nonlinear relationships. The output vector $\boldsymbol{o}_i \in \mathbb{R}^{n_{i,o}}$ of each layer $i \in \mathbb{N}$ with the number of neurons $n_{i,o} \in \mathbb{N}$ can be defined as

$$\boldsymbol{o}_i = \boldsymbol{a}(\mathbf{W}_i^T \boldsymbol{x}_{i,in} + \boldsymbol{b}_i), \quad (2.6)$$

with the activation function $\boldsymbol{a}: \mathbb{R}^{n_{i,o}} \rightarrow \mathbb{R}^{n_{i,o}}$, the weight matrix of the i -th network layer $\mathbf{W}_i \in \mathbb{R}^{n_{i,in} \times n_{i,o}}$, the input vector of the i -th layer $\boldsymbol{x}_{i,in} \in \mathbb{R}^{n_{i,in}}$ with the number of inputs

$n_{i,in} \in \mathbb{N}$, and the bias $\mathbf{b}_i \in \mathbb{R}^{n_{i,\rho}}$. In an MLP, multiple layers are combined, leading to a chain of functions that form the neural network (Goodfellow et al., 2016). Beyond that, recurrent neural networks accounting for temporal dependencies are an obvious option when working with time series data. In a recent review, Hewamalage et al. (2021) conclude that in particular LSTM networks show a competitive performance. LSTM cells can capture long-term dependencies while preventing vanishing gradients through memory and gating mechanisms. A key component of an LSTM cell for storing long-term dependencies is the cell state. In addition, there are three gates that manage the flow of information. The input gate controls the influence of the input on the current state of the cell, while the forget gate determines how much of the current and previous information is retained. Finally, the output gate filters the updated cell state to control the impact on the output of the LSTM cell (Hochreiter & Schmidhuber, 1997). Neural networks are typically trained employing empirical risk minimization, i.e., minimizing the average loss function across the used data samples. This optimization is usually done in multiple epochs, with the whole training data being processed once per epoch, using gradient descent algorithms on so-called mini-batches of the data. By doing so, a subset of the data is passed through the network, the gradient of the loss function with respect to the model's parameters is calculated using backpropagation, and the weights are adjusted accordingly. The size of the mini-batch is a hyperparameter, with smaller batch sizes leading to more frequent model updates but potentially less accurate error estimates (Goodfellow et al., 2016). A typical optimizer for training neural networks is Adam (Kingma & Ba, 2014). Both MLPs and LSTMs are powerful approximators, and thus regularization needs to be employed to prevent overfitting. A common technique in this context is dropout, randomly ignoring a certain ratio of neurons during each training iteration (Srivastava et al., 2014). Furthermore, early stopping can be used, i.e., monitoring the loss on validation data during training and interrupting the process if no improvement is observed for a certain number of epochs (Bishop, 2006).

2.3 Probabilistic Machine Learning for Time Series Forecasting

In probabilistic ML, the parameters \boldsymbol{w} with respect to the likelihood function $p(\mathcal{D} | \boldsymbol{w})$ are not considered to be fixed in contrast to frequentist ML. We assume that the parameters \boldsymbol{w} contain uncertainty, which is achieved by defining a probability distribution over \boldsymbol{w} (Murphy, 2022). From a Bayesian perspective, we define a *prior* belief over our model

parameters $p(\boldsymbol{w})$ before considering evidence, e.g., the observed data points \mathcal{D} . With this evidence, we can define a *posterior* probability $p(\boldsymbol{w} | \mathcal{D})$ using Bayes' theorem

$$p(\boldsymbol{w} | \mathcal{D}) = \frac{p(\mathcal{D} | \boldsymbol{w})p(\boldsymbol{w})}{p(\mathcal{D})}. \quad (2.7)$$

From a practical perspective, probabilistic models have the advantage of providing uncertainties of predicted values, which can be helpful for downstream tasks. Subsequently, we first outline Bayesian regression and Bayesian neural network approaches. Then, we describe GPR, one of the most relevant prediction models in the context of this thesis.

2.3.1 Bayesian Regression and Bayesian Neural Networks

Linear regression can also be formulated in a Bayesian viewpoint, where y_t is assumed to follow a normal distribution

$$p(y_t | \boldsymbol{x}_t, \boldsymbol{w}) = \mathcal{N}(y_t | w_0 + \boldsymbol{w}^T \boldsymbol{x}_t, \sigma^2) \quad (2.8)$$

with mean $w_0 + \boldsymbol{w}^T \boldsymbol{x}_t$, including the bias term $w_0 \in \mathbb{R}$ and model coefficients $\boldsymbol{w} \in \mathbb{R}^m$, and variance $\sigma^2 \in \mathbb{R}_{>0}$. Typically, regularization is applied to prevent overfitting, leading to a so-called maximum a posteriori (MAP) estimation. Putting a zero-mean Gaussian *prior* on the weights $p(\boldsymbol{w} | \nu) = \mathcal{N}(\boldsymbol{w} | 0, \nu^{-1} \boldsymbol{I})$ with precision $\nu \in \mathbb{R}_{>0}$ and the identity matrix $\boldsymbol{I} \in \mathbb{R}^{m \times m}$ leads to Bayesian Ridge Regression, which can be shown to be equivalent to the frequentist approach (Bishop, 2006; Murphy, 2022). Automatic Relevance Determination (ARD) is similar to Bayesian Ridge Regression but uses a different *prior* for the weights \boldsymbol{w} . Each coefficient is drawn from a Gaussian distribution with zero mean and precision $\nu_i \in \mathbb{R}_{>0}$. This leads to $p(\boldsymbol{w} | \boldsymbol{Y}) = \mathcal{N}(\boldsymbol{w} | 0, \boldsymbol{Y}^{-1})$, with the precision matrix $\boldsymbol{Y} \in \mathbb{R}^{m \times m}$ only containing positive values and $\text{diag}(\boldsymbol{Y}) = \{\nu_1, \dots, \nu_m\}$. Finally, pruning features with low variance, i.e., with weights likely to be close to zero, leads to sparser solutions (Bishop, 2006; Tipping, 2001).

Furthermore, Bayesian neural networks, namely Bayesian versions of an MLP and an LSTM network outlined in Section 2.2, are relevant to this thesis. Instead of point estimates of the neural network parameters, i.e., the weights \boldsymbol{W} and biases \boldsymbol{b} , Bayesian neural networks learn probability distributions for them. Thus, a Bayesian neural network can be thought of as an ensemble of networks, each parameterized by weights drawn from a shared probability distribution. By considering the model's parameters this way, we also retrieve uncertainties of the output and the underlying process, respectively, since the uncertainty

is low if the different models agree. Furthermore, by accounting for the uncertainty of the underlying process, we reduce the risk of overfitting. Since in this context exact inference is generally intractable, we have to use approximations. A common approach is variational inference, which reformulates finding the intractable probability distribution p as an optimization problem over a class of tractable distributions to determine the one that most closely approximates the true distribution (Jospin et al., 2022; Murphy, 2023). Bayes by Backprop is a variational inference method that employs a reparametrization trick to enable using regular backpropagation. As a result, a quick transformation of a frequentist implementation is possible (Blundell et al., 2015).

2.3.2 Gaussian Process Regression

GPR is a non-parametric Bayesian ML-based model that can be employed for regression tasks. Non-parametric means that no explicit assumptions are made about the model's functional form, but it automatically adapts to the data. Besides the advantage of inherently providing prediction uncertainties, GPR is a flexible method that works well for small datasets but also scales to large amounts of data when using methodical extensions. Two of the main contributions of this thesis outlined in Sections 3.2 and 3.3 are based on GPR, which is why it is subsequently introduced in detail.

Mathematical Description

The following mathematical description is based on (Rasmussen & Williams, 2006), (Bishop, 2006), and (Williams & Rasmussen, 1995). To understand the details, it makes sense to recall the linear regression model. Let $y_t \in \mathbb{R}$ denote the target value and $\hat{y}_t \in \mathbb{R}$ the prediction based on the input vector $\mathbf{x}_t \in \mathbb{R}^m$ and the model parameters $\mathbf{w} \in \mathbb{R}^m$:

$$\hat{y}_t = \mathbf{x}_t^T \mathbf{w}, \quad y_t = \hat{y}_t + \varepsilon_t \quad (2.9)$$

We assume that the true and the predicted value differ by zero-mean Gaussian noise $\varepsilon_t \sim \mathcal{N}(0, \sigma_n^2)$ with variance σ_n^2 , neglecting a bias term. Further assuming independence of the observations in the training data, we can factorize over all n samples to obtain the likelihood term

$$p(\mathbf{y} | \mathbf{X}, \mathbf{w}) = \prod_{t=1}^n p(y_t | \mathbf{x}_t, \mathbf{w}) = \mathcal{N}(\mathbf{X}^T \mathbf{w}, \sigma_n^2 \mathbf{I}) \quad (2.10)$$

with the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$. Specifying a zero-mean Gaussian *prior* over the model parameters $\mathbf{w} \sim \mathcal{N}(0, \mathbf{C})$ with covariance matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ leads to the *posterior*

distribution. Using Bayes' theorem, we can connect the likelihood term given in Equation 2.10 with the *prior* $p(\boldsymbol{w})$:

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \mathbf{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{w}, \mathbf{X})p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \mathbf{X})}. \quad (2.11)$$

Note that the normalizing constant in the denominator is independent of the model parameters, allowing to continue with the likelihood and *prior* given in the numerator. Finally, this leads to the form of the *posterior* distribution, which is also a Gaussian

$$p(\boldsymbol{w} \mid \boldsymbol{y}, \mathbf{X}) \sim \mathcal{N}(\bar{\boldsymbol{w}}, \mathbf{A}^{-1}) \quad (2.12)$$

with the mean $\bar{\boldsymbol{w}} \in \mathbb{R}^m$ being $\frac{1}{\sigma_n^2} \mathbf{A}^{-1} \mathbf{X} \boldsymbol{y}$ and the covariance matrix $\mathbf{A}^{-1} \in \mathbb{R}^{m \times m}$, where $\mathbf{A} = \sigma_n^{-2} \mathbf{X} \mathbf{X}^T + \mathbf{C}^{-1}$. To get a prediction for an input $\boldsymbol{x}_{test} \in \mathbb{R}^m$, we marginalize out $\boldsymbol{w} \in \mathbb{R}^m$ by averaging over all possible parameter values weighted by the corresponding *posterior* probability, leading again to a Gaussian distribution:

$$\begin{aligned} p(y_{test} \mid \boldsymbol{x}_{test}, \mathbf{X}, \boldsymbol{y}) &= \int p(y_{test} \mid \boldsymbol{x}_{test}, \boldsymbol{w}) p(\boldsymbol{w} \mid \mathbf{X}, \boldsymbol{y}) d\boldsymbol{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma_n^2} \boldsymbol{x}_{test}^T \mathbf{A}^{-1} \mathbf{X} \boldsymbol{y}, \boldsymbol{x}_{test}^T \mathbf{A}^{-1} \boldsymbol{x}_{test}\right) \end{aligned} \quad (2.13)$$

The mean of this distribution is equal to the mean of the *posterior* multiplied by the test input \boldsymbol{x}_{test} , and the variance only depends on the test input and the *posterior* variance.

So far, this is still equal to Bayesian linear regression constrained to linearity, which we overcome by transforming the inputs to a high-dimensional space and applying the linear solution there. For that purpose, we define the function $\boldsymbol{\phi}: \mathbb{R}^m \rightarrow \mathbb{R}^f$ mapping an m -dimensional vector into an f -dimensional feature space with $f \in \mathbb{N}$. Inserting this to the linear model given in Equation 2.9 leads to

$$\boldsymbol{y}_t = \boldsymbol{\phi}(\boldsymbol{x}_t)^T \boldsymbol{w} + \varepsilon_t \quad (2.14)$$

with the model parameters $\boldsymbol{w} \in \mathbb{R}^f$. Using this equation, we can perform the same derivations as we did for the linear model. Substituting $\mathbf{X} \in \mathbb{R}^{n \times m}$ with transformed and transposed $\boldsymbol{\Phi} \in \mathbb{R}^{f \times n}$ and $\boldsymbol{x}_{test} \in \mathbb{R}^m$ with transformed $\boldsymbol{\phi}(\boldsymbol{x}_{test}) \in \mathbb{R}^f$ finally leads to the predictive distribution again:

$$p(y_{test} \mid \boldsymbol{x}_{test}, \mathbf{X}, \boldsymbol{y}) = \mathcal{N}\left(\frac{1}{\sigma_n^2} \boldsymbol{\phi}(\boldsymbol{x}_{test})^T \mathbf{A}^{-1} \boldsymbol{\Phi} \boldsymbol{y}, \boldsymbol{\phi}(\boldsymbol{x}_{test})^T \mathbf{A}^{-1} \boldsymbol{\phi}(\boldsymbol{x}_{test})\right) \quad (2.15)$$

Note that after this transformation, Φ is part of $A = \sigma_n^{-2}\Phi\Phi^T + C^{-1}$ with the covariance matrix $C \in \mathbb{R}^{f \times f}$ of the model parameters w . To obtain predictions according to the above equation, we need to invert A . Inverting an $f \times f$ matrix A becomes computationally expensive for large dimensional feature spaces. Defining $K = \Phi^T C \Phi \in \mathbb{R}^{n \times n}$ enables to rewrite Equation 2.15 with the identity matrix $I \in \mathbb{R}^{n \times n}$ as follows:

$$\begin{aligned} p(\mathbf{y}_{test} | \mathbf{x}_{test}, \mathbf{X}, \mathbf{y}) &= \\ &= \mathcal{N}(\boldsymbol{\phi}(\mathbf{x}_{test})^T \mathbf{C} \Phi (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}, \\ &\quad \boldsymbol{\phi}(\mathbf{x}_{test})^T \mathbf{C} \boldsymbol{\phi}(\mathbf{x}_{test}) - \boldsymbol{\phi}(\mathbf{x}_{test})^T \mathbf{C} \Phi (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \Phi^T \mathbf{C} \boldsymbol{\phi}(\mathbf{x}_{test})) \end{aligned} \quad (2.16)$$

Since the dimension of K is $n \times n$, we only need to invert $n \times n$ matrices in Equation 2.16, which is computationally less expensive if $n < f$. Hence, the computational complexity of the matrix inversion depends on the number of samples n and not on the dimensionality of the feature space f . In Equation 2.16, elements in the dimensionality of the feature space are part of the terms $\Phi^T C \Phi$, $\boldsymbol{\phi}(\mathbf{x}_{test})^T C \Phi$, and $\boldsymbol{\phi}(\mathbf{x}_{test})^T C \boldsymbol{\phi}(\mathbf{x}_{test})$, respectively. We can further define a so-called kernel k with $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T C \boldsymbol{\phi}(\mathbf{x}')$, where \mathbf{x} and \mathbf{x}' are in the training and test set, respectively. This inner product can be further reformulated as the dot product $k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\psi}(\mathbf{x}) \cdot \boldsymbol{\psi}(\mathbf{x}')$, where $\boldsymbol{\psi}(\mathbf{x}) = C^{1/2} \boldsymbol{\phi}(\mathbf{x})$, decomposing positive definite C into $(C^{1/2})^2$. Replacing the inner products in the input space with a kernel to implicitly get to the feature space is also called the kernel trick. Applying this to Equation 2.16 and defining the $n \times n^*$ covariance matrix $K(\mathbf{X}, \mathbf{X}^*) \in \mathbb{R}^{n \times n^*}$ resulting from all pairwise evaluations of n training samples $\mathbf{X} \in \mathbb{R}^{n \times m}$ and $n^* \in \mathbb{N}$ test samples $\mathbf{X}^* \in \mathbb{R}^{n^* \times m}$ yields the predictive distribution with the target vector of the test samples $\mathbf{y}^* \in \mathbb{R}^{n^*}$:

$$p(\mathbf{y}^* | \mathbf{X}^*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\bar{\mathbf{y}}^*, \mathbf{COV}(\mathbf{y}^*)) \quad (2.17)$$

with

$$\bar{\mathbf{y}}^* = \mathbf{K}(\mathbf{X}^*, \mathbf{X}) [\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}, \quad (2.18)$$

$$\mathbf{COV}(\mathbf{y}^*) = \mathbf{K}(\mathbf{X}^*, \mathbf{X}^*) - \mathbf{K}(\mathbf{X}^*, \mathbf{X}) [\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}(\mathbf{X}, \mathbf{X}^*) \quad (2.19)$$

The mean of this distribution $\bar{\mathbf{y}}^* \in \mathbb{R}^{n^*}$ is a linear combination of the observed target values \mathbf{y} , whereas the variance $\mathbf{COV}(\mathbf{y}^*) \in \mathbb{R}^{n^* \times n^*}$ only depends on the features. Equations 2.18 and 2.19 give the mean function and the covariance function of the *posterior* process, respectively. Finally, a Gaussian Process (GP) can be seen as a distribution over functions f_{GP} , which is completely defined by its mean function mf and its covariance function, also

called kernel, k :

$$f_{\mathcal{GP}}(\mathbf{x}) \sim \mathcal{GP}(\mathbf{mf}(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (2.20)$$

Often, the mean function \mathbf{mf} is set to zero for simplicity, which is not a significant limitation as the mean of the *posterior* given in Equation 2.18 is not restricted to zero. The kernel however is an essential choice when building a GPR model and reflects the similarity between data points \mathbf{x} and \mathbf{x}' .

Kernel Functions

A kernel is a function k that maps the inputs $\mathbf{x} \in \mathcal{X}$ and $\mathbf{x}' \in \mathcal{X}$ into \mathbb{R} , which is symmetric by definition $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. We can further define the covariance matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ containing $K_{i_*} = k(\mathbf{x}_i, \mathbf{x}_{i_*})$ for n inputs \mathbf{x}_i . A real matrix has to be positive semidefinite to be a valid covariance matrix. Common base kernels describing specific patterns are, for instance, the following ones:

- Linear: $k_{Lin}(\mathbf{x}, \mathbf{x}') = \sigma^2(\mathbf{x} - z)(\mathbf{x}' - z)$, with variance $\sigma^2 \in \mathbb{R}$ and offset $z \in \mathbb{R}$
- SquaredExponential: $k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{(\mathbf{x} - \mathbf{x}')^2}{2\iota^2}\right)$,
with variance σ^2 and lengthscale $\iota \in \mathbb{R}_{>0}$
- Rational Quadratic: $k_{RQ}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \frac{(\mathbf{x} - \mathbf{x}')^2}{2\tau\iota^2}\right)^{-\tau}$,
with variance σ^2 , lengthscale ι , and scale mixture parameter $\tau \in \mathbb{R}_{>0}$
- Matérn52: $k_{Mat}(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\mathbf{x} - \mathbf{x}'}{\iota}\right)^\nu K_\nu\left(\sqrt{2\nu} \frac{\mathbf{x} - \mathbf{x}'}{\iota}\right)$,
here with $\nu \in \mathbb{R}_{>0} = 5/2$, variance σ^2 , lengthscale ι , and a modified Bessel function K_ν

For each of them, we visualize the kernel and show samples drawn from a GP parameterized with the respective kernel in Figure 2.1a. We observe a distinct behavior for each base kernel, which we can use to define the *prior* of the GP to specify which structures are likely. It is also possible to combine kernels by multiplication and summation while still meeting the requirements for a valid kernel. Composite kernels can model a more complex system behavior, which is often required in practice. In Figure 2.1b, we visualize some examples of combined kernels based on the base kernels given above. For instance, we observe that the product of two linear kernels can be used to model a quadratic function and that the summation of a linear and a periodic kernel shows a periodic behavior with a linear trend (Bishop, 2006; Duvenaud et al., 2013; Rasmussen & Williams, 2006).

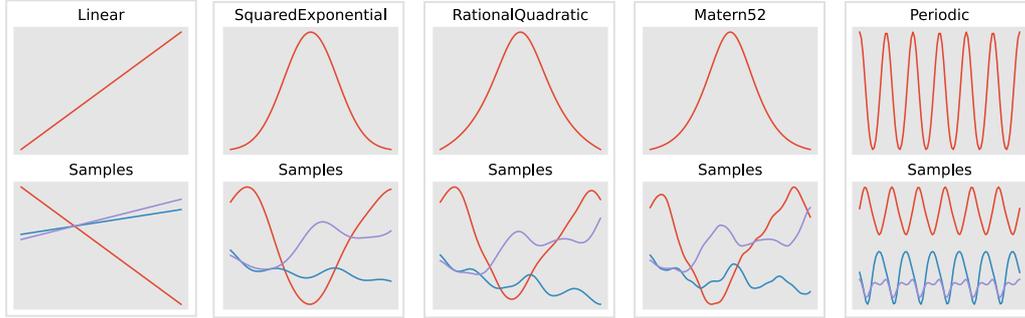
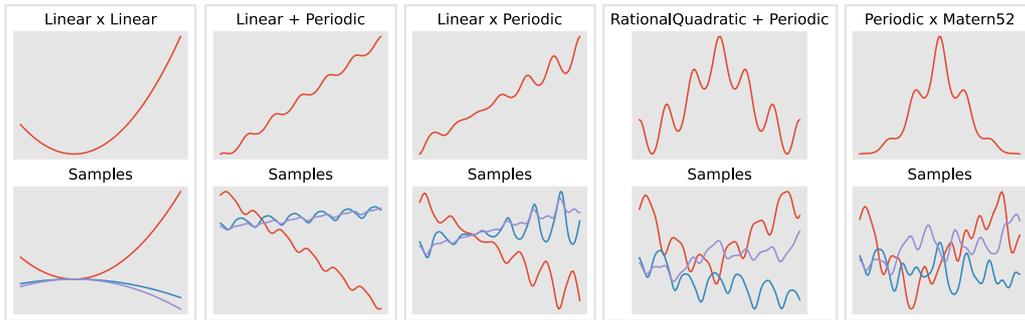
(a) Examples of base kernels**(b) Examples of combined kernels**

Figure 2.1: **Examples of kernels k** : For each kernel, we visualize the kernel $k(\cdot, 1)$ in the upper part of the plot as well as samples drawn from a GP with the respective kernel. The x-axes have the same range for all plots. (a) Examples of base kernels. (b) Examples of combined kernels created by adding or multiplying base kernels.

Kernel Search

The definition of the kernel function is a decisive choice for a GP. However, there are many options, especially when considering combinations of base kernels. Since thus a manual selection is difficult, even with expert knowledge, we need automatic data-driven search methods to find the kernel that best describes the data to be modeled. A common criterion to evaluate how well a $\mathcal{GP}(\mathfrak{m}_f, k)$ with mean function \mathfrak{m}_f and kernel k fits for a dataset \mathcal{D} with n samples is the log marginal likelihood $\mathcal{L}_{lml}(\mathcal{GP}(0, k), \mathcal{D})$ defined as follows:

$$\begin{aligned} \mathcal{L}_{lml}(\mathcal{GP}(0, k), \mathcal{D}) &= \log p(\mathbf{y} \mid \mathbf{X}) = \\ &= -\frac{1}{2} \mathbf{y}^T (\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K}(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I}| - \frac{n}{2} \log 2\pi \end{aligned} \quad (2.21)$$

For simplicity, we set the mean function to constant zero (Rasmussen & Williams, 2006).

In this thesis, we consider three kernel search approaches: treating the kernel as a hyperparameter during the optimization process, Compositional Kernel Search (CKS) (Duvenaud et al., 2013), and Adjusting Kernel Search (AKS) (Hüwel et al., 2021). The first option is straightforward, as we only need to generate different combinations by summation and multiplication, i.e., a set of kernel expressions \mathcal{K} , based on a set of base kernels \mathcal{B} . Then, we consider the choice of the kernel $k \in \mathcal{K}$ as a hyperparameter during optimization by evaluating the performance based on a criterion, see Section 2.4.

Compositional Kernel Search (CKS) however is a stage-wise algorithm, iteratively yielding the final kernel expression k . For the formulation of CKS, we first need to define the following operations:

- *AddSums*(k, \mathcal{B}): generate composed kernel expressions $k_c = k + k_{base}$ by adding any base kernel $k_{base} \in \mathcal{B}$ to the kernel expression k
- *AddProducts*(k, \mathcal{B}): generate composed kernel expressions $k_c = k k_{base}$ by multiplying the kernel expression k with any base kernel $k_{base} \in \mathcal{B}$
- *Replace*(k, \mathcal{B}): generate composed kernel expressions k_c by replacing any base kernel in k with any base kernel $k_{base} \in \mathcal{B}$

Algorithm 1 Pseudocode of the Compositional Kernel Search (CKS)

Require: dataset $\mathcal{D} = \{(x_t, y_t) \mid t = 1, \dots, n\}$, set of base kernels \mathcal{B} , maximum number of iterations i_{CKS}

```

1:  $k \leftarrow \underset{k_{base} \in \mathcal{B}}{\operatorname{argmax}} (\mathcal{L}_{lml}(\mathcal{GP}(0, k_{base}), \mathcal{D}))$  ▷ see Eq.(2.21)
2: for  $i = 0, \dots, i_{CKS} - 1$  do
3:    $\mathcal{K} \leftarrow k$ 
4:    $\mathcal{K} \leftarrow \mathcal{K} \cup \operatorname{AddSums}(k, \mathcal{B})$ 
5:    $\mathcal{K} \leftarrow \mathcal{K} \cup \operatorname{AddProducts}(k, \mathcal{B})$ 
6:    $\mathcal{K} \leftarrow \mathcal{K} \cup \operatorname{Replace}(k, \mathcal{B})$ 
7:    $k_{new} \leftarrow \underset{k_c \in \mathcal{K}}{\operatorname{argmax}} (\mathcal{L}_{lml}(\mathcal{GP}(0, k_c), \mathcal{D}))$  ▷ see Eq.(2.21)
8:   if  $k_{new} == k$  then
9:     terminate
10:  else
11:     $k \leftarrow k_{new}$ 

```

Algorithm 1 outlines the procedure of CKS. Besides the dataset \mathcal{D} , we need to define a set of base kernels \mathcal{B} as well as a maximum number of iterations $i_{CKS} \in \mathbb{N}$. Then, CKS first

chooses the most appropriate base kernel $k_{base} \in \mathcal{B}$ based on the log marginal likelihood $\mathcal{L}_{lml}(\mathcal{GP}(0, k_{base}), \mathcal{D})$ given in Equation 2.21. After this step, the iterative process starts for, at most, i_{CKS} runs, defining the maximum number of elements used for the final kernel expression. In each of the iterations, a set of candidate kernels \mathcal{K} is generated using the operations $AddSums(k, \mathcal{B})$, $AddProducts(k, \mathcal{B})$, and $Replace(k, \mathcal{B})$. Among these, we then choose again the composed kernel expression k_c with the highest log marginal likelihood $\mathcal{L}_{lml}(\mathcal{GP}(0, k_c), \mathcal{D})$. If the found kernel expression k_{new} does not differ from k , the algorithm terminates, and continues otherwise. At the latest after i_{CKS} runs, we retrieve the final kernel expression k . By doing so, complex kernels can be generated in a data-driven way at the cost of multiple optimizations and evaluations per iteration (Duvenaud, 2014; Duvenaud et al., 2013).

Algorithm 2 Pseudocode of the Adjusting Kernel Search (AKS)

Require: dataset $\mathcal{D} = \{(x_t, y_t) \mid t = 1, \dots, n\}$, set of base kernels \mathcal{B} , maximum number of iterations i_{AKS} , starting kernel k_0

```

1:  $k \leftarrow k_0$ 
2: for  $i = 0, \dots, i_{AKS} - 1$  do
3:    $\mathcal{K} \leftarrow k$ 
4:    $\mathcal{K} \leftarrow \mathcal{K} \cup AddSums(k, \mathcal{B})$ 
5:    $\mathcal{K} \leftarrow \mathcal{K} \cup AddProducts(k, \mathcal{B})$ 
6:    $\mathcal{K} \leftarrow \mathcal{K} \cup Replace(k, \mathcal{B})$ 
7:    $\mathcal{K} \leftarrow \mathcal{K} \cup Remove(k, \mathcal{B})$ 
8:    $k_{new} \leftarrow \underset{k_c \in \mathcal{K}}{\operatorname{argmax}} (\mathcal{L}_{lml}(\mathcal{GP}(0, k_c), \mathcal{D}))$  ▷ see Eq.(2.21)
9:   if  $k_{new} == k$  then
10:    | terminate
11:   else
12:    |  $k \leftarrow k_{new}$ 

```

Since the complexity of a \mathcal{GP} is $\mathcal{O}(n^3)$ and CKS always starts from scratch, this can be computationally expensive. AKS is based on CKS but addresses this issue by not starting from scratch and also considering kernel expressions with a lower complexity during each iteration. To achieve this, AKS defines a further operation $Remove(k, \mathcal{B})$ generating composed kernel expressions k_c by removing any base kernel $k_{base} \in \mathcal{B}$ from k . With this extension, AKS can be used to adjust an initial kernel expression k_0 that was, for instance, found with CKS on an initial window of data points, potentially requiring less maximum iterations $i_{AKS} \in \mathbb{N}$. Since we consider time series data with continuously incoming new data points, AKS may suit well, especially when assuming that the best-performing kernels do not differ entirely for consecutive segments. Due to the additional operation

$Remove(k, \mathcal{B})$, AKS considers $|k|$ (number of base kernels in k) more kernel expressions for each iteration. Nevertheless, Hüwel et al. (2021) show that AKS can work with fewer iterations and lead to lower runtimes than CKS, especially if more complex kernel expressions are necessary for which starting from scratch is computationally more expensive than a few iterations of AKS. However, for low-complexity datasets and significant changes of the kernel expression, AKS can be inferior in terms of runtime (Hüwel et al., 2021).

2.4 Model Selection and Evaluation

After outlining several time series forecasting methods, we describe different data splitting and hyperparameter optimization approaches. Finally, we introduce several evaluation metrics and simple baselines.

2.4.1 Data Splits

The classical time series forecasting approaches described above and some ML-based techniques, e.g., LSTM networks, require a chronological order of samples. This requirement must be taken into account when splitting the data. In a regular j -fold cross-validation, we split the dataset \mathcal{D} , or a subset of it if we separate samples for a test set in advance, into $j \in \mathbb{N}$ subsets as equal in size as possible. Then, we run j iterations using one of the subsets for validation and the remaining ones for training (Bishop, 2006). With this procedure, we neglect the chronological order of the dataset, which makes this split inappropriate for some of the applied methods. Another option would be to split the whole dataset into three subsets for training, validation and test according to the chronological order. This approach maintains the temporal order, but has the disadvantage that the model selection is based on the results obtained on a single validation set.

Therefore, time series cross-validation may be a good option to estimate the performance of a model based on multiple validation sets while accounting for the chronological order. In Figure 2.2a, we visualize time series cross-validation with three folds and a separate test set. First, we assign a certain amount of the most recent samples to the test set, for example, 20% of the entire dataset. Then, we use the remaining data for a j -fold time series cross-validation, where we divide the data into $j + 1$ equally sized parts, taking into account the chronological order. For the first fold, we use the first subset for training and the second for validation. Then, these two subsets form the training set for the second fold, where we validate using the third subset. This process continues until we reach the last

fold. Finally, we usually average the performance across all validation sets and retrain the best-performing model using the whole training and validation data. This prediction model can then be tested on the initially held-out test data. However, depending on the size of the dataset, time series cross-validation can be problematic, leading to folds with a small number of samples. This is particularly relevant for seasonal data, where each training and validation set should contain at least one full season. In such cases, the aforementioned split into three subsets for training, validation, and test, while maintaining the temporal order, may be more appropriate (Bishop, 2006; Hyndman & Athanasopoulos, 2021).

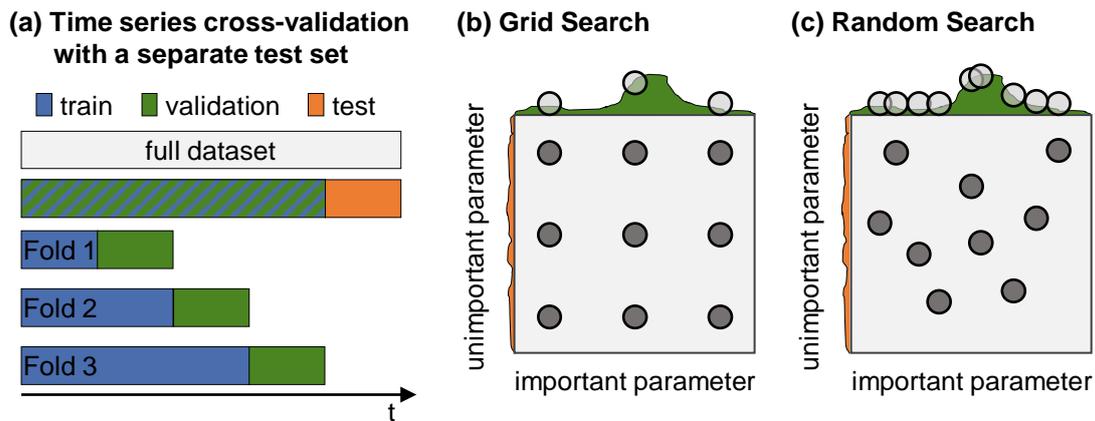


Figure 2.2: **Time series cross-validation, Grid and Random Search:** (a) Visualization of a time series cross-validation with a separate test set shown for three folds used for training and validation. (b) and (c): Visualization of Grid and Random Search, see Section 2.4.2. We show nine trials for optimizing two parameters, of which one is important, i.e., influencing the prediction result significantly. With Grid Search, only three distinct places of the important parameter are tested, whereas Random Search tests a different place in each trial. Figures (b) and (c) are similar to (Bergstra & Bengio, 2012).

2.4.2 Hyperparameter Optimization

Model selection is usually driven by the performance on validation data, as described in the previous section. Many prediction models, especially the ML-based approaches, contain hyperparameters that significantly affect their performance, such as the number of layers in a neural network. Thus, in addition to selecting the overall best-performing model, the performance on validation data is usually also used to optimize the models' hyperparameters. With a hyperparameter set $\theta \in \mathbb{R}^{n_h}$ containing values for $n_h \in \mathbb{N}$ hyperparameters, which we assume to be real numbers, and a hyperparameter space $\Theta \in \mathbb{R}^{n_h \times n_t}$ containing all

$n_t \in \mathbb{N}$ hyperparameter sets, we can define

$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta} \in \Theta}{\operatorname{argmin}} \mathcal{L}_v(\boldsymbol{\theta}), \quad (2.22)$$

where $\mathcal{L}_v(\boldsymbol{\theta}) \in \mathbb{R}$ is the validation loss achieved using a given parameter set $\boldsymbol{\theta}$, and $\boldsymbol{\theta}^* \in \mathbb{R}^{n_h}$ is the one that yields the best result, assuming we want to minimize the loss.

A straightforward option in this context is Grid Search. With this method, we define certain values for each hyperparameter and build the hyperparameter space Θ using the combinations of all these values. After evaluating all of these, we choose the parameter setting with the best validation performance. In Figure 2.2b, we visualize Grid Search for two parameters, of which one is important, i.e., significantly influencing the performance, and the other one is not. We observe that evaluating different values for the unimportant parameter does not affect the result much. Furthermore, Grid Search is usually computationally exhaustive. For instance, evaluating three hyperparameters with ten possible values for each would already lead to trying $n_t = 10^3 = 1000$ configurations, of which many might not influence the performance much. Thus, this approach does not scale well to higher dimensions with several hyperparameters and values (Murphy, 2022).

In contrast to Grid Search, Random Search does not employ a fixed grid of parameter settings, but randomly samples from a distribution defined for each hyperparameter. Thus, we do not need to specify fixed hyperparameter values, which may be difficult even with expert knowledge, but only an upper and lower bound for a usually applied uniform distribution. We can further set the number of trials to run, i.e., the width of the hyperparameter space Θ , which is fixed by the number of combinations for Grid Search. As we observe in Figure 2.2c, this leads to evaluating multiple values for the important and unimportant parameter in contrast to Grid Search, where we define certain values in advance. In the given example, both approaches run nine trials, but six are uninformative for Grid Search. Beyond this example, it has been shown empirically and theoretically that Random Search is more efficient (Bergstra & Bengio, 2012; Murphy, 2023).

Both Grid and Random Search do not use information gained during the optimization process, which is a drawback compared to Bayesian optimization. In Bayesian optimization, the hyperparameter space Θ is not fixed in advance, but hyperparameter sets $\boldsymbol{\theta}_i$ are suggested based on the performance of previously tested ones. In Algorithm 3, we show the procedure of a sequential model-based optimization algorithm, which is commonly used for Bayesian optimization. We may start the optimization with $n_{finished} \in \mathbb{N}$ previously

tested hyperparameter combinations with corresponding objective values, gathered in \mathcal{H} , and an initial surrogate model \mathcal{M}_0 . First, we must decide which parameter setting θ_i to test based on a particular policy. Then, we can use θ_i to observe the corresponding objective value $ov_i \in \mathbb{R}$, e.g., by running a cross-validation and retrieving the performance. With this result, we can extend the set of historical observations \mathcal{H} , which we then use to update the surrogate model \mathcal{M}_i . This procedure is repeated until we reach a termination criterion; in terms of Bayesian optimization, for instance, the maximum number of trials. Finally, we retrieve the parameter set θ^* that yields the best objective value ov .

Algorithm 3 Pseudocode of a sequential model-based optimization algorithm

Require: historical observations $\mathcal{H} = \{(\theta_i, ov_i) \mid i = 1, \dots, n_{finished}\}$, initial surrogate model \mathcal{M}_0 $\triangleright \mathcal{H}$ may be empty

- 1: $\mathcal{M}_i \leftarrow \mathcal{M}_0$
- 2: **repeat**
- 3: $\theta_i \leftarrow POLICY(\mathcal{H}, \mathcal{M}_i)$
- 4: $ov_i \leftarrow OBSERVE(\theta_i)$
- 5: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\theta_i, ov_i)\}$
- 6: $\mathcal{M}_i \leftarrow UPDATE(\mathcal{M}_i, \mathcal{H})$
- 7: **until** termination condition \triangleright e.g., maximum number of trials reached

\triangleright may be minimum or maximum

Hence, we need to select approaches for the *POLICY* and *OBSERVE* steps and a surrogate model. For the *OBSERVE* step, it is common to use performance on validation data as the objective value ov that we want to optimize. With respect to the surrogate model, a GP can be constructed using the historical observations \mathcal{H} . Another possibility is the so-called tree-structured Parzen estimator, a lightweight implementation that reduces the problem to density ratio estimation. Based on the surrogate model, we can choose a *POLICY* to select the parameter values for the next iteration. One of the most common approaches is expected improvement, for which we define the improvement function $i: \mathbb{R}^{n_h} \rightarrow \mathbb{R}_{\geq 0}$ with $i(\theta) = \max(0, of_{t+1}(\theta) - of(\theta^+))$. The objective function $of: \mathbb{R}^{n_h} \rightarrow \mathbb{R}$ returns the objective value ov_i and θ^+ is the hyperparameter configuration that currently yields the best result. We can now determine the expected improvement $EI(\theta) = \mathbb{E}[i(\theta)]$, and choose the parameter values with the largest expected improvement over the current best configuration. With respect to hyperparameter optimization, the *OBSERVE* step is usually computationally expensive. In this step, we need to train a prediction model parameterized with the selected hyperparameter values and observe its performance, which requires, for example, to run a full cross-validation. Thus, despite the resources spent on

the surrogate model and the policy, Bayesian optimization is potentially more efficient than Grid or Random Search, requiring fewer iterations to converge, i.e., less *OBSERVE* steps are necessary (Bergstra et al., 2011; Garnett, 2023; Snoek et al., 2012).

2.4.3 Evaluation Metrics and Baselines

We further require metrics that we can use for model selection and evaluation. For a dataset \mathcal{D} with n samples, the true target value y_t , and the predicted value \hat{y}_t at time step t , we can define the following common criterions:

- Mean absolute error (MAE): $MAE = \frac{1}{n} \sum_{t=1}^n |y_t - \hat{y}_t|$
- Mean squared error (MSE): $MSE = \frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2$
- Root mean squared error (RMSE): $RMSE = \sqrt{\frac{1}{n} \sum_{t=1}^n (y_t - \hat{y}_t)^2}$
- Mean absolute percentage error (MAPE): $MAPE = \frac{100\%}{n} \sum_{t=1}^n \left| \frac{y_t - \hat{y}_t}{y_t} \right|$
- Symmetric mean absolute percentage error (sMAPE):
 $sMAPE = \frac{100\%}{n} \sum_{t=1}^n \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2}$

A lower value reflects a better performance for all five metrics. MAE, MSE, and RMSE are scale-dependent and can be used to compare the results of different methods on the same dataset, but not across datasets. MSE and RMSE are furthermore sensitive to outliers, although RMSE has the advantage of being on the same scale as the data. Despite these properties, these three metrics are still widely used, i.a., because of their interpretability and straightforward computation. Furthermore, MAPE and sMAPE are two common metrics that are percentage errors and thus scale-independent. This property allows to compare results across datasets. To prevent division by zero, which is particularly an issue for intermittent sales data, a small value is often added to the denominator. Although this problem is more likely with MAPE, it can also occur with sMAPE, since the prediction can also be zero for time steps with a true value of zero. MAPE is known to penalize negative errors more than positive ones. This means that for the same absolute difference, cases where $\hat{y}_t > y_t$ leads to a higher MAPE. For example, $\hat{y}_t = 150$ with $y_t = 100$ gives a MAPE of 50.0%, but $\hat{y}_t = 100$ with $y_t = 150$ results in 33.3%. This problem does not

occur for sMAPE, which yields the same values in such cases. However, if we consider an evaluation metric to be symmetric if the same deviation upwards and downwards from a fixed true value leads to the same result, then MAPE would be symmetric in contrast to sMAPE (Flores, 1986; Hyndman & Athanasopoulos, 2021; Hyndman & Koehler, 2006).

Because of this lack of a universal evaluation criterion, it is common to include simple baseline models as comparison partners. With a true value y_t , a season length n_{seas} , and a window size n_w , the predicted value at time step \hat{y}_t for the given simple baselines is defined as follows:

- Historical average: $\hat{y}_t = \frac{1}{t-1} \sum_{i=1}^{t-1} y_i$
- Moving average: $\hat{y}_t = \frac{1}{n_w} \sum_{i=t-n_w}^{t-1} y_i$
- Random walk / Naïve method: $\hat{y}_t = y_{t-1}$
- Seasonal random walk / Seasonal naïve method: $\hat{y}_t = y_{t-n_{seas}}$

For many time series, these simple approaches work surprisingly well (Hyndman & Athanasopoulos, 2021; Hyndman & Koehler, 2006)

2.5 Change Point Detection

Changing data distributions are a challenge in time series forecasting, as outlined in Section 1.2. A straightforward yet computationally exhaustive approach to account for this issue is to periodically trigger a refitting of the prediction model during the live operation, i.e., the online phase with new samples becoming available continuously (Hyndman & Athanasopoulos, 2021). However, this can lead to wasting computational resources in case of unnecessary model adjustments or working with outdated models for less frequent triggers. Another option is to trigger a model refitting event-based using a CPD method. The time step t at which a shift of the data distribution $p(y_t | x_t)$ occurs is called a change point. In the context of this thesis, in particular, for the contributions outlined in Sections 3.2 and 3.3, we want to detect change points for sequentially arriving data points. Thus, we do not consider offline approaches that work retrospectively, such as change point kernels for GPR that switch between kernel expressions at certain time steps. Besides working online, the CPD method should enable a quick reaction requiring only a few samples of the new data distribution and be computationally efficient to facilitate

the integration in a forecasting system. A simple yet commonly used CPD approach is cumulative sum (CUSUM). Thereby, deviations from an expected value, e.g., the difference between predictions and observed values, are summed up and a change point is declared if this score exceeds a certain threshold (Page, 1954).

ChangeFinder applies a two-stage learning and smoothing strategy to detect a change point, reformulating the problem to outlier detection in time series. First, ChangeFinder fits a sequentially discounting autoregressive (SDAR) model on incoming data points of \mathbf{y} , see Section 2.1, gradually lowering the influence of older samples. Using this SDAR model, we can determine a probability density p_t^{CF} at time step t . Based on the sequential updating at each time step, we obtain a sequence of probability densities $p_1^{CF}, \dots, p_t^{CF}$. Then, at the end of the first learning stage, we can determine a so-called outlier score $score(\mathbf{y}_t) = -\log p_{t-1}^{CF}(\mathbf{y}_t)$. In the first smoothing stage, we calculate a moving average within a time window n_w of these outlier scores, yielding a sequence of smoothed outlier scores:

$$\mathbf{y}_t^{CF} = \frac{1}{n_w} \sum_{i=t-n_w+1}^t score(\mathbf{y}_i) \quad (2.23)$$

We can interpret $\mathbf{y}^{CF} \in \mathbb{R}^t$ as the difference between consecutive time periods. Using \mathbf{y}^{CF} as input, we perform the same steps in a second learning and smoothing stage to identify abrupt changes between two consecutive differences. Finally, we obtain a sequence of change point scores $\mathbf{z}^{CF} \in \mathbb{R}^t$. A higher score indicates a higher probability of a distributional shift, requiring the specification of a threshold to declare a change point. The smoothing lowers the risk of false alarms due to isolated outliers in \mathbf{y} . Larger window sizes n_w for the moving average calculation lead to a less sensitive CPD by filtering out outliers and only detecting significant change points. However, this might result in a delayed identification of a change point. Smaller window sizes instead enable quick detection but the risk of false alarms due to outliers is higher (Aminikhanghahi & Cook, 2017; Iwata et al., 2019; Takeuchi & Yamanishi, 2006).

Bayesian Online Change Point Detection (BOCPD) is a probabilistic method assuming that a time series can be divided into non-overlapping intervals, each of which is related to a certain probability distribution. BOCPD defines a so-called run length $r_t \in \mathbb{N}_0$ at time step t , which indicates the time elapsed since the last change point occurred and is set to zero when a change point is identified. Thus, we can determine the *posterior* distribution

of r_t using Bayes' theorem as

$$p(r_t | \mathbf{y}_{i=1}^t) = \frac{\sum_{r_{t-1}} p(r_t | r_{t-1}) p(\mathbf{y}_t | r_{t-1}, \mathbf{y}_t^r) p(r_{t-1}, \mathbf{y}_{i=1}^{t-1})}{\sum_{r_t} p(r_t, \mathbf{y}_{i=1}^t)}, \quad (2.24)$$

where $\mathbf{y}_t^r \in \mathbb{R}^{r_t}$ are the observations associated with the run length r_t and $\mathbf{y}_{i=1}^t$ are all observations from time step 1 to t . The *prior* $p(r_t | r_{t-1})$ is defined as

$$p(r_t | r_{t-1}) = \begin{cases} H(r_{t-1} + 1) & \text{if } r_t = 0 \\ 1 - H(r_{t-1} + 1) & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases} \quad (2.25)$$

with the so-called hazard function $H(t') = \frac{p_{gap}(g=t')}{\sum_{t=t'}^{\infty} p_{gap}(g=t)}$, i.e., the ratio between the probability density of the current run and the sum of all probability densities, where p_{gap} denotes the probability distribution of the interval between change points. The computation of the *prior* is thus efficient, as it only contains two non-zero cases, which both only require the calculation the hazard function's value. With respect to p_{gap} , we could assume a geometric distribution with success probability $p_s \in]0, 1]$, leading to a hazard function not depending on time $H(t') = p_s$. Finally, we can use the *posterior* distribution of the run length given in Equation 2.24 to determine whether a change point occurred (Adams & MacKay, 2007; Aminikhanghahi & Cook, 2017).

Results

Subsequently, the results of this thesis based on the four main publications given in Section 1.5.1 are described. For each publication, we show the bibliographic information and give the individual author contributions before summarizing the content and outlining the key findings. For one section, we include further unpublished results obtained in additional experiments after the time of publication.

3.1 Time Series Forecasting for Small and Medium-Sized Companies

The following publication, for which the original full version can be found in Appendix A, is summarized in this section:

PUBLICATION A:

Florian Haselbeck, Jennifer Killinger, Klaus Menrad, Thomas Hannus, and Dominik G. Grimm (2022). Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions. *Machine Learning with Applications*, 7, 100239. <https://doi.org/10.1016/j.mlwa.2021.100239>

The individual contributions of the authors are as follows: Florian Haselbeck and Dominik G. Grimm conceptualized the study. Florian Haselbeck implemented the analysis pipeline, prepared the data, and performed the experiments. Florian Haselbeck and Dominik G. Grimm analyzed and interpreted the results with the help of Jennifer Killinger. Florian Haselbeck created the original manuscript draft, which was reviewed and edited by Dominik G. Grimm with the support of Klaus Menrad, Thomas Hannus, and Jennifer Killinger.

3.1.1 Summary

Despite sales in the billions in Germany alone, potential economic and environmental benefits, and several scientifically interesting characteristics such as abrupt changes in demand, horticulture is a domain with limited attention in forecasting research so far, see Section 1.2 for more details. To assess the potential of time series forecasting methods to predict horticultural sales despite changing data distributions, we presented a comparative study using three classical forecasting and nine ML-based approaches on five retail sales datasets. With respect to classical forecasting methods, we included both the univariate approaches Exponential Smoothing and SARIMA as well as the multivariate extension of the latter SARIMAX (Brown, 1956; Holt, 1957; Hyndman & Athanasopoulos, 2021; Winters, 1960). Regarding ML-based models, we applied linear regression with different regularization terms, both in a frequentist and probabilistic formulation (Hoerl & Kennard, 1970; Tibshirani, 1996; Zou & Hastie, 2005). We further employed the ensemble learner XGBoost, two neural network-based approaches with an MLP and an LSTM network, and the nonparametric Bayesian model GPR (Chen & Guestrin, 2016; Goodfellow et al., 2016; Hochreiter & Schmidhuber, 1997; Rasmussen & Williams, 2006; Williams & Rasmussen, 1995). For details regarding the forecasting methods, see Sections 2.1, 2.2, and 2.3.

For our study, we used five horticultural retail sales datasets from Germany based on two data sources, see Figure 3.1a. Two datasets were manually documented and contain daily sales numbers of tulips over one three-month-long season with abrupt changes in demand. The manual documentation shows a 16-day-long gap with missing values after two months, which is why we derived two datasets: one only containing values before this gap (*OwnDoc_SoldTulips_short*) and one for which we imputed the missing values (*OwnDoc_SoldTulips_long*). The other three datasets were collected using an electronic cashier system and show weekly sales numbers from December 2016 to August 2020 without missing values. In contrast to the two manually documented ones, they contain multiple seasons and revenues are aggregated into product groups. Based on this second data source, we created a dataset with sales of cut flowers (*CashierData_CutFlowers*) and two datasets showing sales of potted plants. For potted plants, we can observe a sharp increase in revenue in the first half of 2020, probably due to the SARS-CoV-2 pandemic. We thus distinguish between *CashierData_PotTotal_short* ending in December 2019 and *CashierData_PotTotal_long* containing the whole time period to assess whether the prediction models are able to handle this change in the data distribution.

(a) Overview of datasets and data sources

 data sources	OwnDoc <i>manual documentation</i>	CashierData <i>electronic cashier system</i>
 datasets	OwnDoc_SoldTulips_short 07.02. – 09.04.2020	CashierData_CutFlowers 12/2016 – 08/2020
	OwnDoc_SoldTulips_long 07.02. – 11.05.2020	CashierData_PotTotal_short 12/2016 – 12/2019
		CashierData_PotTotal_long 12/2016 – 08/2020

(b) Featureset compositions and preprocessing methods

features	 univariate <i>target variable</i>	 sub1 <i>raw features</i>	 sub2 <i>raw & calendric feat.</i>	 sub3 <i>raw & statistical feat.</i>	 full <i>all features</i>
imputation	✓ none	✓ none	✓ none	✓ none	✓ none
	∅ mean	∅ mean	∅ mean	∅ mean	∅ mean
	✓ k-NN	✓ k-NN	✓ k-NN	✓ k-NN	✓ k-NN
	⌚ iterative	⌚ iterative	⌚ iterative	⌚ iterative	⌚ iterative
	<i>not included for CashierData</i>				
dimensionality reduction	✓ none	✓ none	✓ none  PCA	✓ none  PCA	✓ none  PCA

Figure 3.1: **Datasets, featuresets, and pre-processing methods for our study:** This figure contains parts of Figures 1 and 2 of the original publication and is shown for better readability (Haselbeck et al., 2022). (a) Overview of the five horticultural retail sales datasets based on two data sources. (b) Composition of the four featuresets that we used in addition to the univariate case as well as the missing value imputation and dimensionality reduction methods we considered for each featureset.

We enriched all datasets with features from external sources, namely public and school holidays as well as weather-related information, for example, the mean temperature and sun duration. Subsequently, these are referred to as raw features. We also derived calendric and statistical features to assess their influence. Regarding calendric features, we determined, i.a., date-based information such as the month and countdowns indicating special days that could increase demand, e.g., Valentine’s Day. In terms of statistical features, we considered past sales and weather information, as both could influence future horticultural demand. For instance, we determined mean values within a period before the time of sale, calculated features using the information at a similar time step in previous seasons for the

three datasets containing multiple years, and derived weekday-specific sales statistics for the two datasets with daily time resolution. For more detailed information regarding the features, see Section 3.1.2 of the original publication in Appendix A. As shown in Figure 3.1b, in addition to the univariate case, these features enabled us to define four featuresets. Besides the raw features, these include the following ones: no additional features (*sub1*), with additional calendric features (*sub2*), with additional statistical features (*sub3*), and with both (*full*).

The inclusion of information from previous seasons as part of the statistical features results in missing values for the beginning of the dataset. Furthermore, as outlined, there is a 16-day-long documentation gap in *OwnDoc_SoldTulips_long*. Since most forecasting methods cannot handle missing values, we considered three imputation methods: mean, k-Nearest Neighbor (k-NN), and iterative imputation. Both mean and k-NN use average values for imputation, but whereas the former uses all, k-NN determines the k closest training samples. Iterative imputation instead tries to predict missing values using all other features. We did not consider all imputation methods for every feature setting, see Figure 3.1b. *CashierData* and the calendric features do not have missing values, and the raw weather information only contains a few, so the effect of testing different approaches seems negligible for these variants. For dimensionality reduction on the larger featuresets, we applied a Principal Component Analysis (PCA) selecting components that explain at least 95 % of the variance (Figure 3.1b).

To simulate the productive operation of a forecasting system with a continuous update of sales data and to account for potentially changing data distributions, we applied time series cross-validation with a regular model refitting. We used 80 % of the whole dataset for hyperparameter tuning with a Random Search, with a second in-depth optimization for the best-working configurations. With the remaining data, we simulated the above-mentioned productive operation, predicting the next value and then refitting the prediction model with fixed hyperparameters for each new sample. For evaluation, we averaged the performance over the whole test set using the metrics RMSE, MAPE and sMAPE, see Section 2.4. We further conducted runtime experiments, for which we ran a full training cycle using 100 randomly sampled parameter combinations and the test procedure.

Considering the results for all 15 comparisons (five datasets and three evaluation metrics) visualized in Figure 3 of our publication (Appendix A), an ML-based approach performed best in each of them. The ensemble learner XGBoost was the clear winner with the best results in 13 out of 15 comparisons, often closely followed by LSTM and GPR, and twice

outperformed by the former. This observation also holds with respect to the in-depth optimization of the best-performing configurations, after which XGBoost surpassed LSTM in one further comparison. While the performance of the classical forecasting methods was relatively comparable for the smaller *OwnDoc* datasets, the advantage of the ML-based models increased for *CashierData*. For *CashierData_PotTotal_long*, which contains a sharp increase in sales numbers, we observed a worse performance than for the other two datasets based on *CashierData*. Thus, all prediction models were limited in handling this change in the output scale of the data. Interestingly, the univariate method SARIMA outperformed its multivariate extension SARIMAX several times. Nevertheless, the superiority of the ML-based techniques generally suggests that external factors are important for predicting horticultural demand. We observed that the best-performing featureset depends on the prediction model and the dataset, with no clear overall winner. For *OwnDoc*, we identified advantages for the *full* featureset, whereas *sub2* containing raw and calendrical features worked best for *CashierData*. The feature importances of the top performer XGBoost also showed varying results in terms of the most influential features for the different datasets. Regarding the runtime, linear regression approaches and univariate Exponential Smoothing required the lowest resources, but XGBoost also proved its efficiency.

3.1.2 Key Findings

In this first-time comparative study, we showed that time series forecasting approaches are promising for predicting horticultural demand, especially ML-based methods and in particular the ensemble learner XGBoost. Moreover, external factors such as holiday and weather-related information were beneficial, even though we could not identify a best-performing featureset for all prediction models and datasets. However, all prediction models were limited in handling the change in the output scale for *CashierData_PotTotal_long* despite a regular model refitting scheme. For more general conclusions, our findings need to be verified in a broader study using data from multiple companies along the entire value chain and products with diverse characteristics. Besides these key findings, we identified two factors that impede demand forecasting. First, as already explained, changing data distributions as observed for *CashierData_PotTotal_long* are challenging, probably requiring methods that specifically account for this issue. In Sections 3.2 and 3.3, two approaches designed for this purpose are outlined. Second, no comprehensive and state-of-the-art time series forecasting framework is available that is easy to use and extend to ensure reproducible and comparable results as well as accessibility of novel approaches. For this reason, we have published ForeTiS, which is introduced in Section 3.4.

3.2 Event-Triggered Augmented Refitting of Gaussian Process Regression

The following publication, for which the original full version can be found in Appendix B, is summarized in this section:

PUBLICATION B:

Florian Haselbeck and Dominik G. Grimm (2021). EVARS-GPR: Event-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data. *KI 2021: Advances in Artificial Intelligence*, 12873, 135–157. https://doi.org/10.1007/978-3-030-87626-5_11

The individual contributions are as follows: Florian Haselbeck and Dominik G. Grimm developed the presented method. Florian Haselbeck implemented the code, prepared the simulated and the real-world data, and performed the experiments. Florian Haselbeck and Dominik G. Grimm analyzed and interpreted the results. Florian Haselbeck wrote the original draft of the manuscript, which was reviewed and edited by Dominik G. Grimm.

3.2.1 Summary

Changing data distributions over time is a well-known challenge in time series forecasting, which can lead to incorrect predictions using an outdated forecasting model and subsequent damage, e.g., financial loss. In Section 1.2, this issue is outlined in more detail. To ensure an up-to-date forecasting model that delivers accurate predictions at all times, we developed **Event-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data** (EVARS-GPR). This novel online forecasting algorithm addresses shifts in the target variable scale of seasonal data. In contrast to many existing approaches, EVARS-GPR is event-triggered and thus computationally efficient while ensuring an immediate reaction after a detected change point. Moreover, EVARS-GPR does not require *a priori* assumptions about potential changes of the system behavior, and augments existing data to be able to reuse it. Our algorithm is focused on seasonal data, a characteristic that is true for many time series when considering seasonality as a periodic system behavior. As the prediction model, we employ GPR (Williams & Rasmussen, 1995), see Section 2.3.2, which showed a good performance in our previous publication outlined in Section 3.1 and delivers prediction uncertainties that might be useful for downstream applications.

For EVARS-GPR, we distinguish between the offline phase during which the initial prediction model is built and the online phase, i.e., the productive operation of a forecasting system.

In Figure 3.2, we give an overview of EVARS-GPR to provide an intuitive understanding. We refer to the original publication in Appendix B for mathematical and algorithmic details.

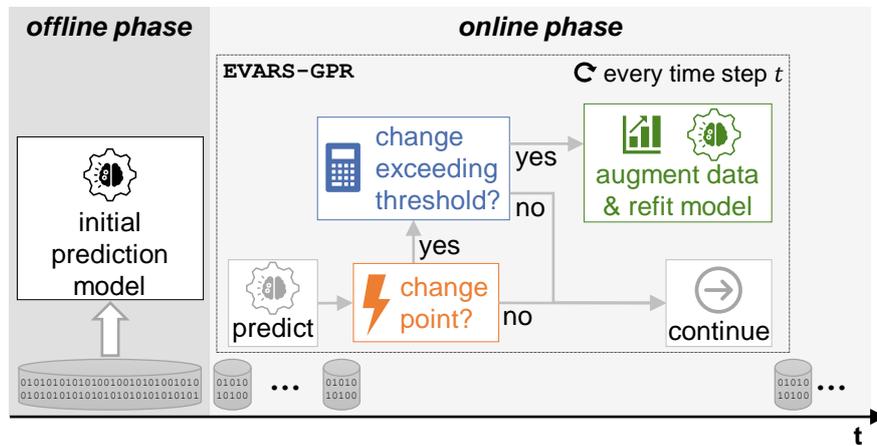


Figure 3.2: **Overview of EVARS-GPR:** This figure is similar to Figure 1 of the original publication and is shown for better readability (Haselbeck & Grimm, 2021). During the offline phase, the initial prediction model is determined. With every new sample available during the online phase, the next target value is first predicted before an online change point detection algorithm runs. If a change point is detected, the algorithm checks whether the extent of the output scale change compared to previous seasons exceeds a threshold. In case this condition is also fulfilled, the current prediction model is refitted using augmented existing data. EVARS-GPR continues with the current prediction model if one of these conditions is not met.

During the offline phase, EVARS-GPR is not different from the usual approach to determine the prediction model, for instance, in a cross-validation setup to optimize hyperparameters. Since EVARS-GPR is an online algorithm, its main procedure starts in the online phase with continuously available new samples. First, EVARS-GPR forecasts the next value of the target variable using its current prediction model. After this step, we run an online CPD method to check if a change in the data distribution has occurred. If this is true, we determine the extent of the output scale change by setting the target values within a window of the current season in relation to previous seasons. This output scaling factor is then compared to a threshold value to trigger the augmented refitting if it is exceeded. Online CPD approaches are prone to false alarms, e.g., due to outliers. With this additional check and smoothing of values within a window, we intend to prevent unjustified model refittings. In the augmented refitting step, we reuse the data prior to the change point and

augment it using the information we gained on the output scale change. For computational efficiency and to focus more on recent samples, we limit the data we use to a certain number of previous seasons. Finally, we refit the current prediction model using this augmented dataset. If we do not detect a change point or the output scaling factor does not exceed the threshold, EVARS-GPR continues with its current prediction model. This procedure is repeated for every time step t with a new sample becoming available.

To select the CPD and data augmentation methods and their as well as EVARS-GPR's parameters, such as the number of previous seasons and the window size to consider for the the output scaling factor, we used synthetic data. In total, we simulated 67 scenarios that differed, for instance, in terms of the extent and speed of the output scale change, and applied a Random Search for hyperparameter optimization, see Section 2.4. Based on this, we have finally selected ChangeFinder (Takeuchi & Yamanishi, 2006) and not BOCPD (Adams & MacKay, 2007) for CPD, see Section 2.5, and a simple scaling of the original dataset for augmentation. For more details on the data simulation and the selection of the algorithm components and parameters, we refer to the original publication.

We further used the synthetic data to analyze the behavior of EVARS-GPR under pre-defined conditions, see Section 6.1 of the original publication. In summary, EVARS-GPR showed broad applicability for various output scale changes and consistently performed at least as well as a configuration without model updates. Beyond that, we evaluated EVARS-GPR on ten real-world datasets, separating the last 20 % to simulate an online scenario. EVARS-GPR was robust without any incorrect model adjustments for five datasets that did not show an output scale change. For the other five datasets, EVARS-GPR outperformed all comparison partners with a comparable computational resource consumption in terms of RMSE, on average by at least 20.8 %. Not surprisingly, EVARS-GPR yielded a higher RMSE than computationally exhaustive methods employing periodical refittings. However, for three of the five datasets, EVARS-GPR delivered comparable results and, in addition, showed a six-fold runtime reduction averaged over all datasets in relation to the approach with the second-lowest runtimes.

3.2.2 Further Unpublished Results

We further extended EVARS-GPR to EVARS-GPR+, which is unpublished and shown in Algorithm 4. In contrast to EVARS-GPR, its extension aims to also account for less significant changes of the output scale, or even data distribution changes that are unrelated to the output scale. Since EVARS-GPR's augmented refitting showed a good performance

for datasets containing an output scale change and demonstrated broad applicability on synthetic data, we kept the augmented adjustment scheme. In addition to the augmented refitting, we include a second retraining mechanism if a change point is detected, but the output scale change criterion is not met, see line 14 of Algorithm 4. For this purpose, we complement the dataset of the last augmented refitting with the unchanged samples since then to perform a non-augmented refitting of the current prediction model.

Algorithm 4 EVARS-GPR+ including a non-augmented refitting in contrast to EVARS-GPR

Require: initial prediction model M_{base} ; offline data $\mathcal{D}_{off} = \{(x_t, y_t) \mid t = 1, \dots, t_{off}\}$; parameters for output scale change criterion: number of previous seasons $n_\eta \in \mathbb{N}$, window size $n_w \in \mathbb{N}$, seasonal length $n_{seas} \in \mathbb{N}$, threshold $\pi_\eta \in \mathbb{R}_{\geq 0}$; parameters for CPD method ChangeFinder θ_{CF} .

```

1:  $M_{current} \leftarrow M_{base}$ 
2:  $\eta_{old} \leftarrow 1$ 
3:  $\mathcal{D}' \leftarrow \{\}$ 
4:  $t_{last} \leftarrow t_{off}$ 
5: for  $t = t_{off}, \dots, t_{end}$  do
6:   predict next value:  $\hat{y}_{t+1} \leftarrow \text{predict}(M_{current}, x_t)$ 
7:   run change point detection:  $cp \leftarrow \text{ChangeFinder}(\mathcal{D}_{i=1}^t, \theta_{CF})$   $\triangleright$  see Sec. 2.5
8:   if  $cp$  then
9:     get output scaling factor:  $\eta \leftarrow \text{calc\_eta}(\mathcal{D}_{i=1}^t, n_\eta, n_{seas}, n_w)$   $\triangleright$  see Eq.(1) App. B
10:    if  $|\eta - \eta_{old}| / \eta_{old} > \pi_\eta$  then
11:       $M_{current}, \mathcal{D}' \leftarrow \text{augmented\_refitting}(M_{current}, \eta, \mathcal{D}_{i=1}^t)$ 
12:       $t_{last} \leftarrow t$ 
13:       $\eta_{old} \leftarrow \eta$ 
14:    else
15:       $M_{current} \leftarrow \text{non\_augmented\_refitting}(M_{current}, \mathcal{D}_{i=t_{last}}^t, \mathcal{D}')$ 

```

To evaluate the performance of EVARS-GPR+, we also used the ten real-world datasets from our original publication to simulate an online scenario with the last 20% of the data. For our comparison, we included M_{base} , which does not refit the prediction model at all, and the computationally exhaustive approaches PR1 and PR2, which perform a refitting in every respective every second time step. Moving-Window GPR (MWGPR) combines a refitting in every time step with discarding the oldest samples and a recursive bias term to correct the model based on the previous performance (Ni et al., 2012), see Section 1.3.2. In our original publication, we have already shown that EVARS-GPR outperformed algorithms that do not augment at all (CPD_retrain and CPD_MW), which is why we did not include these. Table 3.1 gives an overview of the results. Compared to EVARS-GPR,

we observe that EVARS-GPR+ leads to equal or better results in nine out of ten cases. We only see a slight RMSE increase of 1.0% for *AirPassengers*. For five datasets, we obtain better results, with an improvement of up to 59.8%. In summary, the second refitting mechanism seems to be beneficial in most cases, making EVARS-GPR+ a straightforward yet reasonable extension.

Table 3.1: **Overview of results in terms of RMSE on real-world data:** The unpublished extension EVARS-GPR+ is compared to EVARS-GPR, M_{base} without any refitting during the online phase, and the computationally exhaustive approaches PR1, PR2, and MWGPR. PR1 and PR2 refit the prediction model in every and every second time step, respectively. Moving-Window GPR (MWGPR) performs a model refitting every time step, discarding the oldest sample and a recursive bias term for model correction (Ni et al., 2012). We show results for the real-world datasets used in the EVARS-GPR publication, half of which does not contain an output scale change. The best result for each dataset is highlighted in bold. For EVARS-GPR+, we show the percentage change in RMSE compared to EVARS-GPR. Dataset sources: (Haselbeck et al., 2022): *CashierData*; (Hyndman & Athanasopoulos, 2021): *DrugSales*, *VisitorNights*, and *Beer*; (Box et al., 2016): *AirPassengers*; (Earth System Research Laboratory, 2021): *MaunaLoa*; (Makridakis & Wheelwright, 1989): *ChampagneSales*; (Chakrabarty, 2021): *TouristsIndia*; (Makridakis et al., 1998): *Milk*, and *USDeaths*.

dataset	M_{base}	PR1	PR2	MWGPR	EVARS-GPR	EVARS-GPR+
datasets with an output scale change						
<i>CashierData</i>	1351.43	1119.23	1185.82	1098.16	1125.34	1094.77 (↓2.7%)
<i>DrugSales</i>	6.15	2.44	2.54	2.86	2.75	2.75 (→)
<i>AirPassengers</i>	171.58	69.06	74.28	72.27	93.88	94.80 (↑1.0%)
<i>MaunaLoa</i>	34.37	11.12	12.60	9.88	27.96	11.25 (↓59.8%)
<i>VisitorNights</i>	10.97	5.08	5.21	5.85	5.11	5.11 (→)
datasets without an output scale change						
<i>ChampagneSales</i>	1158.26	1096.10	1098.79	1369.95	1158.26	1158.26 (→)
<i>TouristsIndia</i>	90707.30	92083.88	73924.96	190762.90	90707.30	90707.30 (→)
<i>Milk</i>	15.16	10.69	11.09	8.31	15.16	12.01 (↓20.8%)
<i>Beer</i>	16.88	14.54	14.53	18.82	16.88	14.98 (↓11.3%)
<i>USDeaths</i>	276.72	261.07	263.98	253.81	276.72	260.80 (↓5.8%)

3.2.3 Key Findings

EVARS-GPR demonstrated its computational efficiency, its broad applicability to various output scale changes, and its ability to achieve results that are even comparable to computationally exhaustive approaches. Regarding the results of the unpublished extension EVARS-GPR+, we observed that a second refitting mechanism without data augmentation was beneficial. Hence, we developed two computationally efficient online forecasting algorithms that can handle sudden changes in the target variable scale of seasonal data. As a further plus, both EVARS-GPR and EVARS-GPR+ are model-agnostic, allowing for the integration of a different prediction model, which may be beneficial for some forecasting tasks. However, we also observed that for some datasets, e.g., *Milk* and *AirPassengers*, both EVARS-GPR and its extension were significantly outperformed in terms of RMSE. A potential reason is that the parameterization is based on synthetic data, which might not lead to the best configuration for every real-world dataset. Furthermore, we keep the hyperparameters, particularly the kernel expression of the GPR model, fixed for both adjustment mechanisms. Thus, an accurate adaptation to larger system behavior changes might be difficult using our approaches.

3.3 Dynamically Self-Adjusting Gaussian Processes

The following publication, for which the original full version can be found in Appendix C, is summarized in this section:

PUBLICATION C:

Jan D. Hüwel*, **Florian Haselbeck***, Dominik G. Grimm and Christian Beecks (2022). Dynamically Self-adjusting Gaussian Processes for Data Stream Modelling. *KI 2022: Advances in Artificial Intelligence*, 13404, 96-114. https://doi.org/10.1007/978-3-031-15791-2_10

**Both authors contributed equally and share first authorship.*

The individual contributions are as follows: Florian Haselbeck and Jan D. Hüwel developed the presented method. Jan D. Hüwel implemented the code with contributions from Florian Haselbeck. Florian Haselbeck prepared the simulated and the real-world data. Jan D. Hüwel performed the experiments with help of Florian Haselbeck. Florian Haselbeck and Jan D. Hüwel analyzed and interpreted the results with contributions from Dominik G. Grimm and Christian Beecks. Florian Haselbeck and Jan D. Hüwel prepared the manuscript draft, which was reviewed and edited by Dominik G. Grimm and Christian Beecks.

3.3.1 Summary

With our previous publication outlined in Section 3.2, we primarily address changes in the data distribution in seasonal time series that are reflected in a change of the output scale. However, as we do not change the kernel expression of the GPR prediction model, other changes, i.a., a shift in the periodicity, may be challenging. To provide useful predictions using an up-to-date model at all times, we presented dynamically self-adjusting GPs with our novel algorithm **Event-Triggered Kernel Adjustments in Gaussian Process modeling (ETKA)**, which we subsequently summarize. For more details, we refer to our original publication in Appendix C. ETKA combines a novel GP-based CPD approach with AKS (Hüwel et al., 2021), see Section 2.3.2, during the online phase. Previous AKS applications perform a periodical model reconfiguration. We replace this procedure by an event-triggered mechanism employing a CPD, to prevent on the one hand extended periods with an outdated model and on the other hand unnecessary and computationally expensive kernel searches.

Similar to the common approach, we first use CKS (Duvenaud et al., 2013), see Section 2.3.2, during the offline phase to determine the best-fitting kernel expression for the GPR model (Williams & Rasmussen, 1995). Then, with each new sample becoming available, we predict the next target value before running our novel and efficient CPD approach, which is CUSUM-based (Page, 1954). For this CPD method, we accumulate the absolute prediction error $|y_t - \hat{y}_t|$ using the true value y_t and the predicted value \hat{y}_t to determine a change point score $s \in \mathbb{R}_{\geq 0}$:

$$s = \max(0, s + |y_t - \hat{y}_t| - 2 \cdot \delta \cdot \sigma) \quad (3.1)$$

We further use the square root of the noise σ^2 of the GP model and a tolerance factor $\delta \in \mathbb{R}$ to not count predictions that are within the inner $\delta \cdot 100\%$ of the prediction model's confidence interval. In case this change point score s exceeds a threshold $\epsilon_{ETKA} \in \mathbb{R}_{>0}$, we detect a change point and reset s to zero. With this computationally efficient approach, we make use of the GP's uncertainty. The further a predicted value is outside of the tolerance interval mentioned above, the more it counts towards declaring a change point. If a change point is detected, we use AKS to adjust the GPR prediction model using the most recent data points within a window of size n_w .

We again used simulated data to determine the algorithm's parameters δ and ϵ_{ETKA} and to analyze its behavior under controlled conditions. However, the simulation setup differs

from our previous publication. We also considered other types of data distribution changes in addition to output scale shifts, i.a., a changing period length. For evaluation, we used the MAE during the online phase. We further included AKS employing periodical refitting scheme (PER AKS) as well as a change point-triggered adjustment of the GP without a kernel search (CPD HPO) as comparison partners.

With respect to the results averaged across all simulated scenarios, we observed that ETKA outperforms both comparison partners in terms of the prediction error and showed the best performance for a rather sensitive CPD with $\delta = 0.5$ and $\epsilon_{ETKA} = 5.0$. Since providing an up-to-date model at all times is the primary objective of ETKA, we decided to use this configuration at the cost of a longer runtime, which, not surprisingly, decreases for a less sensitive CPD. Using this parameterization, we further tested ETKA on 14 real-world datasets from different domains showing varying characteristics, such as the number of samples. Averaged across all 14 datasets, ETKA outperformed PER AKS and CPD HPO regarding the prediction error by 2.73 % and 6.19 %, respectively. However, at the cost of longer runtimes for several datasets. Interestingly, when focusing on the datasets for which the runtime advantage of PER AKS is bigger than 100 %, we observed a lower prediction error of ETKA for three out of four cases, with a decrease of up to 19.30 %.

3.3.2 Key Findings

Analyzing ETKA's behavior using simulated data, we again identified broad applicability despite different change point patterns, with ETKA mostly outperforming its comparison partners. We also observed a lower prediction error on most real-world datasets, at the cost of longer runtimes, since we have chosen a rather sensitive CPD parameterization. This parameterization could be changed for applications focused on fast processing, to trade a potential decrease in prediction performance for lower runtimes. Furthermore, the number of samples and AKS iterations for the model configuration could be adjusted. Thus, in addition to EVARS-GPR and EVARS-GPR+, which focus on output scale changes in seasonal data, we presented a flexible online method that handles various change point patterns in seasonal and non-seasonal data and automatically provides an up-to-date model at all times.

3.4 A Comprehensive Time Series Forecasting Framework

The following publication, for which the original full version can be found in Appendix D, is summarized in this section:

PUBLICATION D:

Josef Eiglsperger*, **Florian Haselbeck*** and Dominik G. Grimm (2023). ForeTiS: A comprehensive time series forecasting framework in Python. *Machine Learning with Applications*, 12, 100467. <https://doi.org/10.1016/j.mlwa.2023.100467>

*Both authors contributed equally and share first authorship.

The individual contributions of the authors are as follows: Florian Haselbeck and Josef Eiglsperger designed the software framework with contributions from Dominik G. Grimm. Florian Haselbeck and Josef Eiglsperger implemented the code and created the online documentation. Josef Eiglsperger prepared the data and conducted the experiments for the case studies of the publication. Josef Eiglsperger and Florian Haselbeck prepared the original draft of the manuscript, which was reviewed and edited by Dominik G. Grimm.

3.4.1 Summary

A key finding of our first comparative study on time series forecasting for horticultural sales prediction and a general issue as outlined in Section 1.2 is the need for a comprehensive and state-of-the-art framework. This facilitates to conduct such studies while ensuring reproducible and comparable results as well as accessibility of novel methods. Therefore, we have developed ForeTiS, a time series forecasting framework in Python that covers the whole pipeline from data pre-processing over feature engineering to hyperparameter optimization and model selection using state-of-the-art approaches. In addition, we support novice users with a comprehensive online documentation, including various hands-on tutorials and video guides, at <https://foretis.readthedocs.io/>.

The main modules of ForeTiS are structured according to the common time series forecasting workflow, see Figure 3.3. The largest benefit for end users is the possibility to perform state-of-the-art time series forecasting with a single line of code. Users simply need to provide data, for instance, as a CSV file, and a dataset-specific configuration file. This implementation allows for straightforward yet customizable usage. The data pre-processing, feature engineering, and hyperparameter search are fully automated. Regarding data pre-processing, we integrate three missing value imputation methods and PCA for dimensionality reduction. We offer to include calendrical, for instance, date-based information, e.g., the

day of the week, and statistical features, i.a., statistics over past sales. For seasonal time series, a characteristic that many datasets show, we allow deriving features based on previous seasons, which is likely to be predictive.

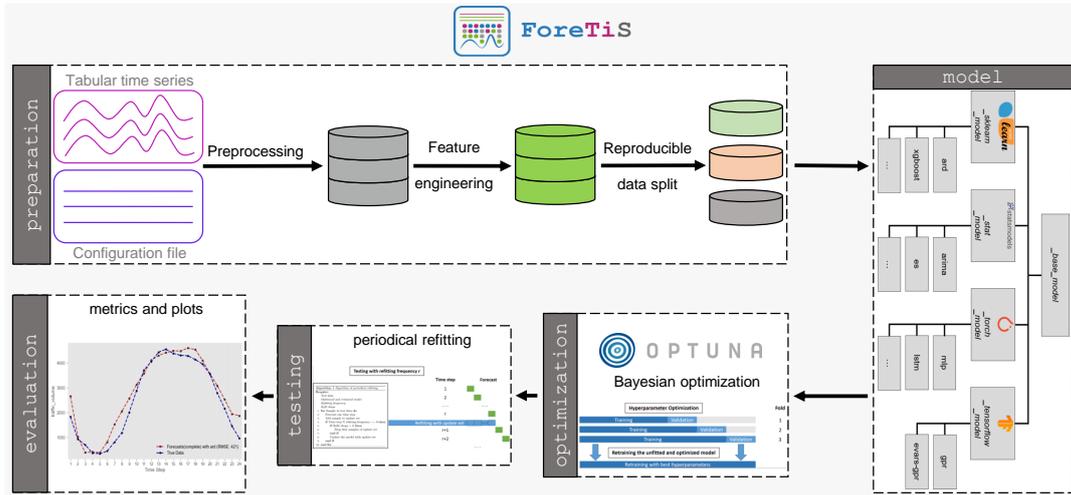


Figure 3.3: **Overview of the main components of ForeTiS:** This figure is similar to Figure 1 of the original publication and is shown for better readability (Eiglsperger et al., 2023). The submodule preparation contains the data preprocessing and feature engineering methods. We designed model to allow for easy integration of further prediction models and already included several ones. For hyperparameter search, we leverage Bayesian optimization via the Python package Optuna. Our testing module enables to test different refitting schemes, of which the results can be analyzed using evaluation.

Furthermore, the user can freely choose from a variety of already integrated prediction models ranging from classical time series forecasting approaches over ML-based methods to deep learning (DL)-based architectures. We consider both frequentist and probabilistic approaches, i.a., GPR or Bayesian neural networks. Regarding data splits, see Section 2.4, ForeTiS includes a regular and a time-series cross-validation, as well as a split into training, validation and test data maintaining the chronological order. The subsequent hyperparameter search is completely automated using Bayesian optimization via the Python package Optuna (Akiba et al., 2019; Bergstra et al., 2011). We further include procedures to test different refitting approaches, for instance, periodical schemes or CPD-based EVARS-GPR (Haselbeck & Grimm, 2021). For evaluation, we provide common metrics and simple baselines, e.g., Random Walk. We further support results analysis with functions that summarize optimization outcomes and generate plots showing predicted and true values.

For advanced users who aim to develop novel methods, `ForeTiS` and, in particular, its submodule `model` are designed to allow for quick integration and benchmarking of new prediction approaches. In a base class, we define methods that are useful for all prediction models, as well as methods and attributes that need to be implemented by each child class. For the common Python frameworks `scikit-learn` (Pedregosa et al., 2011), `statsmodels` (Seabold & Perktold, 2010), `PyTorch` (Paszke et al., 2019) and `TensorFlow` (Abadi et al., 2015), we already implement some of these mandatory methods, for instance, the training loops. Thus, users only need to implement two methods defining the prediction model and the hyperparameters with their ranges. This concept allows advanced users to focus on the design of the forecasting method while relying on a thoroughly-tested and fully-automated remaining pipeline. Furthermore, we support quick and fair benchmarking as we have already integrated various forecasting approaches.

3.4.2 Key Findings

In summary, we provide a powerful resource for both end users and forecasting experts that ensures highly reproducible and comparable results. `ForeTiS` is easy to use, its main pipeline is fully automated yet customizable, and it is easily extendable. Users can conduct comparative studies to evaluate, e.g., which prediction model, featureset and refitting scheme works best for their application. A further plus is the possibility to quickly integrate and benchmark new forecasting approaches, increasing the development speed and lowering the risk of errors by using a reliable framework. Moreover, we ensure user support through a comprehensive online documentation, including various hands-on guides and video tutorials.

Discussion

After outlining the four main contributions of this thesis, we subsequently discuss the findings. The discussion is structured with respect to the objectives of this thesis given in Section 1.4. Since the main contributions of this work addressing these objectives also include discussions, some points outlined below are inspired by (Haselbeck et al., 2022), (Haselbeck & Grimm, 2021), (Hüwel et al., 2022), and (Eiglsperger et al., 2023). Besides aspects related to the objectives, we present further general points of discussion, also with respect to current challenges in time series forecasting outlined in Section 1.2 but not addressed in this thesis.

4.1 Time Series Forecasting for Small and Medium-Sized Companies

One objective of this thesis is to assess the potential of time series forecasting methods despite changing data distributions for sales prediction of small and medium-sized companies dealing with perishable goods based on the example of horticulture. In a first-time comparative study in this domain, we used five horticultural retail sales datasets, i.e., the trade level that is closest to the end customer and thus probably more directly influenced by external factors such as weather. We further enriched these datasets with external features, e.g., weather and calendric information, and employed three classical as well as nine ML-based forecasting methods. More details regarding the experimental setup and results can be found in Section 3.1 and Appendix A. Most of the employed methods outperformed simple baselines, see Section 2.4. Manual demand predictions by an expert, the most common current procedure in horticulture, were not available for comparison.

Indicators for manual predictions are past sales in the current and previous seasons. The simple baselines we included in our comparison, e.g., a random walk and a seasonal random walk, use the same information. Thus, these approaches, which were mostly outperformed, can be considered to be similar to manual predictions, indicating that time series forecasting is beneficial for predicting horticultural demand. XGBoost, an ensemble learner leveraging gradient boosting, performed best in most comparisons. This observation is consistent with scientific publications in other application domains, for which ensembling, particularly gradient boosting methods, has shown a good performance (Bojer & Meldgaard, 2021; Makridakis et al., 2022). Furthermore, LSTM and GPR performed comparably well. The competitive performance of ensemble learners and LSTM networks was also observed for domains with similar characteristics to horticulture, namely food and tourism, which is described in Section 1.3.1. In accordance with findings for these domains, we identified a superiority of ML-based approaches, especially for larger datasets (Huber, 2019; Jiao & Chen, 2019; Priyadarshi et al., 2019; Turgut & Erdem, 2022). Besides the amount of available data, a potential reason for this improved performance is that multivariate patterns and nonlinear relationships may be more prominent in these larger datasets, and that ML-based methods have the ability to capture these. However, the top performer XGBoost also showed the lowest error for most of the smaller datasets. As shown in our publication, a further plus of this ML-based ensemble learner is its comparably low runtime. With respect to changing data distributions and a potentially required model refitting, this advantage becomes even more important.

We further observed a significant increase in sales in 2020 for one of our datasets, i.e., a change in the data distribution, probably due to the SARS-CoV-2 pandemic. Regarding the results on the relative measures sMAPE and MAPE, we obtained higher values than for the other datasets, indicating that such a shift in the scale of the target variable is challenging. All prediction models were limited in their ability to capture this change, even though we performed regular model refittings that were probably beneficial. However, the amount of data resulting from the changed data distribution was comparatively low in relation to the whole refitting data, potentially influencing the prediction model only to a limited extent. For this dataset, XGBoost was also the top performer. Surprisingly, the univariate classical approach SARIMA performed well. The features used for the multivariate methods could even have impeded the forecasting quality, since the data distribution changed and the prediction model was trained on an outdated relationship between the features and the target variable. This observation is not true for the other datasets, for which we did not detect a changing data distribution, with the top performer always being a multivariate

approach. Thus, in general, external factors are probably beneficial for horticultural sales prediction, in particular for the ML-based methods that have proven their strength when these features are available. Unfortunately, we could not determine an overall superior set of features, but rather a need to re-evaluate different ones for each task.

We made some simplifications for this first-time comparative study. As explained in Section 1.2, sales numbers only approximate the actual demand, particularly in out-of-stock situations. Furthermore, stock figures were not recorded, which could be used to better estimate the real demand. Hence, we had to approximate the real demand using sales figures. Moreover, we enriched our dataset with weather information. For this purpose, we leveraged historical weather observations. However, for a real-world forecasting system, only short-term weather forecasts with a lead time of about two weeks would be available, which also contain uncertainty. With the test data, we simulated the live operation of a forecasting system with a continuous data stream. We first predicted the next day's or week's sales, depending on the dataset. Hence, using historical weather data seems to be a reasonable simplification, as the weather forecasts for the next day or week are likely to be less uncertain than long-term predictions. However, due to the lack of historical weather forecast records, we also had to use historical weather data to optimize the prediction models that we used to simulate the live operation. For this hyperparameter optimization, we used longer forecasting horizons, for which weather forecasts would not be available in practice or contain uncertainty. This simplification is common when using weather data, but should be evaluated in more detail for long-term forecasts. For instance, it would be possible to create separate models for different forecasting horizons that use different external information, neglecting weather-related features for long-term prediction models. Beyond that, we only considered calendar and weather data as external information, but other features, such as promotional or communication activities of a retailer, could be predictive. However, these are often not recorded in computer systems and must be collected manually, leading to costs and potentially inaccurate data.

This study is the first proof of concept to predict horticultural demand with time series forecasting approaches based on retail data. We provide some insights and show the potential of such an approach. However, a broader study using data from various companies across the whole value chain and products with different characteristics is essential to draw more general conclusions. Besides providing guidance for such a broader study with our findings, we can derive several points that are interesting for future research. With XGBoost, an ensemble learner lead to the best result, which is in accordance with the literature.

Therefore, it is interesting for future research to evaluate other combined approaches, possibly using different types of methods. For instance, the winning method of the M4 forecasting competition combined Exponential Smoothing with a recurrent neural network, i.e., a classical forecasting and a DL-based approach (Smyl, 2020). We further chose a rather computationally exhaustive approach with periodical model refittings, however with limited success for a dataset with a changing distribution. Thus, our results demonstrate the necessity of approaches that specifically account for changing data distributions. We address this issue, as outlined, in two further publications, see Sections 3.2 and 3.3, and discuss the findings in the next section of this chapter. Beyond that, we employed retail sales data for our study, the level of trade potentially most directly affected by external features such as weather data. However, data from the whole value chain, e.g., from wholesalers or producers cooperating with a retailer, could be predictive. Furthermore, retail sales data could also be interesting for demand predictions at these upstream parts of the value chain.

A challenge for time series forecasting for small and medium-sized companies is comparably small datasets. Beyond that, horticultural companies often have a very wide range of products with various different characteristics, i.a., seasonality or peak sales periods. In addition, some articles have rather short seasons, resulting in sparse and intermittent datasets. A potential way to address this issue in future research is to combine data from different companies and thus benefit from the unified information based on several sources. However, this is challenging due to the diversity of the products and their quantity, as well as the heterogeneity of many horticultural companies, e.g., specializing on different plants and services. To enable a prediction model based on such a combined dataset, groups of companies and products with similar characteristics must be defined, which would be facilitated in a larger consortium. Predicting individual items, for instance, cut roses, makes it difficult to combine datasets and would require the development of many prediction models. Thus, meta-products that summarize several articles could be defined to allow for the development of a more general model. Due to the large number of products, automated approaches are needed to define the product groups. A grouping based on the biological taxonomy might not be useful. For instance, the family *Rosaceae* includes ornamental plants such as roses, as well as fruit trees such as apples, with completely different purposes of use and sales characteristics (Erhardt et al., 2014). With respect to demand prediction, the grouping should rather be guided by similar sales characteristics, which may require other data-driven approaches. Demand predictions for such meta-products could support an expert's operational decisions with respect to specific items that are part of the product

group. Generative models are a further option for enriching small datasets, which is outlined in more detail in Section 4.4 together with further points of discussion related to synthetic data.

As discussed, several challenges, such as heterogeneous data and company structures, must be solved with technological measures. Overcoming these issues is essential because creating and maintaining many prediction models would be expensive and error-prone. However, there are further issues impeding the practical operation of a forecasting system for horticultural demand. Most companies are small or medium-sized and independent. Horticultural retailers, for instance, often operate only one store (Bundesministerium für Ernährung und Landwirtschaft, 2021). This company size limits their potential financial benefits when using accurate demand predictions and thus probably the amount of money they are willing to pay for such a service. As a result, the ratio between the cost of integrating a company and the revenue that a forecasting system operator can generate with that business may be economically unattractive. The cost-income ratio may be better for other domains dealing with perishable goods, e.g., bakeries, which often operate multiple stores. Furthermore, several suppliers for merchandise management systems exist, often with additional customization for certain companies. This heterogeneity of the information technology infrastructure can additionally increase the efforts of a forecasting system operator. Collaborations with cooperatives representing the interests of multiple member companies could be an option to standardize interfaces and sales documentation as an enabler for a forecasting system. For a live operation, a standardized interface to a merchandise management system would be essential to enable an automated access to the latest data. Thus, integrating a forecasting system into a merchandise management system could be beneficial. Besides simplifying interfaces, users would not need to use different tools. Orders could be even placed fully automatically or by an expert supported by accurate predictions. To better estimate the real demand, stock recordings could be helpful. However, stock recordings are expensive and not common due to the wide range of products, which are often fast-moving, and high manual efforts for additional quality assessments. Thus, the simplification of using sales numbers as an approximation of the demand must probably still be made. These issues impeding horticultural sales predictions can provide guidance for other domains with small and medium-sized companies dealing with perishable goods when aiming to offer a forecasting service.

4.2 Novel Approaches for Changing Data Distributions

A further objective of this thesis is to develop novel and computationally efficient online methods that quickly adapt to changing data distributions. Such distributional shifts are a common problem in time series forecasting that we also observed in our first-time comparative study. As already described, predictions from a model trained on an outdated data distribution could be useless and even cause damage, e.g., financial loss when misestimating demand. Computational efficiency and quick adaptation are important for the integration of an algorithm into a live-operated forecasting system and potentially needed model adjustments. However, the primary objective is often to get accurate predictions at all times.

To address changes visible in the output scale, see Objective 2A outlined in Section 1.4, we first developed EVARS-GPR. EVARS-GPR focuses on this type of change pattern and is limited to seasonal data, a property that many time series show. We demonstrated the broad applicability and good performance of EVARS-GPR on both simulated and real-world data. To assess the limitations and strengths of the algorithm, we used synthetic data with different times of occurrence, durations, speeds, and extents of the change in the output scale. In all scenarios, EVARS-GPR outperformed a non-refitting scheme. As expected, the benefit of using EVARS-GPR was bigger for longer lasting and larger output scale changes. Shorter and less pronounced changes were rather hard to detect for the integrated CPD method, reducing the advantage of EVARS-GPR. Regarding the speed of the output scale change, i.e., the abruptness of the change, we observed a broad applicability with minor differences. However, lower speeds were harder to detect, which lead to a later adjustment of the prediction model. EVARS-GPR further proved its usefulness on real-world data, outperforming comparison partners with a similar resource consumption on average by 20.8 % with respect to RMSE compared to the second-best competitor. In a further benchmark against computationally exhaustive periodical refitting schemes, EVARS-GPR performed, as expected, worse in terms of RMSE. Nevertheless, our algorithm delivered a comparable performance for three out of five datasets. The other two datasets contained rather small output scale changes, so probably not all change points may have been detected by the CPD integrated into EVARS-GPR. Furthermore, these datasets may contain changing data distributions that are not visible in the output scale, e.g., a change in the periodicity. A major advantage of EVARS-GPR compared to the periodical refitting schemes is its computational efficiency. We observed a more than six-fold lower averaged runtime in relation to the second-most efficient approach,

which triggers a refitting at every second time step. For *MaunaLoa* and *AirPassengers*, the two datasets for which EVARS-GPR was outperformed by the largest margin regarding RMSE, the runtime reduction was even 250 and 16 times, respectively. This high runtime reduction in conjunction with the higher RMSE also indicates that probably not all change points have been detected to trigger the augmented refitting. The parameterization of a CPD method is usually a trade-off between unjustified alarms and not detecting actual changes. As shown on datasets that do not contain an output scale change, EVARS-GPR was robust against false alarms and did not lead to augmented refittings in these cases. However, the parameterization of the CPD could also be a reason for the worse results compared to computationally exhaustive approaches for datasets with smaller changes in the output scale.

Hence, accurately detecting change points during the live operation is essential for EVARS-GPR. To prevent false alarms, for instance, in case of outliers, we additionally introduced a threshold for the change of the output scale in comparison with previous seasons. Improvements regarding the online CPD that is part of our algorithm could probably lead to a better downstream performance regarding the prediction error. As described above, the parameterization of the CPD method within EVARS-GPR strongly influences the downstream performance. We employed simulated data to parameterize the whole pipeline, taking into account a diverse set of potential output scale changes. Although this approach is reasonable, the parameter values may not be suitable for all datasets. Hence, besides the above-described enhancement of the CPD method itself, improvements with respect to the hyperparameter optimization of EVARS-GPR are interesting for future research. For instance, simulating changing distributions of real-world data could be integrated into the cross-validation that is performed to determine the initial prediction model. In this way, dataset-specific parameters could be obtained, potentially accounting for the dataset's characteristics. However, such an approach would require additional computational resources during the offline phase of EVARS-GPR. Moreover, EVARS-GPR has proven its computational efficiency, but periodical refitting schemes have shown lower prediction errors. Thus, a combination of both could be beneficial, with low frequently triggered model refittings in EVARS-GPR in conjunction with the current procedure of CPD-based and augmented adjustments. This combination could prevent extended periods using an outdated prediction model if a change point was not detected while still ensuring computational efficiency. A further plus would be that refittings would not only be performed in case of a detected output scale change, so other types of changing data distributions could also be addressed. Beyond that, we selected GPR as the prediction model due to its

good performance in our first comparative study and the inherent prediction uncertainties. Nevertheless, for some datasets, other prediction models could be of advantage. EVARS-GPR is model-agnostic, and thus integrating other forecasting approaches seems interesting for future research. In a Master's thesis, we conducted initial experiments to consider a dataset-specific parameterization and the integration of further prediction models. However, these preliminary results showed limited success (Golubovic, 2022). Further research is needed to analyze these challenges in more detail.

EVARS-GPR+, which we integrated as unpublished results in this thesis, is an extension of EVARS-GPR that additionally includes a non-augmented refitting in case a change point is detected, but the threshold for output scale change in comparison with previous seasons is not exceeded. We assessed EVARS-GPR+ using the parameterization of EVARS-GPR on ten real-world datasets. For the five datasets that contain an output scale change, we observed a worse performance for only one dataset, no change in RMSE for two, and an improvement by 2.7% and 59.8%, respectively. The largest improvement was observed for *MaunaLoa*, for which EVARS-GPR was outperformed by its computationally exhaustive comparison partners by a large margin. However, EVARS-GPR+ even achieved a comparable performance. Interestingly, EVARS-GPR+ also led to an averaged improvement of 12.6% for three of the five datasets without an output scale change, while the results for the other two datasets remained unchanged. In summary, the additional model adjustments of EVARS-GPR+ were beneficial. In EVARS-GPR+, the threshold for the change of the output scale compared to previous seasons is not used to prevent false alarms of the CPD, but to decide whether to perform an augmented refitting or a regular model adjustment. Hence, EVARS-GPR+ prevents to perform an unjustified dataset augmentation before the model adjustment, and also accounts for smaller output scale changes. This assumption is supported by the large improvement on *MaunaLoa*, which shows a rather small change in the output scale. Even though the parameterization of the integrated CPD was guided by a simulation of several output scale changes, other types of changing data distributions were potentially detected, since we achieved an improvement on datasets without output scale changes. Nevertheless, in case of other types of data distribution changes, model adjustment procedures that specifically account for these may still be more beneficial. EVARS-GPR+ is a small and straightforward enhancement of EVARS-GPR that led to better results. However, the points for future research described above for EVARS-GPR, e.g., improving the CPD method, enhancing the pipeline parameterization, or integrating additional prediction models, also apply to EVARS-GPR+.

EVARS-GPR and EVARS-GPR+ focus on changes in the output scale in seasonal data. However, other types of changing data distributions, such as a change in the period length, may require different model adjustments, for example a kernel search. To address this issue with respect to Objective 2B outlined in Section 1.4, we developed ETKA. ETKA also applies a change point-triggered model adjustment during the online phase, but employs AKS to find a new kernel. With this model adjustment, it may be possible to react appropriately to a more drastic change in system behavior. Similar to EVARS-GPR, we parameterized and assessed ETKA using simulated data consisting of diverse scenarios and different change point patterns. We observed that ETKA outperformed its comparison partners in most cases, also for scenarios with multiple change points within the same time series. Thus, we can assume that the CPD developed for ETKA works for various change point settings. However, we have also identified weaknesses of ETKA. In scenarios with slower changes in periodicity or output scale, the prediction errors were higher compared to more abrupt shifts. This effect is probably caused by a delayed reaction of the CUSUM-based CPD, for which the change point score increases more slowly for less pronounced changes. Consequently, this leads to longer periods with an outdated prediction model, but probably with less potential damage, since the predictions should also be more accurate than in scenarios with more abrupt changes. Based on the results on simulated data, we parameterized the CPD to be rather sensitive, focusing on an up-to-date model at all times and accepting potential disadvantages in terms of computational efficiency. ETKA also yielded the most accurate predictions on real-world data, but as expected, at the cost of higher runtimes. While the performance improvement of ETKA was rather stable, the runtime difference varied greatly depending on the number of change points detected and consequently triggered model adjustments. The largest benefit in terms of the prediction error was achieved for *Airline*, *Radio*, and *Airquality*. For *Airline*, a detailed analysis of the results showed that ETKA detected fewer change points but these more accurately than the comparison partners. With respect to the other two datasets, ETKA was advantageous in terms of the prediction error, but showed a higher runtime, which was caused by the multiple change points present and detected by ETKA. Thus, the higher runtimes are justified in these cases. Not surprisingly, the performance of ETKA depends on the success of the initial kernel search using CKS on offline-collected data. AKS starts its search depending on this initial kernel expression, leading to a less accurate model and consequently more model adjustments if this starting point is unsuitable. We observed such behavior for the dataset *Internet*, for which ETKA was outperformed both in terms of prediction error and runtime.

A periodical refitting scheme at constant intervals, as used for AKS in the original publication, is highly dependent on coincidence. For instance, a change point could occur immediately after a model adjustment, resulting in a prolonged period with an outdated prediction model and wasted computational resources due to an unnecessary model refitting. In contrast, ETKA employs a change point-triggered refitting scheme that does not depend on coincidence, which is potentially more accurate and efficient. Nevertheless, the performance of ETKA, similar to EVARS-GPR and EVARS-GPR+, is strongly influenced by the integrated CPD. The parameterization was again determined using a diverse set of simulated data. As outlined for EVARS-GPR, this may not lead to the best parameter set for all real-world datasets, although it is a reasonable approach. A dataset-specific parameterization could also be beneficial for ETKA, making this an interesting point for future research. Furthermore, we focused on an up-to-date model at all times, accepting potential drawbacks in terms of computational resources. For settings that require lower runtimes, ETKA offers several adjustment options that lead to higher computational efficiency but potentially result in a less accurate prediction model. For example, the number of samples and base kernels considered for the model adaptation could be reduced. Furthermore, the number of AKS iterations could be changed, and the parameters of the CPD could be adjusted to result in less sensitivity. Our experiments on simulated data show the trade-off between prediction error and runtime when setting the CPD parameters. Higher tolerance factors and thresholds lead to a lower runtime, but also to less accurate predictions. In addition, we set the window size of samples used for the model adjustments to be constant. However, this could also be determined dynamically, depending on the time elapsed since previous changes. Moreover, AKS within ETKA is efficient and applicable if the system behavior does not change completely. In such cases, even a new kernel search from scratch may be necessary to achieve an accurate prediction model.

With EVARS-GPR, its extension EVARS-GPR+, and ETKA, we have developed three approaches that address the important issue of changing data distributions in time series forecasting. All three methods work online without requiring *a priori* knowledge or assumptions about changes in the data distribution, neither about the time of occurrence nor the type. In contrast to Garnett et al., 2009 and Automatic Bayesian Covariance Discovery (ABCD) (Lloyd et al., 2014), we do not need to pre-define the time steps at which change points occur, and we allow for multiple changes in the data distribution. Moreover, some methods in the literature are designed for systems for which certain steady states can be defined, a limitation that does not apply to our approaches (Jin et al., 2015; Liu et al., 2015; Liu & Gao, 2015). Nevertheless, if it is possible to define such steady states, these

techniques could be a good solution, potentially providing an accurate model for each state based on data collected offline. In cases where this definition is not possible, our approaches may be advantageous. INstant TEmporal structure Learning (INTEL) (Liu et al., 2020) creates an ensemble of models by defining potentially occurring changes in the data distribution. This procedure requires *a priori* assumptions regarding the types of changes that might occur during the online phase, which is not necessary for our three approaches. A key advantage of EVARS-GPR and EVARS-GPR+ over existing methods such as GP-non-Bayesian clustering (GP-NBC) is the reuse of data through augmentation for the adjustment the prediction model, which enables a quick adaptation. Beyond that, we empirically demonstrated the computational efficiency of EVARS-GPR. Moving-Window GPR (Ni et al., 2012) involves computationally exhaustive periodical model refittings at each time step, which we also observed in the runtime comparison with EVARS-GPR. ETKA was outperformed by its comparison partners in terms of runtime. However, we compared our algorithm with a rather low-frequent periodical refitting scheme with three equidistant model adjustments during the whole online phase. As expected, a pure hyperparameter refitting without a new kernel search triggered by a CPD required lower runtimes, but was also outperformed with respect to the prediction error.

Despite these advantages compared to existing methods, our novel approaches also have potential for further improvement. EVARS-GPR and EVARS-GPR+ focus on changing data distributions visible in the output scale change and have proven their efficiency and applicability for these cases. However, in case of unjustified model adaptations, e.g., if the CPD detects a change point that is not related to the output scale, the data augmentation of EVARS-GPR could even lead to a model that is less accurate than an unadjusted predictor. ETKA performs a kernel search using AKS for model adaptation, which is computationally more expensive but also accounts for other types of data distribution changes. However, we also observed that a new kernel search from scratch might be necessary in cases with a more drastic change in system behavior. An interesting point for future research could be the combination of these three approaches. The change point-triggered model adjustment has advantages such as efficiency and a fast reaction after a change occurred, which we also demonstrated empirically. Thus, a CPD developed for various change point patterns should be part of such a combined algorithm. To account for different types of changing data distributions, it would be interesting to integrate them into the simulated scenarios or manipulate the real-world data for a dataset-specific parameterization accordingly to enable the CPD to detect all distributional shifts in the first place. Then, model adjustments could be made that explicitly account for the type of changing data distribution. To enable such

an approach, it would be necessary to distinguish between different change point patterns and react accordingly. For instance, in case of a more significant change in the output scale, EVARS-GPR might be suitable and efficient, and in case of a shift in the period length, a model adjustment using AKS could be appropriate. The accurate identification of change point patterns as an enabler is an interesting point for future research.

4.3 A Comprehensive Time Series Forecasting Framework

As a further contribution of this thesis, we aimed to publish a comprehensive time series forecasting framework that is easy to use, even without expert knowledge, and easy to extend, ensuring reproducible and comparable results as well as accessibility of novel approaches. With `ForeTiS`, we provide a powerful Python framework that addresses this objective. Our framework covers the entire time series forecasting pipeline, in contrast to many existing programming libraries, as Meisenbacher et al. (2022) conclude. `ForeTiS` is specifically focused on state-of-the-art time series forecasting, fully automating the pipeline while allowing for customization. Similar to AutoML frameworks, e.g., `auto-sklearn` (Feurer et al., 2015) or `AutoGluon` (Erickson et al., 2020), `ForeTiS` is user-friendly. However, these frameworks usually provide limited access to understand the procedure happening in the background, which we ensure for `ForeTiS` as an essential part when conducting research. Nevertheless, `ForeTiS` provides an easy-to-use interface that requires only a single line of code to conduct a comparative study employing state-of-the-art methods. This ease of use is one of the things that distinguishes `ForeTiS` from `pyWATTS` (Heidrich et al., 2021), `PyCaret` (Moez, 2023), and `sktime` (Löning et al., 2019). With these frameworks, users need to write several lines of code to achieve similar functionality, potentially requiring expert knowledge. In addition, `ForeTiS` already contains several time series prediction models as well as templates for the most widely-used frameworks `scikit-learn` (Pedregosa et al., 2011), `statsmodels` (Seabold & Perktold, 2010), `PyTorch` (Paszke et al., 2019), and `TensorFlow` (Abadi et al., 2015). This design allows for a straightforward extension of `ForeTiS` with various types of models and for a quick benchmarking of newly developed approaches. The support for this wide range of DL-based prediction models differs from `sktime` (Löning et al., 2019) and `PyCaret` (Moez, 2023), which only offer support for `Keras` and none of the common DL frameworks, respectively. As a big plus of `ForeTiS` compared to, for example, `Darts` (Herzen et al., 2022), `sktime` (Löning et al., 2019), and `Merlion` (Bhatnagar et al., 2021), our framework integrates and fully automates state-of-the-art Bayesian optimization for hyperparameter search. Another advantage of

ForeTiS is that it provides a command line interface via Docker (Merkel, 2014) as well as a Python package, enabling integration into existing code. We further support users with comprehensive online documentation, including step-by-step guides and hands-on video tutorials. In summary, we provide a powerful resource for end users who want to conduct comparative studies and researchers who want to develop novel time series forecasting approaches that differs from existing packages in several ways.

Nevertheless, several aspects are interesting for the future development of ForeTiS. Data exploration and visualization capabilities could assist users in determining the pipeline's parameters. For multivariate approaches, there are several options regarding features that could be predictive. For instance, one could consider external information such as weather or statistical values, e.g., past sales on the same day of the week in case of demand forecasting. This procedure quickly leads to a large number of possible features. One option for dimensionality reduction that is already integrated is PCA, but this method has the disadvantage of reducing interpretability. Regarding automated feature selection, various approaches exist, for example, forward and backward selection or correlation-based filtering, with the need to prevent information leakage to validation and test data (Guyon & Elisseeff, 2003). Integrating such approaches while taking into account computational resources is of great interest for future research. Moreover, there are several options for extending ForeTiS with further prediction models. First, approaches that simultaneously predict multiple target variables could be beneficial and efficient for several tasks. Second, ensemble forecasting, not only by using already integrated ensemble methods such as XGBoost, but also by combining different model types, is a potential improvement. Such methods have shown superior performance in recent forecasting competitions (Makridakis et al., 2020, 2022). Third, additional published prediction models, i.a., DeepAR (Salinas et al., 2020), could be integrated. The implementation of these new models is supported by ForeTiS, which already provides template classes and general functionalities and allows for quick benchmarking. Regarding benchmarking and the development of novel methods, it is also interesting to integrate techniques for simulating time series, allowing for the evaluation of novel approaches under pre-defined conditions. A major advantage of packages such as PyCaret (Moez, 2023) and sktime (Löning et al., 2019) is the large number of contributing software developers, which enables fast and comprehensive enhancements of their frameworks. ForeTiS provides useful capabilities, but is currently maintained and developed by a single research lab, making it difficult to compete with existing large libraries in the long run. Thus, we have intended to reduce obstacles for further contributors with our easy-to-extend design and by providing a comprehensive documentation.

4.4 Further Points of Discussion

As discussed above, we addressed some of the current challenges described in the introduction. In addition to these, we outline further points of discussion below, i.e., more general aspects and current challenges that we did not focus on.

Feature engineering is an essential part when employing multivariate time series forecasting approaches. Besides external information, such as meteorological or calendrical data, some datasets include covariates of the target variable. For example, a sales dataset often contains the number of sold units of the target variable and other articles. Using statistical information about these covariates, e.g., past sales of a competing product, could be helpful in attempting to predict future sales of the target variable. However, a dataset may contain various covariates, resulting in a large number of features when multiple statistical values are derived for each of them. Many of these features are probably not influencing the target variable significantly and are thus not helpful in training a prediction model. Moreover, a large number of features is problematic for most ML-based models, and performing an extensive feature selection would be computationally exhaustive. Therefore, choosing a subset of covariates for feature engineering makes sense. An expert could perform this selection. However, for larger numbers of covariates, this task becomes cumbersome and requires domain expertise in addition to knowledge about the dataset itself, e.g., characteristics of a company's customers in case of sales prediction. Hence, data-driven methods that automatically identify relevant ones to derive features are highly interesting. One possibility would be to consider correlations between the covariates and the target variable on a subset of the data and select the most relevant ones. However, this relationship may change as changing data distributions are an issue in time series forecasting. Furthermore, the datasets of small and medium-sized companies are often limited, so further reducing the number of samples for training a model could be problematic. Therefore, future research is necessary in this direction, which may have implications for several areas of time series forecasting.

Another challenge in demand forecasting, as described in Section 1.2, is that sales figures are often used to approximate demand, but the actual demand is unknown. If stock numbers are tracked, there are methods to include this information to account for out-of-stock situations in which demand and sales are not equal. For instance, this information could be included as a feature, or time steps in which an out-of-stock situation occurred could be treated as missing values and imputed (Fildes et al., 2022). Besides the drawbacks of these approaches, stock numbers may not have been recorded, especially for fast-moving

and perishable goods, as we observed in horticulture. Thus, the development of new methods that account for out-of-stock situations is an interesting topic for future research. Some patterns could be indicators of out-of-stock situations, e.g., a sharp decline in sales followed by an increase, which, in case of a seasonal time series, may also not be present in previous seasons. However, such patterns could also suggest a change point, especially when considering an online algorithm where the subsequent increase in sales is not visible at the time of processing. Thus, it seems reasonable to integrate such methods with CPD to prevent false alarms. By doing so, we could react accordingly, perhaps in a combined approach that takes into account different types of changes in the data distribution, as described in Section 4.2.

Many time series forecasting approaches, especially ML-based methods, require a certain amount of historical data. As outlined in Section 1.2, this is a challenge in this domain since data collection may take several years in case of annual seasonality, which applies, for instance, to many horticultural products. This issue is particularly relevant for small and medium-sized companies with less data and often a limited information technology infrastructure that has already been used for data collection without considering a time series forecasting project in the first place. In recent years, generative models, such as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) and Variational Auto-Encoders (VAEs) (Kingma & Welling, 2014), have shown impressive performance in creating synthetic data. Most of these methods were originally developed for image data, but have been successfully transferred to the synthesis of time series and tabular data. Therefore, it seems interesting to assess whether synthetically generated data can support the training of a time series forecasting model even when a limited amount of real data is available. As a major advantage, this approach could reduce the time needed to collect real historical data. In two Master's theses, we evaluated several generative models developed for synthesizing tabular and time series data. Besides time series data generators, we also included models designed for tabular data, since time series data could also be considered in this way. While this leads to a loss of temporal dependency, we can choose from a wider range of generative models. In a first Master's thesis (Eiglsperger, 2022), we focused on GANs, namely CTGAN (Xu et al., 2019) and TimeGAN (Yoon et al., 2019). We extended this work by benchmarking CTAB-GAN+ (Zhao et al., 2022), TVAE (Xu et al., 2019), diffusion model-based TAB-DDPM (Kotelnikov et al., 2022), and self-implemented diffusion models employing an MLP and an LSTM network, respectively (Pecini, 2023). In these two theses, we assessed the ability of the models to generate realistic samples, yielding promising results. It seems highly interesting to extend this preliminary research by

evaluating the influence on downstream prediction tasks when training models on synthetic data or a mixture of artificially generated and real data. Addressing the need to collect historical data by leveraging generative models could be beneficial for several time series forecasting applications.

Conclusion

The four main contributions of this thesis address several current challenges in time series forecasting. We first assessed the applicability of time series forecasting methods to predict horticultural sales, an application domain selected as representative of small and medium-sized companies dealing with perishable goods, despite changes in the data distribution. In a first proof of concept outlined in Section 3.1, we observed promising results and gained insights, but also identified limited capability of handling changing data distributions with respect to all prediction models. Nevertheless, our findings need to be assessed in a broader comparison using data from several companies along the value chain for more general conclusions, for which we provide guidance and interesting points for future research. In particular, we focused on online methods that account for changing data distributions and adjust a prediction model, namely GPR, accordingly. In this context, we contribute by publishing two algorithms: EVARS-GPR, described in Section 3.2, and ETKA, summarized in Section 3.3. We further extended the former to EVARS-GPR+, which is included as an unpublished result in this thesis. All three approaches leverage a CPD-based adaptation of the prediction model, but differ with respect to the integrated CPD and the model adjustment mechanism. EVARS-GPR augments existing data to achieve a quick refitting of the model's parameters, whereas ETKA employs a new kernel search using AKS. While the former focuses on changes that are visible on the output scale change in seasonal time series, ETKA is intended for further types of changes, e.g., a change in the periodicity, in both seasonal and non-seasonal data. EVARS-GPR+ extends EVARS-GPR by including a non-augmented model adjustment when a change point was detected, but the extent of the output scale change compared to previous seasons does not exceed a specified threshold used to trigger the augmented refitting. We demonstrated the usefulness of EVARS-GPR

and ETKA on both simulated and real data. Beyond that, we developed ForeTiS, a comprehensive time series forecasting framework in Python. Our framework is easy to use and extend, making it a powerful resource for end users as well as model developers, see Section 3.4. In summary, this thesis contributes to research in time series forecasting by (i) assessing time series forecasting for demand prediction in a domain with mainly small and medium-sized companies dealing with perishable goods in the presence of changing data distributions, (ii) providing accurate and computationally efficient approaches to account for changing data distributions, and (iii) developing a user-friendly and easy-to-extend Python framework that supports accessibility and reproducibility of time series forecasting as well as the development of new methods. Beyond these contributions, there are several interesting opportunities for future research based on this work, as outlined in Chapter 4. Thus, we address some of the current challenges in time series forecasting, especially changing data distributions, provide several follow-up ideas, and leave room for future research.

Appendix A

Publication A

The subsequent pages show the original full version of the following publication:

PUBLICATION A:

Florian Haselbeck, Jennifer Killinger, Klaus Menrad, Thomas Hannus, and Dominik G. Grimm (2022). Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions. *Machine Learning with Applications*, 7, 100239. <https://doi.org/10.1016/j.mlwa.2021.100239>



Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions

Florian Haselbeck^{a,b,*}, Jennifer Killinger^{a,b}, Klaus Menrad^{c,d}, Thomas Hannus^e, Dominik G. Grimm^{a,b,f,*}^a Technical University of Munich, TUM Campus Straubing for Biotechnology and Sustainability, Bioinformatics, Schulgasse 22, D-94315 Straubing, Germany^b Weihenstephan-Triesdorf University of Applied Sciences, Bioinformatics, Petersgasse 18, D-94315 Straubing, Germany^c Technical University of Munich, TUM Campus Straubing for Biotechnology and Sustainability, Marketing and Management of Biogenic Resources, Schulgasse 22, D-94315 Straubing, Germany^d Weihenstephan-Triesdorf University of Applied Sciences, Marketing and Management of Biogenic Resources, Am Essigberg 3, D-94315 Straubing, Germany^e Weihenstephan-Triesdorf University of Applied Sciences, Retail Management, Am Staudengarten 10, D-85354 Freising, Germany^f Technical University of Munich, Department of Informatics, Boltzmannstr. 3, D-85748 Garching, Germany

ARTICLE INFO

Dataset link: <https://github.com/grimmlab/HorticulturalSalesPredictions>

Keywords:

Sales Forecasting
Time Series Forecasting
Comparative study
Horticulture
Machine Learning

ABSTRACT

Forecasting future demand is of high importance for many companies as it affects operational decisions. This is especially relevant for products with a short shelf life due to the potential disposal of unsold items. Horticultural products are highly influenced by this, however with limited attention in forecasting research so far. Beyond that, many forecasting competitions show a competitive performance of classical forecasting methods. For the first time, we empirically compared the performance of nine state-of-the-art machine learning and three classical forecasting algorithms for horticultural sales predictions. We show that machine learning methods were superior in all our experiments, with the gradient boosted ensemble learner XGBoost being the top performer in 14 out of 15 comparisons. This advantage over classical forecasting approaches increased for datasets with multiple seasons. Further, we show that including additional external factors, such as weather and holiday information, as well as meta-features led to a boost in predictive performance. In addition, we investigated whether the algorithms can capture the sudden increase in demand of horticultural products during the SARS-CoV-2 pandemic in 2020. For this special case, XGBoost was also superior. All code and data is publicly available on GitHub: <https://github.com/grimmlab/HorticulturalSalesPredictions>.

1. Introduction

Predicting future demand to support corporate analysis and decision-making is a potential competitive advantage in many domains. One solution for forecasting customer demand are time series prediction methods. With accurate estimations, company managers can quickly react to changing market signals and consequently adjust their procurement and production plans. These possibilities may lead to increased revenues when early adoption due to a rising demand is applied or decreasing costs as a response to a decline (Ivanov et al., 2019). This becomes even more relevant when dealing with goods that have a limited shelf life and can therefore only be kept in stock for a few days (Duan et al., 2012). Horticultural products are strongly influenced by these issues.

Currently, there are no scientific publications regarding time series prediction for sales of horticultural products, although total sales of

ornamental plants are worth €9.4 billion in 2020 in Germany (Zentralverband Gartenbau e.V., 2020). Horticultural sales are usually characterized by a strong seasonality. In addition, sales cycles of certain products are shaped by abrupt changes in both directions of rising and decreasing numbers. Moreover, various external factors such as holidays or regional events affect demand. In addition, some of these factors, such as weather forecasts, are uncertain and only available for a short period (Behr et al., 2012). Furthermore, the short shelf life of horticultural products, particularly of cut flowers, is a challenge for operational decisions. In practical operations, this may cause out-of-stock situations with possibly missed sales as well as excess-stock cases, which often lead to the disposal of products. Besides financial loss, the latter induces environmental damage due to wasted resources during production and transport.

* Corresponding authors at: Technical University of Munich, TUM Campus Straubing for Biotechnology and Sustainability, Bioinformatics, Schulgasse 22, D-94315 Straubing, Germany.

E-mail addresses: florian.haselbeck@hswt.de (F. Haselbeck), jennifer.killinger@tum.de (J. Killinger), klaus.menrad@hswt.de (K. Menrad), thomas.hannus@hswt.de (T. Hannus), dominik.grimm@hswt.de (D.G. Grimm).

<https://doi.org/10.1016/j.mlwa.2021.100239>

Received 22 October 2021; Received in revised form 3 December 2021; Accepted 15 December 2021

Available online 21 December 2021

2666-8270/© 2021 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Classical forecasting methods such as Autoregressive Integrated Moving Average and Exponential Smoothing are still widely applied in research and industry. Despite their rather simple concept, they often show a competitive performance. As demonstrated in several publications and contests such as the M-Competitions, they are even able to outperform more complex machine learning (ML) approaches, e.g. Artificial Neural Networks (Kolassa, 2021; Makridakis et al., 2020, 2021). Thus, although ML algorithms are becoming more common in the forecasting literature, it is not clear that they are superior, but dependent on the application and data.

These horticultural product-specific aspects and unsolved questions in forecasting encourage conducting research in this domain. Hence, in this paper, we present a first-time comparison of ML-based and classical time series prediction methods to forecast sales for products such as potted plants, cut flowers and shrubs. We did not consider non-herbal products merchandized in this industry, e.g. plant pots, and edible products such as vegetables and fruits, which were already taken into account by other researchers (Arunraj & Ahrens, 2015; Priyadarshi et al., 2019; Sankaran, 2014; Shukla & Jharkharia, 2011). First, we want to answer the question whether ML is superior compared to classical forecasting approaches for horticultural sales predictions. Second, we investigate potential improvements by multivariate concepts making use of external factors, such as weather or holiday data, by comparing with classical univariate methods. Third, we examine the computational resource consumption of the applied methods, a critical issue with the necessity of model refittings due to potentially changing data distributions during the live operation of a forecasting system in mind. In addition, we evaluate whether the models are able to capture sudden change in data, such as strong increases in demand during the SARS-CoV-2 pandemic in 2020.

To this end, we present a comparative study of nine state-of-the-art ML and three classical forecasting methods. Regarding classical forecasting approaches, we used the univariate techniques Exponential Smoothing and Seasonal Autoregressive Integrated Moving Average as well as the multivariate extension of the latter. Furthermore, we show a comparison to Multiple Linear Regression models with different regularizations. We included nonlinear ML methods such as Artificial Neural Networks as well as Long Short-Term Memory Networks. Moreover, we applied the ensemble learner Extreme Gradient Boosting (XGBoost), and a nonparametric Bayesian approach by implementing Gaussian Process Regression. Furthermore, we implemented a setup with a regular refit of the forecasting model in order to simulate a potential scenario for a productive operation with a continuous update of company-specific sales data and a potentially changing data distribution in mind. Finally, we are able to provide initial insights in the new forecasting domain of horticultural sales.

The remainder of this paper is organized as follows. In Section 2, we provide an overview of related work. Afterwards, we describe the materials and methods, see Section 3. In Section 4, we outline and discuss our results, before we draw conclusions.

2. Related Work

In the following section, we summarize classical forecasting and ML methods that can be used for predicting demand. Then, we describe empirical comparisons and related work from similar domains, such as the food and tourism sector.

2.1. Machine Learning-based and Classical Time Series Forecasting Methods

In Table 1, we provide an overview of the approaches discussed in this paper. Classical forecasting methods use chronologically ordered time series data and try to predict future sequences, usually by projecting statistical information recovered from historical data. Methods can be divided into univariate and multivariate approaches, with the

latter one using external information in addition to the time series itself. Two common univariate methods are Exponential Smoothing (ES) and Autoregressive Integrated Moving Average (ARIMA) (Box et al., 2016; Holt, 1957; Winters, 1960). In its simple form, ES is a weighted sum of past observations of a time series with weights decaying exponentially. In contrast, modeling autocorrelations – the correlation between a series and a lagged version of itself – is the key idea of ARIMA. A common extension, SARIMA, involves seasonal parts (Gardner, 2006; Hyndman & Athanasopoulos, 2018). Furthermore, a multivariate version involving external factors called SARIMAX can be formulated (Arunraj et al., 2016).

Sales forecasting can also be defined as a regression task for ML methods. In ML, we can distinguish between Frequentist and Bayesian formulations of various models with the latter one referred to as probabilistic in this work. Here, Frequentist ML methods minimize the empirical risk for a certain loss function including different regularization terms to determine point estimates of the most likely model parameters. Probabilistic approaches instead use Bayes' theorem to calculate a full posterior distribution over the parameters given the data and a prior (Bishop, 2009; James et al., 2017). For regression tasks, Multiple Linear Regression (MLR) is often used as a baseline. Common approaches are Ridge, Lasso and Elastic Net Regression, which involve different regularization terms in order to prevent overfitting (Hoerl & Kennard, 1970; James et al., 2017; Santosa & Symes, 1986; Tibshirani, 1996; Zou & Hastie, 2005). MLR can further be defined in a Bayesian perspective as follows: $p(y|x, \mathbf{w}) = \mathcal{N}(y|x^T \mathbf{w}, \sigma^2)$, where the target variable y follows a Gaussian distribution with the mean $x^T \mathbf{w}$ (determined by the predictors x and the weights \mathbf{w}) and the variance σ^2 . If the weights of the model are constrained to a zero-mean Gaussian prior, the solution is equivalent to Ridge Regression and therefore called Bayesian Ridge Regression (BayesRidge). Automatic Relevance Determination (ARD) is related to it, but introduces an individual variance for each weight (Bishop, 2009; James et al., 2017; Tipping, 2001). A widely applied approach in time series prediction are Artificial Neural Networks (ANN) (Rosenblatt, 1958; Zhang et al., 1998). Especially Recurrent Neural Networks (RNN), such as Long Short-Term Memory Networks (LSTM), are suitable for time series data, since they are able to capture temporal dependencies by definition (Hewamalage et al., 2021; Hochreiter & Schmidhuber, 1997). Several publications and forecasting competitions indicate a superiority of combined methods, e.g. achieved via ensemble learners (Bojer & Meldgaard, 2021; Petropoulos et al., 2018). XGBoost (XGB) is an ensemble technique that uses gradient boosted regression trees. With this approach, decision trees are sequentially added in a greedy manner based on the gradient of the loss function in order to correct the errors made by the current ensemble (Chen & Guestrin, 2016). With regard to the practical use of demand forecasts, the uncertainty of a prediction value seems profitable. Providing those is a main advantage of the nonparametric Bayesian method Gaussian Process Regression (GPR) (Williams & Rasmussen, 1996). The determining parameters of a Gaussian Process are the kernel $k(x, x')$, which consists of the covariance value between any two sample points x and x' resulting in a $n \times n$ matrix for a training set length of n and the mean function $m(x)$. The assumption is that the similarity between samples reflects the strength of the correlation between their corresponding target values. Therefore, the function evaluation can be seen as a draw from a multivariate Gaussian distribution defined by $m(x)$ and $k(x, x')$. Thus, Gaussian Processes are rather a distribution over functions (Rasmussen & Williams, 2008; Roberts et al., 2013). More detailed descriptions can be found in Appendix A.

2.2. Time Series Forecasting Competitions and Related Domains

There are several publications regarding empirical comparisons of forecasting approaches, e.g. the influential M-competitions that started in 1982 (Bojer & Meldgaard, 2021; Hong et al., 2019; Lloyd, 2014;

Table 1
Overview of classical forecasting and machine learning methods with their abbreviations and certain characteristics.

Category	Algorithm	Abbreviation	Univariate	Multivariate	Probabilistic
Classical forecasting	Exponential Smoothing	ES	×		
	Seasonal Autoregressive Integrated Moving Average	SARIMA	×		
	Seasonal Autoregressive Integrated Moving Average with external factors	SARIMAX		×	
Machine learning	Lasso Regression	LassoReg		×	
	Ridge Regression	RidgeReg		×	
	Elastic Net Regression	ElasticNet		×	
	Artificial Neural Network	ANN		×	
	Long Short-Term Memory Network	LSTM		×	
	Extreme Gradient Boosting (XGBoost)	XGB		×	
	Bayesian Ridge Regression	BayesRidge		×	×
	Automatic Relevance Determination	ARD		×	×
	Gaussian Process Regression	GPR		×	×

Makridakis et al., 1982, 1993; Makridakis & Hibon, 2000; Makridakis et al., 2018, 2020, 2021; Stepnicka & Burda, 2017). Nevertheless, these competitions do not show a general superiority of a specific approach. While the M4-competition did not yield advantages for single ML models, but for approaches combined with classical forecasting methods, gradient boosted trees were superior for the recent M5-study with strongly correlated time series and explanatory variables (Makridakis et al., 2018; Seaman & Bowman, 2021). This is in accordance with a recent review of several Kaggle forecasting competitions (Bojer & Meldgaard, 2021). So in summary, combined respectively ensemble methods were advantageous in many empirical comparisons without yielding a generally superior technique.

Besides these broad forecasting competitions, there are publications in similar domains such as food and tourism demand forecasting. Food demand forecasting is a domain with similarities to horticultural sales, especially when dealing with agricultural products such as vegetables or fruits, which are also perishable and seasonal. Arunraj et al. showed the superiority of SARIMAX supplemented with holiday and promotional features over its univariate version SARIMA when predicting sales of bananas in a retail store (Arunraj et al., 2014). A comparison of several ML methods forecasting food sales of a Japanese supermarket chain with an emphasis on the influence of meteorological data was done in (Liu & Ichise, 2017). It yielded a better performance of a LSTM compared to others, e.g. Random Forest. A recently published comparative study on bakery sales predictions, including LSTM and gradient boosted regression trees, indicated a superiority of ML algorithms (Huber & Stuckenschmidt, 2020). Besides that, the demand for ornamental plants is characterized by a strong seasonality with abrupt changes in requests and influenced by various external factors, such as the weather and holidays. Thus, tourism is another domain, which shares similar attributes. Athanasopoulos et al. compared the performance of several univariate time series algorithms with their multivariate extensions based on 366 monthly, 427 quarterly and 518 annual series. They concluded that the first ones deliver better results, but also mentioned that this is in contrast to other publications (Athanasopoulos et al., 2011). Jiao and Chen provided a review of methodological developments in tourism forecasting during the 2008 to 2017 time period. According to them, econometric and time series models are still widely applied and often lead to competitive predictions. Besides that, they observed a trend in enhancing time series with additional features as well as an increasing use of ML methods. Furthermore, combining different approaches often seems to yield better results than single ones (Jiao & Chen, 2019). In summary, these findings in similar domains provide some guidance for horticultural demand forecasting, but also do not show a general superiority of classical or ML-based approaches.

3. Materials and Methods

In the next section, we provide insights into the data and its preparation. Afterwards, we describe the model selection and training. Finally, we outline the evaluation metrics and baselines. All code and data is publicly available on GitHub: <https://github.com/grimmlab/HorticulturalSalesPredictions>

3.1. Data Preparation

In the following, we provide an overview on the datasets. Afterwards, we summarize the feature engineering and data preprocessing steps in detail.

3.1.1. Datasets

For our analysis, we used typical horticultural retail sales data from Germany. We distinguished between five datasets based on two data sources, as shown in Fig. 1. *OwnDoc* was manually created by a daily documentation of sales numbers of tulips. These datasets have scientifically interesting characteristics, such as a strong seasonality in early spring with abrupt changes in demand. Regarding predictions, we focused on private customer sales of tulips, subsequently called *SoldTulips*. Records for the *OwnDoc* dataset begin on February 7, 2020 and last until May 11, 2020, as the tulip season usually ends in mid-May. Thus, data exists for one seasonal cycle lasting about three months. *OwnDoc* shows a detailed product aggregation level of the target variable with the plant species and delivers quantities. As indicated in Fig. 1, we generated two datasets based on *OwnDoc*. There is a 16 day long documentation gap starting on April 10. Thus, we derived a variant for which we only use values before that period (*OwnDoc_SoldTulips_short*), whereas we imputed them for the long one (*OwnDoc_SoldTulips_long*).

CashierData was created based on an electronic cashier system providing a summary of all sales. These sales figures, which are aggregated into product groups, were accumulated to daily turnovers. As we focused on herbal products, we selected cut flowers (*CutFlowers*) and potted plants (*PotTotal*) as target variables. Beyond that, *CashierData* differs from *OwnDoc* regarding several properties. It ranges from December 2016 to August 2020 and as a result contains several seasonal cycles. Additionally, due to the aggregation into product groups, we can only see patterns of these whole clusters instead of individual plants. Since *CashierData* is based on exports from an electronic cashier system, for which a gapless logging is required by tax law, there are no missing values. Furthermore, numbers in *CashierData* represent the turnover in euros. For *CashierData*, we derived three datasets, see Fig. 1. The first one reflects sales of *CutFlowers* over the whole period (*CashierData_CutFlowers*). Besides this, we observe a sharp increase of revenues for *PotTotal* in the first half of 2020, probably caused by the SARS-CoV-2 pandemic. Therefore, we separated an alternative version ending 12/2019 (*CashierData_PotTotal_short*) in addition to the one ranging over the whole period (*CashierData_PotTotal_long*).

Table 2 shows characteristics for all datasets on a daily basis as well as on a weekly one for all *CashierData* versions. As already mentioned, *CashierData* does not contain missing values in contrast to *OwnDoc*. Furthermore, the standard deviations and maximum values are high compared to the mean, reflecting a dataset with strong variations. Due to this and the practical usefulness of predictions, we resampled all variants based on *CashierData* to a weekly cycle.

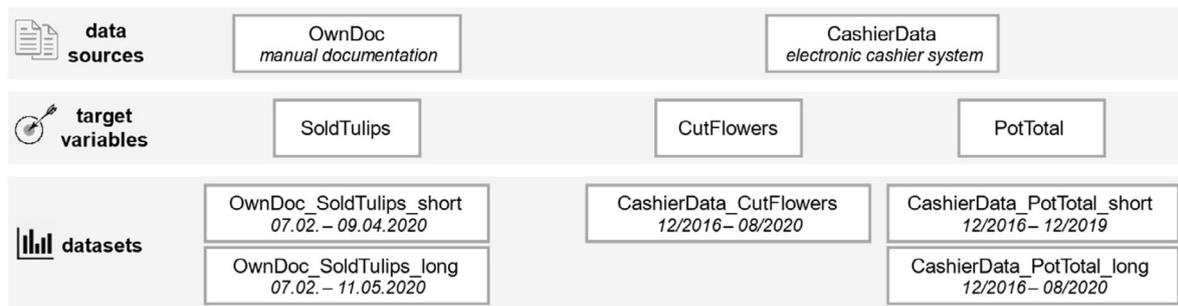


Fig. 1. Overview of all five datasets based on the two data sources OwnDoc and CashierData. OwnDoc_SoldTulips_short was derived due to a 16 day long period with missing values, CashierData_PotTotal_short because of a strong sales increase in 2020.

Table 2

Characteristics for all datasets. Statistics are on a daily basis as well as weekly resampled for CashierData, with the weekly values shown below the daily ones.

Dataset	Period	Samples	Target variable characteristics			
			Missing value ratio	Mean	Standard deviation	Maximum
OwnDoc_SoldTulips_short	07.02.-09.04.2020	63	1/63	158.90	107.47	428.00
OwnDoc_SoldTulips_long	07.02.-11.05.2020	95	17/95	145.36	103.36	428.00
CashierData_CutFlowers	12/2016-08/2020	1359	0/1359	244.00	244.74	2346.45
		195	0/195	1700.47	719.31	4828.85
CashierData_PotTotal_short	12/2016-12/2019	1115	0/1115	160.94	210.41	1605.30
		160	0/160	1121.58	1041.74	5358.70
CashierData_PotTotal_long	12/2016-08/2020	1359	0/1359	189.55	267.04	1926.40
		195	0/195	1321.04	1384.95	7529.25

3.1.2. Feature Engineering

All datasets have been enriched with certain features. To examine whether external factors support the forecasting of horticultural sales, we added daily weather and holiday information. Historical observations for the former were provided by the German Meteorological Service. Regarding holiday information, we considered public as well as school holidays. The included features and their explanations can be found in Table 3.

Further, two categories of additional features were derived: (i) calendric and (ii) statistical features, as summarized in Table 3. Calendric features are date-based properties such as the weekday and typically outstanding selling days, e.g. Valentine’s and Mother’s Day. Furthermore, we created counters for these special days and public holidays as sales might be higher close to such an event and lower afterwards. Statistical features are, among others, lagged variants of the target variables (both seasonal and non-seasonal). Moreover, we added lagged weather information (mean temperature, precipitation amount and sun duration), because the conditions prior to a date of sale might be influential. Beyond that, we derived rolling statistical values (both seasonal and non-seasonal), such as its mean over a fixed period prior to a sales date. Finally, for daily data, we calculated rolling statistical numbers for a specific weekday, e.g. the mean sales of the last four instances. Eventually, we divided these features into four different featuresets, see Table 3. We included raw features from external sources for all four. Beyond that, sub2 was focused on calendric features and sub3 on statistical ones. The full featureset contained all features.

3.1.3. Data Imputation and Dimensionality Reduction

Many algorithms cannot handle missing values, except for XGB. Therefore, we applied different strategies for data imputation: Mean, K-Nearest-Neighbors (KNN) and Iterative Imputation. The featuresets sub3 and full introduce additional missing values at the beginning of the dataset because statistical values such as lagged variables cannot be calculated. We imputed these as well and did not drop samples as the amount of lacking seasonal features would lead to a large information loss. There are no missing values for CutFlowers and PotTotal on CashierData and only few in the raw weather data. Moreover, we did not insert new ones using featuresets sub1 and sub2 as both do not

contain statistical features. Therefore, the effect of different imputation methods seemed negligible, and we only considered Mean Imputation for these setups. In summary, we can distinguish five different feature settings, including the univariate case, which were combined with the data imputation approaches. Fig. 2a provides an overview of all setups we considered.

To examine the effect of dimensionality reduction on the multivariate feature settings, we considered two variants: using the original featureset or running a Principal Component Analysis (PCA) selecting the components which explain at least 95% of the variance (Jolliffe & Cadima, 2016). As shown in Fig. 2a, we did not include PCA for the sub1 featureset due to the low dimensionality.

3.2. Model Selection and Training

We included all algorithms summarized in Table 1 to show a comparative study of ML and classical forecasting methods. ES and SARIMA were selected as two common univariate techniques. SARIMAX was included as a multivariate classical alternative for a fair comparison with ML methods using additional features. MLR approaches were taken into account as a baseline for regression problems. Furthermore, ML methods able to capture nonlinear relationships were considered. ANNs have proven their suitability for time series predictions (Zhang et al., 1998). As temporal dependencies might be important, RNNs were included and designed with LSTM cells as preceding research suggests (Hewamalage et al., 2021). Beyond that, combined methods were superior in many comparisons (Bojer & Meldgaard, 2021; Petropoulos et al., 2018). Therefore, we included XGB. Eventually, with GPR, a nonparametric Bayesian method with its inherent ability to model uncertainty was selected. Furthermore, we applied normalization and standardization methods such as the Yeo-Johnson power transformation (Yeo & Johnson, 2000) or logarithmic scaling as certain methods are sensitive to input distributions. A summary of all hyperparameters and transformations for all algorithms is shown in Appendix B (Tables B.1 to B.8). Hence, we randomly sampled parameter combinations for model and preprocessing configuration.

We employed time series cross-validation with a regular model refit in order to simulate a potential scenario for the productive operation

Table 3

Overview of features and structure of the featuresets. Features are based on external sources or derived based on others. With regard to raw features from external sources, holiday and weather information was added. Furthermore, calendric and statistical features were derived. Based on that, four multivariate featuresets were designed. Their structure can be seen in the last four columns.

Source	Category	Feature	Explanation	Featureset			
				sub1	sub2	sub3	full
Features from external sources	Raw features	Public holiday	Name of public holiday	×	×	×	×
		School holiday	Name of school holiday	×	×	×	×
		Mean temperature	Daily mean air temperature	×	×	×	×
		Mean humidity	Daily mean relative humidity	×	×	×	×
		Precipitation amount	Daily mean and total precipitation amount	×	×	×	×
		Precipitation flag	Ratio of hours with precipitation and flag for full day	×	×	×	×
		Sun duration	Daily mean and total sun duration	×	×	×	×
Derived features	Calendric features	Date based features	Day of the month, weekday, month, quarter		×		×
		Working day	Flag showing if the day is a working day		×		×
		Special days	Valentine's and Mother's Day added to public holidays		×		×
		Public holiday counter	Counter for a public holiday starting seven days before and ending three days after with separate counters for Easter as well as Valentine's and Mother's Day		×		×
	Statistical features	Lagged variables	Lagged versions of target variables and weather features mean_temp, total_prec_height_mm and total_sun_dur_h			×	×
		Seasonal lagged variables	Seasonal lagged versions of same variables as above			×	×
		Rolling statistics	Rolling mean, median and maximum within a window for same variables as above			×	×
		Seasonal rolling statistics	Seasonal rolling mean, median and maximum within a window for target variables			×	×
		Rolling weekday statistics	Rolling mean, median and maximum within a window calculated for each weekday on target variables			×	×

(a) Applied combinations of feature settings and preprocessing methods

features	univariate	sub1	sub2	sub3	full
imputation	✓ None				
	∅ Mean				
	↗ KNN				
	↻ Iterative				
<i>not included for CashierData</i>					
dimensionality reduction	✓ None				
			↘ PCA	↘ PCA	↘ PCA

(b) Time series cross-validation

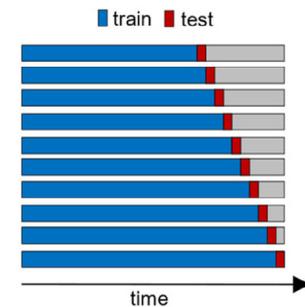


Fig. 2. (a) Overview of the applied combinations of the five feature settings and the preprocessing methods. The overview shows the preprocessing methods combined for each of the feature settings. For CashierData, certain imputation strategies are excluded for settings with no or few missing values. PCA was considered in multivariate cases and not performed on featureset sub1 due to its low dimensionality. (b) Time series cross-validation. Visualization of the evaluation on a rolling forecasting origin, which we employ to simulate a productive operation of a forecasting system with a continuous data update.

of a forecasting tool with a continuous update of company-specific sales data and a potentially changing data distribution in mind. Fig. 2b shows a visualization of this approach. First, we separated a training set covering 80% of the whole dataset to select the best working hyperparameter combination. Then, we simulated an online scenario using the remaining data. Whenever a new sample was available, we forecasted the next target value and refitted the model parameters keeping the already optimized hyperparameters fixed. A model configuration was evaluated by the average performance across the whole test set according to the evaluation metrics described in Section 3.3 (Hyndman & Athanasopoulos, 2018). For the best working solutions, we ran an in-depth optimization selecting more parameter combinations on a denser grid.

Beyond that, we compared the runtime needed by each algorithm. For this purpose, we selected *OwnDocSoldTulipsLong* with *sub1* as well as *CashierData_CutFlowers* with *full* to include opposing examples

regarding size. Furthermore, we randomly sampled 100 parameter combinations for every algorithm. Then, we ran the whole optimization for one training and prediction loop and calculated the average computation time. All runs for these experiments were executed on the same machine with four 4.0 GHz Intel i7-6700K CPUs, 62 GB memory and two Nvidia GeForce GTX 1080 Ti GPUs used for ANN and LSTM optimization.

The source code is written in Python 3.8 and published on GitHub: <https://github.com/grimmlab/HorticulturalSalesPredictions>. A list of all used packages and their versions can be found in this repository, including libraries such as Gpflow (Matthews et al., 2017), Matplotlib (Hunter, 2007), NumPy (Harris et al., 2020), pandas (McKinney, 2010), PyTorch (Paszke et al., 2019), scikit-learn (Pedregosa et al., 2011), SciPy (Virtanen et al., 2020), seaborn (Waskom, 2021) and statsmodels (Seabold & Perktold, 2010). We used Docker to ensure a standardized

working environment on different servers. A Dockerfile for its setup is included in the repository.

3.3. Evaluation Metrics and Baselines

For evaluation, we used the scale-dependent metric Root Mean Squared Error (RMSE) as well as the relative measures Mean Absolute Percentage Error (MAPE) and Symmetric Mean Absolute Percentage Error (sMAPE). With the forecast value \hat{y}_i , the true value y_i and the length of the evaluated set n , they are defined as follows:

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

$$sMAPE = \frac{100\%}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{\frac{(|y_i| + |\hat{y}_i|)}{2}}$$

$$MAPE = \frac{100\%}{n} \sum_{i=1}^n \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

For all three evaluation measures, a lower value reflects a better performance. In order to prevent division by zero when using MAPE, we added a constant (0.1) to the denominator. MAPE values can get large, if sales numbers are close to zero (common in datasets with daily observations). This circumstance is less severe for sMAPE, because it is divided by the mean of y_i and \hat{y}_i , but is still present as the predicted value might also be close to zero in these cases. Furthermore, due to the quadratic term, RMSE is sensitive to outliers. On the basis of these weaknesses and the lack of a universal evaluation metric for forecasting, it is common to assess the performance compared to baseline methods (Hyndman & Koehler, 2006). To that end, we used the ones described in Table 4.

Table 4
Baseline methods with length of training set T and seasonal period m .

Historical Average (HA)	$\hat{y}_i = \frac{1}{i-1} \sum_{j=1}^{i-1} y_j$
Random Walk (RW)	$\hat{y}_i = y_{i-1}$
Seasonal Random Walk (seasRW)	$\hat{y}_i = y_{i-m}$

HA predicts the mean of all known values. RW predictions are equal to the last value, whereas seasRW uses the known observations of the last season (Hyndman & Athanasopoulos, 2018).

4. Results and Discussion

In the next section, we give an overview of our results followed by a more detailed analysis of the best performing algorithms. Afterwards, the influence of different features as well as the runtime are analyzed. Finally, a discussion of the results is provided.

4.1. Results Overview

For a full analysis of our experiments, we generated an overview of the best results for all five datasets with respect to the three evaluation metrics (15 comparisons in total) for all baselines, algorithms and featuresets, see Appendix C and Supplementary 1. Our results show that a baseline (RW, seasRW and HA) was never best overall. However, MLR approaches (LassoReg, RidgeReg, ElasticNet, BayesRidge and ARD) were inferior in several cases on both *OwnDoc* variants and *CashierData_PotTotal_long*. If we exclude MLR algorithms, only in three out of the 15 comparisons, one of the baselines performed better than at least one ML method, with two of these cases occurring on the smallest dataset (*OwnDoc_SoldTulips_short*).

Following this comparison with baseline methods, we selected the best performances of each algorithm, independent of featuresets and preprocessing methods. The results are summarized in Tables D.1 and D.2, see Appendix D.1. Fig. 3a–e visualize them for all datasets (rows) and evaluation metrics (columns). In all comparisons, a ML-based method performed best. The results of the ML and classical approaches

were rather comparable for *OwnDoc*, whereas the advantage of ML techniques was larger for *CashierData*, probably due to the size of the datasets. In total, XGB was the leading method in 13 out of 15 comparisons, outperformed by LSTM two times for *CashierData_PotTotal_short*. Regarding univariate methods, SARIMA was competitive for both *OwnDoc* variants and the SARS-CoV-2-influenced *CashierData_PotTotal_long* dataset. Its multivariate extension, SARIMAX, achieved results close to the best ones for both *OwnDoc* versions. Furthermore, GPR and LSTM delivered competitive results in several cases, especially on all *CashierData* variants. All MLR methods as well as ES fell behind in most conditions, and ANN only kept up in a few instances. However, in summary, we can conclude a superiority of ML-based approaches.

4.2. Best Performing Algorithms

We subsequently focused on the best performing algorithms, so mainly ML-based methods. As described in Section 3.2, we conducted a further optimization of the best performers on a denser grid of hyperparameters. The results of these in-depth runs and a comparison with the results prior to them can be found in Tables 5 and 6. Overall, we again observed a superiority of ML-based approaches. Improvements based on the in-depth runs were rather small or mediocre with only one change regarding the overall ranking: XGB outperformed LSTM on RMSE for *CashierData_PotTotal_short*. In summary, XGB’s predominance was extended, leading in 14 out of 15 comparisons after the in-depth optimization.

With regard to *OwnDoc* variants presented in Table 5, we observed that XGB was the clear winner. However, it was closely followed by SARIMAX for the shorter dataset. Besides that, GPR and LSTM were competitive for sMAPE. GPR was furthermore close for MAPE with two second best performances.

Table 5
Top results for *OwnDoc* after in-depth optimization. A lower value reflects a better performance. Results prior to in-depth optimization are given in brackets in case of an improvement. The best results overall are printed in bold. In-depth optimization was not done for SARIMA on *OwnDoc_SoldTulips_short*.

Algorithm	<i>OwnDoc_SoldTulips_short</i>			<i>OwnDoc_SoldTulips_long</i>		
	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE
SARIMA	–	–	–	48.70 (52.02)	55.83	2366.01 (3675.31)
SARIMAX	58.06	52.58	85.75	48.45 (50.35)	52.37	203.34 (231.52)
LSTM	73.66	52.49	707.61	56.06	50.04	2884.77
XGB	57.11	50.89	34.21 (35.98)	43.48 (43.52)	48.65	52.87 (54.22)
GPR	68.14	52.27	67.99	56.05	52.43	65.84

Table 6 shows a similar overview for all *CashierData* variants. For *CashierData_CutFlowers*, XGB was the top algorithm in all comparisons. Besides that, the ensemble learner improved its performance on RMSE for *CashierData_PotTotal_short* and consequently outperformed the previous best algorithm LSTM. However, the results of the LSTM were comparable for this dataset. Regarding *CashierData_PotTotal_long*, XGB performed best and GPR delivered the second-best results for RMSE and sMAPE.

Beyond that, we show example plots of the best performing algorithms in Fig. 4. Each of them displays the ground truth accompanied by the predictions of two of the top performing models. We observed that several predictions are close to the ground truth, but there are also problematic regions, e.g. the end of the test period on both *OwnDoc* datasets. Partially, this is due to its limitation to only one season. Because of this, we were not able to capture the high sales on May 10 at the end of *OwnDoc_SoldTulips_long*. All algorithms predicted low sales as it was a Sunday, but actually they were high because it was Mother’s Day. Furthermore, the demand peaks of *CashierData_PotTotal_long* during the SARS-CoV-2 pandemic were hard

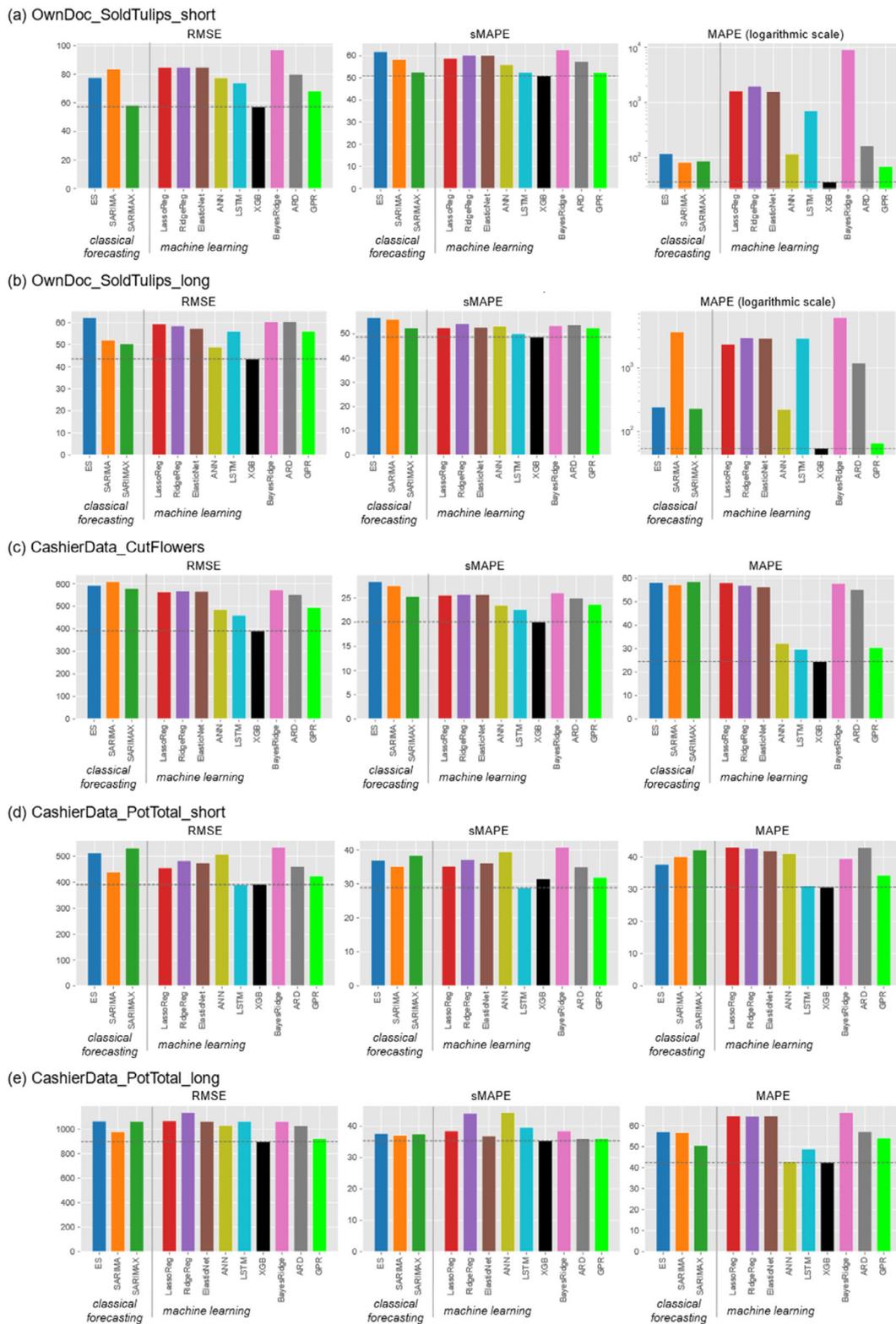


Fig. 3. Top results for each algorithm and dataset (rows) with respect to all evaluation metrics (columns). A lower value reflects a better performance. Classical forecasting and machine learning approaches are separated by a small horizontal offset and a vertical line. The best outcome in each plot is marked by a horizontal dashed line. For the exact values, see Tables D.1 and D.2 in Appendix D.1. MAPE values for both OwnDoc variants are plotted on a logarithmic scale as the results differ in a range from two- to four-digit numbers.

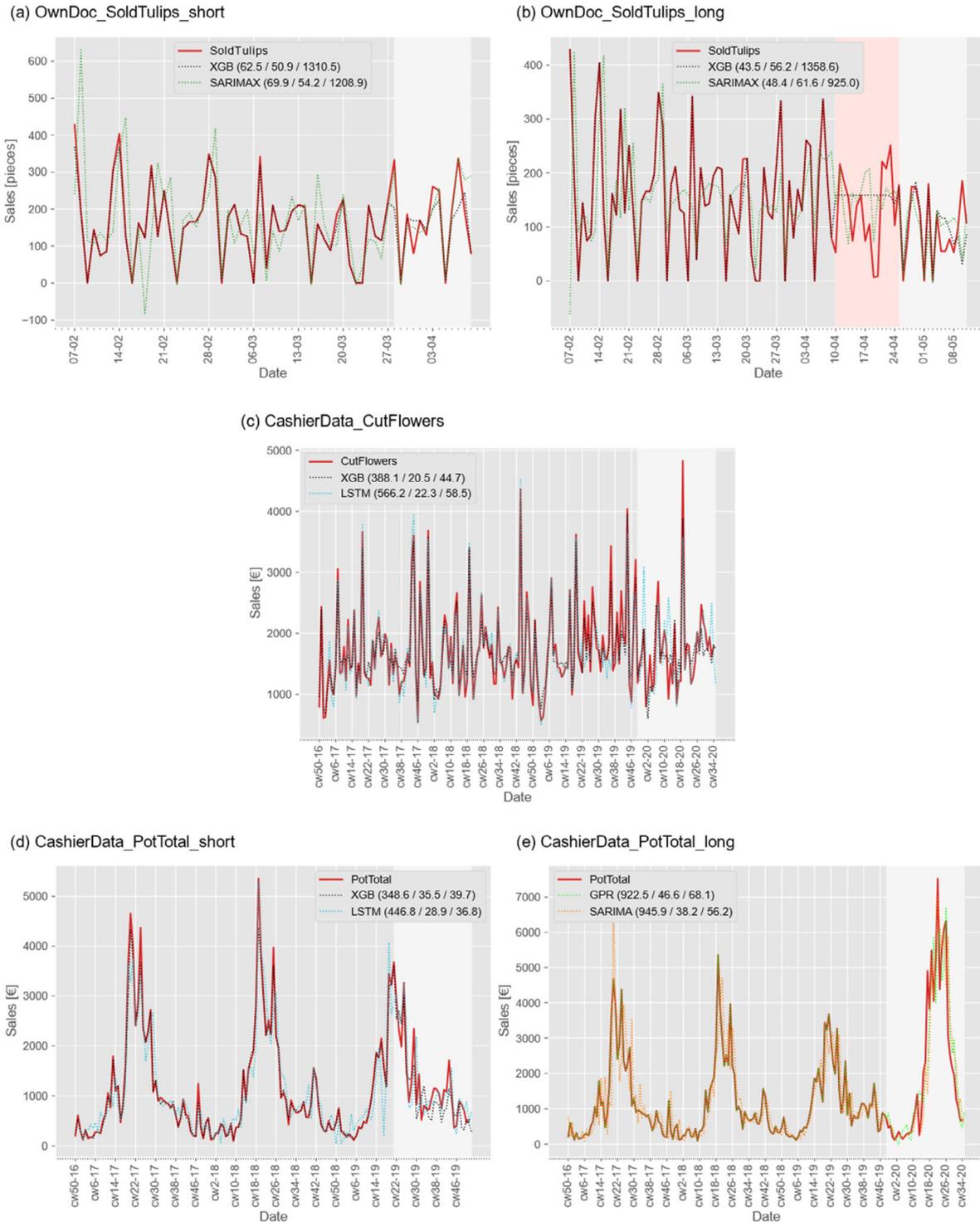


Fig. 4. Example plots of best performing algorithms. Predictions show selected examples of two of the top performers accompanied by the ground truth. The test periods are highlighted in light gray. Missing values for OwnDoc_SoldTulips_long were iteratively imputed and are marked with a red background. In the legends, the evaluation measures RMSE/sMAPE/MAPE are given in brackets. A lower value reflects a better performance. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

to forecast. Nevertheless, overall the plots confirm the promising results of the evaluation measures as most of the curves show a prediction close to the target.

4.3. Feature Analysis

We furthermore analyzed the performance influence of the different featuresets, see Table 3 regarding their structure. For that purpose, we focused on the top performing multivariate approaches SARIMAX,

Table 6

Top results for CashierData after in-depth optimization. A lower value reflects a better performance. Results prior to in-depth optimization are given in brackets in case of an improvement. The best results overall are printed in bold. In-depth optimization was not done for SARIMA on CashierData_CutFlowers and for ANN on the other two variants.

Algorithm	CashierData_CutFlowers			CashierData_PotTotal_short			CashierData_PotTotal_long		
	RMSE	SMAPE	MAPE	RMSE	SMAPE	MAPE	RMSE	SMAPE	MAPE
SARIMA	–	–	–	439.03	34.08 (35.05)	40.13	945.91 (979.10)	36.84 (37.03)	46.41 (56.59)
ANN	475.66 (485.60)	23.31 (23.43)	26.61 (32.19)	–	–	–	–	–	–
LSTM	460.61	21.78 (22.57)	29.65	390.98	28.85	29.71 (31.11)	1037.11 (1064.14)	39.65 (38.96)	47.70
XGB	388.09 (390.96)	19.71 (20.03)	23.98 (24.52)	348.60 (392.31)	29.86 (31.50)	28.59 (30.70)	892.04 (898.56)	34.61 (35.35)	36.69 (42.45)
GPR	493.58	23.61	30.45	421.46 (424.50)	31.00 (31.88)	32.91 (34.42)	922.46	36.03	52.35 (54.20)

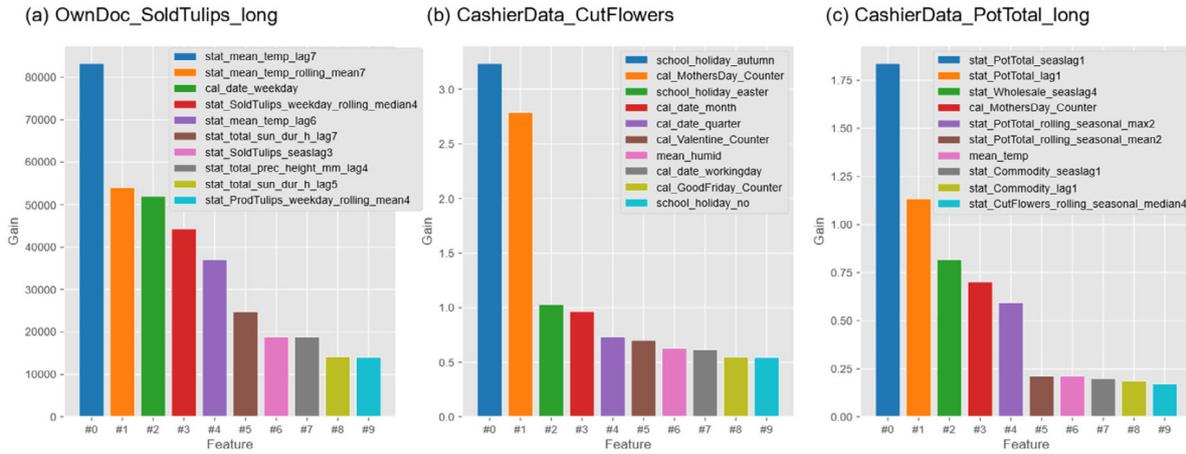


Fig. 5. Ten most important features of XGB according to the gain on selected examples of top performing configurations. The gain of a feature reflects the average improvement across all its usages. A higher value corresponds to a greater importance. (a) OwnDoc_SoldTulips_long with full featureset, (b) CashierData_CutFlowers with sub2 featureset, (c) CashierData_PotTotal_long with full featureset.

LSTM, XGB and GPR. To ensure a fair comparison, we used the predictions prior to the in-depth optimization, as this step was not done for all configurations. We compared the best results achieved with each featureset with respect to all evaluation metrics. An overview of the results (Tables D.3 and D.4) as well as their visualization (Fig. D.1) can be found in Appendix D.2. Based on this, we counted the number of times a featureset led to the best result for each algorithm and dataset, subsequently called a win. Table D.5 in Appendix D.2 summarizes this analysis. Counting all datasets, the win percentages were divided as follows: *sub1* (raw weather and holiday features) 11.7%, *sub2* (focus on calendric features) 41.7%, *sub3* (focus on statistical features) 21.7% and *full* (all features) 25.0%. However, the results varied depending on the dataset and algorithm. We discovered that the *full* featureset was superior for *OwnDoc* with the best result in 12 out of 24 comparisons. For GPR, *sub2*, *sub3* and *full* were on a par, whereas *full* was superior for the other three algorithms. Regarding *CashierData*, *sub2* led to the best result in 20 out of 36 cases (about 55%), followed by *sub3* with seven wins, *sub1* with six and *full* with three. If we exclude the *CashierData_PotTotal_long* dataset influenced by the SARS-CoV-2 pandemic, the advantage of *sub2* was even clearer with 15 wins in 24 comparisons (about 62%). XGB worked best with *sub3*, but *sub2* did so for the other three methods.

Beyond determining the top performing featuresets, we analyzed the feature importance of XGB as this was the best predictor. We present one example for each target variable in Fig. 5. They indicate both a benefit due to external factors such as weather and holiday data as well as derived features such as statistical values. In Fig. 5a, we can see that statistical quantities, both of the external factor weather and past sales numbers, were relevant. The best RMSE for *CashierData_CutFlowers* was achieved with the featureset *sub2* (focus on calendric features),

for which we show the feature importance in Fig. 5b. We found out that calendric features were ranked high, e.g. the counters for the typical high sales days of Mother’s and Valentine’s Day. Furthermore, the school autumn holidays seemed important as they correlate with All Saint’s Day on which religious traditions boost flower sales. Statistical features were the most frequent category for *CashierData_PotTotal_long*. As shown in Fig. 5c, the leading ones were the lagged sales numbers of the previous week and of the same one in the preceding season. Furthermore, delayed information of other product groups improved forecasting. With the mean temperature, a weather property was part of the leaderboard.

4.4. Runtime Comparison

Besides the prediction performance, the computational resource consumption of an algorithm is an important characteristic, especially with a regular model refit due to a changing data distribution in mind. Therefore, we compared the runtimes as described at the end of Section 3.2. The results are visualized in Fig. 6. In summary, the top performer XGB was in both cases in the middle range of the ranking and closer to the efficient methods, which is a further advantage. As we observe in Fig. 6a, ANN and LSTM required the longest runtime for *OwnDoc_SoldTulips_long*, followed by SARIMAX and GPR with about 80% less runtime than LSTM. The MLR approaches showed the lowest values, whereas XGB was comparable to ES and SARIMA. SARIMAX had by far the longest runtime for *CashierData_CutFlowers*, as shown in Fig. 6b, followed by its univariate version and LSTM. ES and the MLR approaches, except ARD, showed low runtimes. The results for GPR and ANN were comparable and XGB was closer to the efficient methods.

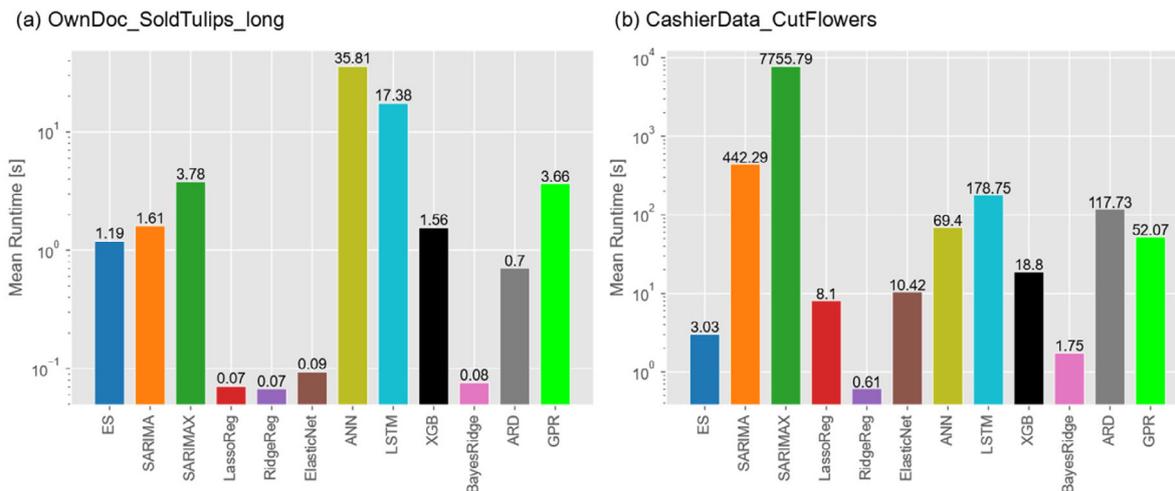


Fig. 6. Mean runtime of one optimization based on 100 random hyperparameter settings for every algorithm. Plots are shown on a logarithmic scale with the exact runtime in seconds above each bar. (a) OwnDoc_SoldTulips_long with sub1 featureset. (b) CashierData_CutFlowers with full featureset.

4.5. Discussion

In this paper, we show a first-time comparison of ML-based and classical forecasting approaches for horticultural sales predictions. Most of the methods outperformed simple baseline comparison partners. Further, our results show a superiority of ML-based methods compared to classical forecasting ones with the former delivering the lowest error in each of the 15 comparisons. The ensemble learner XGB performed best in 14 cases. We assume that its combination of multiple weak learners is beneficial for capturing different effects, which improves the quality and robustness of the final result. LSTM was first in one out of the 15 cases. In summary, LSTM together with GPR were most competitive compared to XGB. The performance of ML algorithms is usually highly influenced by the amount of available data. *CashierData* contains 1359 daily observations, which is about 14 times more than the *OwnDoc* dataset. Nevertheless, XGB also delivered the lowest errors for *OwnDoc*, but the advantage tended to be smaller. With only a few exceptions, the advantage of ML techniques increased on the larger *CashierData* datasets. We assume that, additionally to the dataset size, multivariate patterns and nonlinear relationships in the data, which ML models are rather able to capture, are more pronounced in these datasets that contain several seasons.

One of the research questions of this paper is whether the applied methods cope with the sales increase during the SARS-CoV-2 crisis in *CashierData_PotTotal_long*. We observed that the relative measures sMAPE and MAPE were larger compared to the other two *CashierData* datasets. Beyond that, Fig. 4 suggests that the algorithms were limited in capturing this phenomenon. However, with XGB, an ML approach also performed best for this special case. We assume that the regular refit of the models is beneficial, as recent samples are taken into account. Nevertheless, the samples which comprise this sudden change are only a few compared to the whole train set. Therefore, the algorithms tended to follow the majority of the training instances, which reflect the sales prior to the SARS-CoV-2 pandemic. The comparably good performance of SARIMA might be caused by external factors that correspond to a different data distribution in the training set and consequently could even impede forecasting. But in general, our results indicate that external factors such as weather information as well as derived features, e.g. statistical measures, lead to an improved forecasting performance as all superior methods were multivariate. This additional info is especially useful for ML-based methods. However, regarding the influence of different features, see Section 4.3, a configuration being superior for all algorithms and datasets could not be determined.

Beyond that, the runtime comparison in Section 4.4 shows that SARIMAX – the most competitive classical approach – has poor scalability on larger data- and featuresets. By contrast, XGB was efficient

in both cases. This is an additional advantage of the ML-based method XGB, especially when considering regular refits of the model.

There are several publications on competitions including classical forecasting and ML techniques, based on which a general superiority of one of them cannot be concluded (Bojer & Meldgaard, 2021; Makridakis et al., 2020, 2021). Nevertheless, our conclusion that ML performs better agrees with recent publications on food and tourism demand forecasting (Huber & Stuckenschmidt, 2020; Jiao & Chen, 2019). Several papers have shown a superiority of combined approaches (Bojer & Meldgaard, 2021; Makridakis et al., 2020; Petropoulos et al., 2018). Hence, our results with XGB performing best confirms previous work for the new forecasting domain of horticultural sales. Beyond that, GPR produced competitive results. For this reason and its advantage of providing prediction uncertainties, it might be interesting for horticultural sales prediction and similar domains.

Besides this, we made certain simplifications for our analysis. First, we used historical weather data as external information, but when predicting the demand in production-ready systems, only short-time weather forecasts are available. However, since historical weather forecasts were not retrievable for the whole period, this is a reasonable approximation. Second, the datasets provide sales numbers, which do not fully reflect the potential demand as out-of-stock situations (as an indicator for a higher demand of customers) were not documented. Third, we used holiday and weather data as external information, but there could be additional features which might improve prediction performances (e.g. promotional and communication activities of the company).

This study is the first analysis of horticultural demand forecasting based on typical retail data. So far, it is not obvious if our results generalize, e.g. for companies of the whole value chain. Nevertheless, we provide first insights, which need to be further evaluated before drawing more general conclusions. In accordance with literature, an ensemble method led to the best result. This suggests that combined approaches might also be superior for horticultural demand forecasting. Therefore, the integration of further combined methods, e.g. those including ML-based as well as classical forecasting techniques, is interesting for further research. Beyond that, we employed a computationally expensive approach with a model refit when new samples arrive. This seems reasonable for a first-time adoption with potential changes in data and a productive operation of a forecasting tool with a continuous update of company-specific sales data in mind. Nevertheless, the necessity of model refits should be further examined with multi-step-ahead forecasts and a periodical retraining. The focus of this paper is on horticultural retail sales. It is interesting whether including data of businesses along the whole value chain, e.g. from wholesalers

or producers, might improve forecasts. Furthermore, as horticultural companies are typically small and medium-sized, datasets are heterogeneous and rather small. Thus, novel approaches that combine data of diverse sources and consequently benefit from the information derived from several smaller datasets are an interesting research direction. This might also be beneficial for other domains such as e.g. agricultural producers or small and medium-sized food processing companies. The influence of the SARS-CoV-2 pandemic resulted in a sudden rise in demand for potted plants. Most of the algorithms were not able to deal with this change of the data distribution. Therefore, research based on methods enabling them to cope with such phenomena such as EVARS-GPR could be useful for various applications (Grande et al., 2017; Haselbeck & Grimm, 2021; Liu et al., 2020; Ni et al., 2012).

5. Conclusion

In this paper, we presented a first-time comparative study for horticultural sales predictions with nine state-of-the-art ML and three classical methods. Our study was based on typical horticultural retail data with distinct characteristics, e.g. regarding size and seasonality. We employed a forecasting setup with a regular refit of the model parameters in order to simulate a productive operation of a forecasting system with a continuous data update and a potentially changing data distribution. Our findings show a superiority of ML approaches, especially of the ensemble learner XGB. This advantage increased for larger datasets containing multiple seasons. Beyond that, we experienced a performance increase by including additional features, such as weather or holiday data. Finally, we showed that the top performer XGB is computationally efficient. Based on these first results, a plethora of new research questions arise with a lot of potential for future research. Especially the transfer and verification of these results within the same sector are of general interest and might allow the generalization of our conclusions.

List of abbreviations

ANN — Artificial Neural Network
 AR — Autoregressive part
 ARD — Automatic Relevance Determination
 ARIMA — Autoregressive Integrated Moving Average
 ES — Exponential Smoothing
 GPR — Gaussian Process Regression
 HA — Historical Average
 KNN — K-Nearest-Neighbors
 LSTM — Long Short-Term Memory
 MA — Moving average part
 MAPE — Mean Absolute Percentage Error
 ML — Machine learning
 MLR — Multiple Linear Regression
 PCA — Principal Component Analysis
 ReLU — Rectified Linear
 RMSE — Root Mean Squared Error
 RNN — Recurrent Neural Networks
 RW — Random Walk
 SARIMA — Seasonal Autoregressive Integrated Moving Average
 SARIMAX — Seasonal Autoregressive Integrated Moving Average with external factors
 seasRW — Seasonal Random Walk
 sMAPE — Symmetric Mean Absolute Percentage Error
 XGB — Extreme Gradient Boosting (XGBoost)

CRedit authorship contribution statement

Florian Haselbeck: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing – original draft, Visualization. **Jennifer Killinger:** Validation, Investigation. **Klaus Menrad:** Writing – review & editing, Project administration,

Funding acquisition. **Thomas Hannus:** Writing – review & editing, Project administration, Funding acquisition. **Dominik G. Grimm:** Conceptualization, Methodology, Investigation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All code and data is publicly available on GitHub: <https://github.com/grimmlab/HorticulturalSalesPredictions>.

Acknowledgments

We would like to thank the project funder for supporting our research as well as all project partners. In addition, the authors gratefully acknowledge the compute resources provided by the Leibniz Supercomputing Centre (www.lrz.de). Furthermore, we thank the German Meteorological Service for providing historical weather data.

Funding

The project is supported by funds of the Federal Ministry of Food and Agriculture (BMEL), Germany based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support program [grant number 2818504A18].

Appendix A. Time Series Forecasting Methods

A.1. Classical Forecasting Methods

Classical forecasting methods use chronologically ordered time series data and try to predict future sequences, usually by projecting statistical information recovered from historical data (Hyndman & Athanasopoulos, 2018). In Table 1 of the main document, we provide an overview of the approaches discussed in this paper. Methods can be divided into univariate and multivariate approaches, with the latter one using external information in addition to the time series itself. Two common univariate methods are Exponential Smoothing (ES) and Autoregressive Integrated Moving Average (ARIMA) (Box et al., 2016; Holt, 1957; Winters, 1960). Gardner (2006) gave a detailed overview about the theoretical background of ES and its application (Gardner, 2006). In its simple form, ES is a weighted sum of past observations of a time series with weights decaying exponentially. Therefore, recent values have more influence on the output. More elaborate versions include linear or damped trend as well as seasonal components, which are characteristics that many real-world time series possess. Both components can be formulated in an additive or multiplicative way, depending on the characteristics of the time series.

In contrast, modeling autocorrelations – the correlation between a series and a lagged version of itself – is the key idea of ARIMA. Thereby, autoregressive (AR) and moving average (MA) parts are calculated. The first is a linear combination based on the time series itself, whereas the latter is a weighted sum of past model errors. One of the algorithm's assumptions is stationarity of the input, so trend and seasonality should be removed. One way to achieve this property is differencing prior to optimizing the AR and MA parameters. This step is called the "integrated" part of ARIMA. The determining parameters are thus the degree of differencing and the lags considered when constructing the

AR and MA elements. A common extension, SARIMA, involves seasonal parts. Their formulation is similar to the non-seasonal ones, but uses the observations of previous seasons. Therefore, besides the arguments of the seasonal differencing, AR and MA components, the length of such a period needs to be defined (Hyndman & Athanasopoulos, 2018; Shumway & Stoffer, 2000).

SARIMA is univariate by definition. Consequently, external factors potentially providing useful information are not included. Hence, a multivariate extension called SARIMAX exists. One way to implement it is combining multivariate regression depending on external factors and a SARIMA model, which is driven by the errors of the first one. Usually, both elements are optimized jointly delivering a final output after summation (Arunraj et al., 2016).

A.2. Machine Learning Methods

Sales forecasting can also be formulated as a regression task for ML methods. These data-driven approaches are able to discover complex relationships, which are more likely to be present in multivariate datasets. Therefore, as stated in Table 1 of the main document, we only consider ML approaches in such setups.

For regression tasks, Multiple Linear Regression (MLR) is often used as a baseline. MLR models try to predict a target variable by building a weighted sum of several features and an intercept. Typically, the sum of squared errors between the observed and predicted value is optimized with Ordinary Least Squares during training. With an increasing number of features, this might lead to overfitting. To prevent this, we can use regularized regression approaches by adding a penalty term to the loss function. The goal of this is to learn models which generalize better to unknown data. Common approaches are Ridge, Lasso and Elastic Net Regression (Hoerl & Kennard, 1970; Santosa & Symes, 1986; Tibshirani, 1996; Zou & Hastie, 2005). The first one – also called L2-regularization – introduces a quadratic term penalizing the weights' size. This usually leads to a lower influence of correlated features by distributing the weight among them. However, Ridge Regression does not push the weights of irrelevant features exactly to zero, but reduces their influence. Lasso Regression uses L1-norm (the absolute value of the weights instead of the quadratic ones) as penalty term. This leads to the effect of forcing the weights of unimportant features to zero, which can be seen as an automatic feature selection process (James et al., 2017). Elastic Net Regression combines both L1- and L2-regularization with an additional hyperparameter controlling the influence of each one. Consequently, the variable selection effect of Lasso as well as the grouping mechanism for correlated features of Ridge are included (Zou & Hastie, 2005).

Moreover, we used ML methods because they can cover nonlinearities. A common approach in time series prediction, as shown by Zhang et al. (1998), are Artificial Neural Networks (ANN) (Rosenblatt, 1958; Zhang et al., 1998). Their advantages, such as the ability to model nonlinear and complex relationships in a data-driven way, make them an appropriate alternative. As we work with time series data, it seems obvious to use Recurrent Neural Networks (RNN), which are able to capture temporal dependencies as they – intuitively speaking – possess a memory of previous states. Hewamalage et al. presented an overview of RNNs in time series forecasting (Hewamalage et al., 2021). They concluded that RNNs are competitive and that Long Short-Term Memory Networks (LSTM) show the best performance. Due to their memory and gating mechanisms, LSTMs can capture long-term dependencies and prevent the vanishing gradient problem (Hochreiter & Schmidhuber, 1997). Based on these scientific findings, it is common to design recurrent networks for time series forecasting with these cells. However, the flexibility of ANNs and RNNs might lead to overfitting. To prevent this, regularization techniques can be employed. A common regularization method is dropout. With this approach, a specified share of neurons are randomly ignored during each training iteration, leaving a reduced network (Srivastava et al., 2014). Another regularization

procedure is early stopping. Thereby, the loss on a validation set (independent data) is monitored during training and if there is no improvement for a certain period, the optimization gets terminated (Bishop, 2009).

Many publications and forecasting competitions indicate a superiority of combined methods, e.g. achieved via ensemble learners and mechanisms such as Bagging or Boosting (Bojer & Meldgaard, 2021; Petropoulos et al., 2018). XGB is an ensemble technique that uses gradient boosted regression trees as weak learners. With this approach, decision trees are sequentially added in a greedy manner based on the gradient of the loss function in order to correct the errors made by the current ensemble. This is in contrast to algorithms using Bagging, e.g. Random Forests, which train several weak learners in parallel on bootstrapped samples. Furthermore, Boosting algorithms showed a good performance in forecasting competitions (Bojer & Meldgaard, 2021). XGB is a sparse-aware and computationally efficient implementation of this technique (Chen & Guestrin, 2016).

Besides the frequentist approach, for which we view probability as the relative frequency of an event in an experiment, we can interpret probability in a Bayesian perspective. Thereby, we define a prior belief before considering evidence as we assume that certain hypotheses are more plausible than others. Then, we take evidence into account, e.g. observations in an experiment, leading to a posterior belief. This is regularized by the prior, as we usually do not completely reject our previous belief. For ML methods, we can define the prior as a probability distribution over the model parameters and calculate their posterior based on observations in a dataset. Therefore, in contrast to a frequentist viewpoint with fixed parameter values, probability distributions over the model parameters reflecting their uncertainty are estimated (Bishop, 2009).

MLR can be defined in a Bayesian perspective as follows: $p(y|x, \mathbf{w}) = \mathcal{N}(y|x^T \mathbf{w}, \sigma^2)$, where the target variable y follows a Gaussian distribution with the mean $x^T \mathbf{w}$ (determined by the predictors x and the weights \mathbf{w}) and the variance σ^2 . If the weights of the model are constrained to a zero-mean Gaussian prior, the solution is equivalent to Ridge Regression and therefore called Bayesian Ridge Regression (BayesRidge). Automatic Relevance Determination (ARD), also known as Sparse Bayesian Learning or Relevance Vector Machine, is related to it, but introduces an individual variance for each weight. If the variance of a feature is low, the corresponding weight is likely to be close to zero and can be pruned leading to sparser solutions (Bishop, 2009; James et al., 2017; Tipping, 2001).

With regard to the practical use of demand forecasts, the uncertainty of a prediction value seems profitable. Providing those by its definition is a main advantage of the nonparametric Bayesian method Gaussian Process Regression (GPR) (Williams & Rasmussen, 1996). To explain it, it makes sense to go back to the linear model. This is defined as

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}, \quad y = f(\mathbf{x}) + \epsilon,$$

with \mathbf{x} being the input vector, \mathbf{w} the vector of weights, the function value $f(\mathbf{x})$ and observed target value y with additive noise ϵ assumed to follow a zero-mean Gaussian. Combined with the independence assumption of the observation values, we get the likelihood, which reflects how probable the observed target values y are for the different inputs \mathbf{X} and weights \mathbf{w} :

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^j p(y_i|x_i, \mathbf{w})$$

As usual for a Bayesian formulation, we define a prior over the weights, for which we again choose a zero-mean Gaussian. With the defined prior and the likelihood based on the observed data, we can use Bayes' rule to get the posterior of the weights given the data:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})}$$

This is also called the maximum a posteriori estimate which – provided the data – delivers the most likely set of weights \mathbf{w} . As $p(\mathbf{y}|\mathbf{X})$ is independent of \mathbf{w} , we can reformulate this equation expressing the posterior distribution with a Gaussian defined by a mean and covariance matrix:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{w}}, \mathbf{A}^{-1})$$

During inference, we marginalize out \mathbf{w} and as a result take the average based on all possible \mathbf{w} weighted by their posterior probability:

$$\begin{aligned} p(y_{T_{est}}|\mathbf{x}_{T_{est}}, \mathbf{X}, \mathbf{y}) &= \int p(y_{T_{est}}|\mathbf{x}_{T_{est}}, \mathbf{w}) p(\mathbf{w}|\mathbf{X}, \mathbf{y}) d\mathbf{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma^2} \mathbf{x}_{T_{est}}^T \mathbf{A}^{-1} \mathbf{X} \mathbf{y}, \mathbf{x}_{T_{est}}^T \mathbf{A}^{-1} \mathbf{x}_{T_{est}}\right) \end{aligned}$$

Therefore, we do not only get an output value, but also an uncertainty. So far, we reached the Bayesian formulation of linear regression with its limited expressiveness. To overcome this constraint to linearity, we can project the inputs into a high-dimensional space and apply the linear concept there. This transformation can be accomplished using basis functions $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^i$ leading to the following model with i weights \mathbf{w} :

$$f(\mathbf{x}) = \phi(\mathbf{x})^T \mathbf{w}$$

Conducting the same derivation as shown above results in a similar outcome:

$$\begin{aligned} p(y_{T_{est}}|\mathbf{x}_{T_{est}}, \mathbf{X}, \mathbf{y}) &= \mathcal{N}\left(\frac{1}{\sigma^2} \phi(\mathbf{x}_{T_{est}})^T \mathbf{A}^{-1} \Phi(\mathbf{X}) \mathbf{y}, \right. \\ &\quad \left. \phi(\mathbf{x}_{T_{est}})^T \mathbf{A}^{-1} \phi(\mathbf{x}_{T_{est}})\right) \end{aligned}$$

The need of inverting the $i \times i$ matrix \mathbf{A} possibly causes computational problems if the dimension of the feature space i becomes large. To solve this, we can reformulate the above using the so-called “kernel trick”. This leads to the formulation of a Gaussian Process, which is completely specified by its mean and covariance function:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \quad k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]$$

$k(\mathbf{x}, \mathbf{x}')$ consists of the covariance value between any two sample points \mathbf{x} and \mathbf{x}' resulting in a $n \times n$ matrix for a training set length of n . The assumption is that the similarity between samples reflects the strength of the correlation between their corresponding target values. Therefore, the function evaluation can be seen as a draw from a multivariate Gaussian distribution defined by $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$. Thus, Gaussian Processes are a distribution over functions rather than parameters, in contrast to Bayesian linear regression. For simplicity, the mean function is often set to zero or a constant value. There are many forms of kernel functions, which need to fulfill certain properties, e.g. being positive semidefinite and symmetric. Furthermore, they can be combined, e.g. by summation or multiplication. The choice of the covariance kernel function is a determining configuration of GPR and its parameters need to be optimized during training (Rasmussen & Williams, 2008; Roberts et al., 2013).

Appendix B. Hyperparameters

Subsequently, the hyperparameters and transformations we used for all algorithms can be found. For every one, we randomly sampled hyperparameter combinations for optimization. All tables show the values used for the comparison before the best working solutions were analyzed in an in-depth optimization. Beyond that, we used different imputation strategies, from which KNN and Iterative Imputation needed to be parametrized. We empirically determined $k = 10$ for KNN and a maximum number of iterations of 100 for Iterative Imputation.

B.1. Exponential Smoothing (ES)

See Table B.1.

Table B.1
Hyperparameters Exponential Smoothing.

Parameter	Values	Notes
seasonal_periods	[7, 14, 21]	Only for <i>SoldTulips</i> : period for seasonality
trend	['add', None]	Trend component
damp	[False, True]	Damp trend
seasonality	['add', 'mul', None]	Seasonal component
remove_bias	[False, True]	Force average residual to zero
use_brute	[False, True]	Search starting values using brute force or use a naive set
transf	[False, 'log', 'pw']	Transformation method

B.2. Seasonal Autoregressive Integrated Moving Average (with external factors) (SARIMA(X))

See Table B.2.

Table B.2
Hyperparameters Seasonal Autoregressive Integrated Moving Average (with external factors).

Parameter	Values	Notes
p/q/P/Q	[0, 1, 2, 3]	Non-seasonal/seasonal lag for AR and MA component
d/D	[0, 1]	Non-seasonal/seasonal differencing
exog	[False, True]	Use exogeneous variables (SARIMA or SARIMAX)
transf	[False, 'log', 'pw']	Transformation method

B.3. Regularized Regression (LassoReg, RidgeReg, ElasticNet)

See Table B.3.

Table B.3
Hyperparameters Regularized Regression.

Parameter	Values	Notes
normalize	[False, True]	Normalization prior to regression
alpha	[10** x for x in range(-5, 5)]	Constant multiplying the regularization penalty term
l1_ratio	Np.arange(0.1, 1, 0.1)	Only for Elastic Net: ratio of L1-regularization

B.4. Artificial Neural Network (ANN)

See Table B.4.

Table B.4
Hyperparameters Artificial Neural Network.

Parameter	Values	Notes
activation_function	ReLU	Activation function after each neuron
optimizer	Adam	Optimizer for hyperparameter optimization
dropout_rate	[0.0, 0.5]	Rate for dropout layer
batch_size	[4, 8, 16, 32]	Batch size for training
learning_rate	[1e-4, 1e-3, 1e-2, 1e-1]	Learning rate for Adam optimizer
min_val_loss_improvement	[100, 1000]	Minimum validation loss improvement for early stopping
max_epochs_wo_improvement	[20, 50, 100]	Maximum epochs without improvement for early stopping
n_hidden	[10, 20, 50, 100]	Number of hidden neurons of input layer
num_hidden_layer	[1, 2, 3]	Number of hidden layer (including input layer)

B.5. Long Short-Term Memory Network (LSTM)

See Table B.5.

Table B.5
Hyperparameters Long Short-Term Memory Network.

Parameter	Values	Notes
activation_function	ReLu	Activation function after each neuron
optimizer	Adam	Optimizer for hyperparameter optimization
dropout_rate	[0.0, 0.5]	Rate for dropout layer
batch_size	[4, 8, 16, 32]	Batch size for training
learning_rate	[1e-3, 1e-2, 1e-1]	Learning rate for Adam optimizer
min_val_loss_improvement	[0.01, 0.1]	Minimum validation loss improvement for early stopping
max_epochs_wo_improvement	[20, 50, 100]	Maximum epochs without improvement for early stopping
lstm_hidden_dim	[5, 10, 50]	Dimensionality of the hidden state
lstm_num_layers	[1, 2]	Number of recurrent layers
seq_length	[1, 4, seasonal_periods]	Sequence length for input

B.6. Extreme Gradient Boosting (XGB)

See [Table B.6](#).

Table B.6
Hyperparameters Extreme Gradient Boosting.

Parameter	Values	Notes
learning_rate	[0.05, 0.1, 0.3]	Learning rate / eta
max_depth	[3, 5, 10]	Maximum tree depth of a base learner
subsample	[0.3, 0.7, 1]	Subsample ratio of a training instance
n_estimators	[10, 100, 1000]	Number of gradient boosted trees
gamma	[0, 1, 10]	Minimum loss reduction for a partition on a leaf node
alpha	[0, 0.1, 1, 10]	Weight for L1-regularization
reg_lambda	[0, 0.1, 1, 10]	Weight for L2-regularization

B.7. Bayesian Regression (BayesRidge, ARD)

See [Table B.7](#).

Table B.7
Hyperparameters Bayesian Regression.

Parameter	Values	Notes
normalize	[False, True]	Normalization prior to regression
alpha_1	[10** x for x in range(-6, 1)]	Shape parameter for Gamma distribution over alpha
alpha_2	[10** x for x in range(-6, -4)]	Rate parameter for Gamma distribution over alpha
lambda_1	[10** x for x in range(-6, 1)]	Shape parameter for Gamma distribution over lambda
lambda_2	[10** x for x in range(-6, 1)]	Rate parameter for Gamma distribution over lambda
threshold_lambda	[10** x for x in range(2, 6)]	Only for ARD: threshold for pruning weights

B.8. Gaussian Process Regression (GPR)

The determining parameter for GPR is the kernel function. We used different kernel functions as well as combinations of up to three kernels (sums and products). The base kernels we used can be found in the following list, which are formulated according to the used library GPflow ([Matthews et al., 2017](#)):

- *SquaredExponential()*
- *Matern52()*
- *White()*
- *RationalQuadratic()*

- *Polynomial()*,
- *Periodic(kernels=SquaredExponential(), period=seasonal_periods)*,
- *Periodic(kernels=Matern52(), period=seasonal_periods)*,
- *Periodic(kernels=RationalQuadratic(), period=seasonal_periods)*

Furthermore, we used the subsequent hyperparameters.

Table B.8
Hyperparameters Gaussian Process Regression.

Parameter	Values	Notes
mean_function	[None, Constant()]	Mean function used for Gaussian distribution
noise_variance	[0.01, 1, 10, 100]	Noise variance added to diagonal of kernel matrix
standardize_x	[False, True]	Standardize features
standardize_y	[False, True]	Standardize target variable

Appendix C. Results Overview Tables

The tables, which reflect an overview of the results of the comparison performed in this paper, are published in supplementary fashion as “*Supplementary 1 - Results Overview Tables.xlsx*”. This file contains several tabs, of which each one presents the results of one dataset. These show an overview of the best optimization results with respect to the three evaluation metrics of every algorithm for each experimental setting. Every table is grouped in “*Univariate Methods and Baselines*” as well as “*Multivariate Methods*”. Besides that, every chart is structured by the used featureset, imputation strategy and dimensionality reduction. If a specific combination was not included in the comparison, the related cell is filled in gray. The best results are printed in bold, and the cells are highlighted with a green background.

Appendix D. Top Results Tables

The following tables show top results for several setups such as the best performance of every algorithm on each dataset over all featuresets and preprocessing methods, see [Tables D.1](#) and [D.2](#). As in all subsequent tables, the best results are printed in bold and highlighted in green. The results based on each feature- and dataset for the multivariate top performers are summarized in [Appendix D.2](#). The evaluation is based on the results before the in-depth optimization, as this second optimization was not conducted for all configurations.

D.1. Top Results Overall

See [Tables D.1](#) and [D.2](#).

Table D.1
Top results overall for the OwnDoc datasets.

algorithm	OwnDoc					
	OwnDoc_SoldTulips_short			OwnDoc_SoldTulips_long		
	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE
ES	77.63	61.84	116.77	62.17	56.61	241.40
SARIMA	83.31	58.19	81.52	52.02	55.83	3675.31
SARIMAX	58.06	52.58	85.75	50.35	52.37	231.52
LassoReg	84.65	58.86	1615.95	59.32	52.50	2354.94
RidgeReg	84.73	60.29	1978.83	58.47	54.12	2970.38
ElasticNet	84.73	60.16	1569.83	57.27	52.64	2888.18
ANN	77.48	56.06	115.73	48.93	53.09	223.10
LSTM	73.66	52.49	707.61	56.06	50.04	2884.77
XGB	57.11	50.89	35.98	43.52	48.65	54.22
BayesRidge	96.88	62.62	9153.80	60.25	53.28	6221.16
ARD	79.89	57.41	162.77	60.35	53.68	1194.57
GPR	68.14	52.27	67.99	56.05	52.43	65.84

Table D.2
Top results overall for the CashierData datasets.

	CashierData								
	CashierData_CutFlowers			CashierData_PofTotal_short			CashierData_PofTotal_long		
algorithm	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE
ES	591.89	28.29	58.18	513.26	36.87	37.74	1066.36	37.63	57.12
SARIMA	608.13	27.50	57.23	439.03	35.05	40.13	979.10	37.03	56.59
SARIMAX	578.15	25.31	58.53	531.76	38.33	42.29	1063.66	37.52	50.57
LassoReg	563.27	25.53	58.12	455.28	35.19	43.05	1067.88	38.53	64.61
RidgeReg	566.51	25.66	56.98	482.28	37.14	42.68	1136.46	44.09	64.42
ElasticNet	565.27	25.63	56.43	475.17	36.16	41.85	1063.28	36.85	64.65
ANN	485.60	23.43	32.19	507.19	39.40	41.04	1031.24	44.36	42.82
LSTM	460.61	22.57	29.65	390.98	28.85	31.11	1064.14	39.65	48.90
XGB	390.96	20.03	24.52	392.31	31.50	30.70	898.56	35.35	42.45
BayesRidge	572.96	26.06	57.88	534.58	40.73	39.59	1062.94	38.53	66.32
ARD	550.98	24.91	55.24	460.64	34.94	42.93	1026.61	36.01	57.27
GPR	493.58	23.61	30.45	424.50	31.88	34.42	922.46	36.03	54.20

Table D.3
Top results per featureset of the OwnDoc datasets.

		OwnDoc					
		OwnDoc_SoldTulips_short			OwnDoc_SoldTulips_long		
algorithm	featureset	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE
SARIMAX	sub1	74.39	57.46	222.57	53.40	52.37	585.06
	sub2	64.27	52.74	138.81	50.35	55.23	231.52
	sub3	60.37	54.98	110.87	71.98	61.57	237.38
	full	58.06	52.58	85.75	55.73	65.63	254.73
LSTM	sub1	86.28	56.27	4906.27	58.24	54.57	8237.34
	sub2	88.90	57.63	4450.57	58.74	54.88	6398.16
	sub3	73.66	52.49	914.54	56.21	50.87	5481.53
	full	83.90	55.25	707.61	56.06	50.04	2884.77
XGB	sub1	120.97	65.83	5936.07	47.90	54.22	4745.91
	sub2	67.67	57.19	43.90	48.15	54.03	54.22
	sub3	57.11	53.24	37.95	48.82	48.65	1221.83
	full	62.52	50.89	35.98	43.52	52.36	1346.26
GPR	sub1	96.01	59.95	79.00	58.14	52.82	74.03
	sub2	68.14	57.91	79.02	56.05	54.88	73.91
	sub3	76.63	55.79	67.99	63.10	52.61	65.84
	full	75.02	52.27	70.54	56.56	52.43	69.28

D.2. Featureset-specific Top Results for SARIMAX, LSTM, XGB and GPR

To visualize the top results per featureset, we derived the stacked bar plots shown in Fig. D.1 based on the information presented in Tables D.3 and D.4. Furthermore, Table D.5 summarizes the featuresets leading to the best result for each dataset.

Appendix E. Supplementary Data

Supplementary material related to this article can be found online at <https://doi.org/10.1016/j.mlwa.2021.100239>.

Table D.4
Top results per featureset of the CashierData datasets.

		CashierData								
		CashierData_CutFlowers			CashierData_PofTotal_short			CashierData_PofTotal_long		
algorithm	featureset	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE	RMSE	sMAPE	MAPE
SARIMAX	sub1	706.69	30.93	60.95	587.51	38.33	50.16	1084.95	37.52	54.97
	sub2	578.15	25.31	58.53	531.76	39.05	42.29	1063.66	41.03	50.57
	sub3	679.88	30.93	60.95	587.51	39.29	50.32	1086.67	37.64	54.47
	full	585.82	26.13	63.34	545.43	39.74	44.81	1095.59	41.45	51.04
LSTM	sub1	460.61	23.74	34.15	401.52	28.85	34.59	1100.44	41.99	52.26
	sub2	487.53	22.57	37.74	391.28	30.69	34.81	1064.14	39.65	48.90
	sub3	577.10	25.16	34.22	407.37	29.95	31.11	1143.76	42.15	53.06
	full	511.28	24.61	29.65	390.98	30.83	31.77	1116.59	41.71	50.47
XGB	sub1	582.96	21.95	24.52	512.48	37.92	37.79	1494.98	47.08	51.60
	sub2	390.96	20.03	31.52	428.25	33.97	30.70	1360.75	40.91	44.22
	sub3	640.82	23.68	46.30	392.31	31.50	34.09	898.56	35.35	42.45
	full	467.24	22.40	45.71	430.47	33.19	35.37	974.64	36.18	42.63
GPR	sub1	683.74	27.35	45.72	543.26	40.44	44.74	1657.81	56.60	67.56
	sub2	493.58	23.61	30.45	424.50	31.88	34.42	1358.78	54.06	56.27
	sub3	655.74	26.43	51.25	448.09	36.00	39.30	978.78	36.36	54.20
	full	625.42	26.79	51.97	458.47	38.05	37.49	922.46	36.03	55.08

Table D.5
Win counts for each featureset on every algorithm and dataset. A win is defined as a featureset leading to the best result for one of the evaluation metrics. The wins of every featureset are summed up. The leading featuresets for every algorithm on OwnDoc and CashierData are printed in bold and highlighted in green.

		OwnDoc			CashierData			
		SoldTulips_short	SoldTulips_long	summary	CutFlowers	PofTotal_short	PofTotal_long	summary
algorithm	featureset	wins	wins	wins	wins	wins	wins	wins
SARIMAX	sub1	0	1	1	0	1	1	2
	sub2	0	2	2	3	2	2	7
	sub3	0	0	0	0	0	0	0
	full	3	0	3	0	0	0	0
LSTM	sub1	0	0	0	2	1	0	3
	sub2	0	0	0	1	0	3	4
	sub3	2	0	2	0	1	0	1
	full	1	3	4	0	1	0	1
XGB	sub1	0	0	0	1	0	0	1
	sub2	0	1	1	2	1	0	3
	sub3	1	1	2	0	2	3	5
	full	2	1	3	0	0	0	0
GPR	sub1	0	0	0	0	0	0	0
	sub2	1	1	2	3	3	0	6
	sub3	1	1	2	0	0	1	1
	full	1	1	2	0	0	2	2

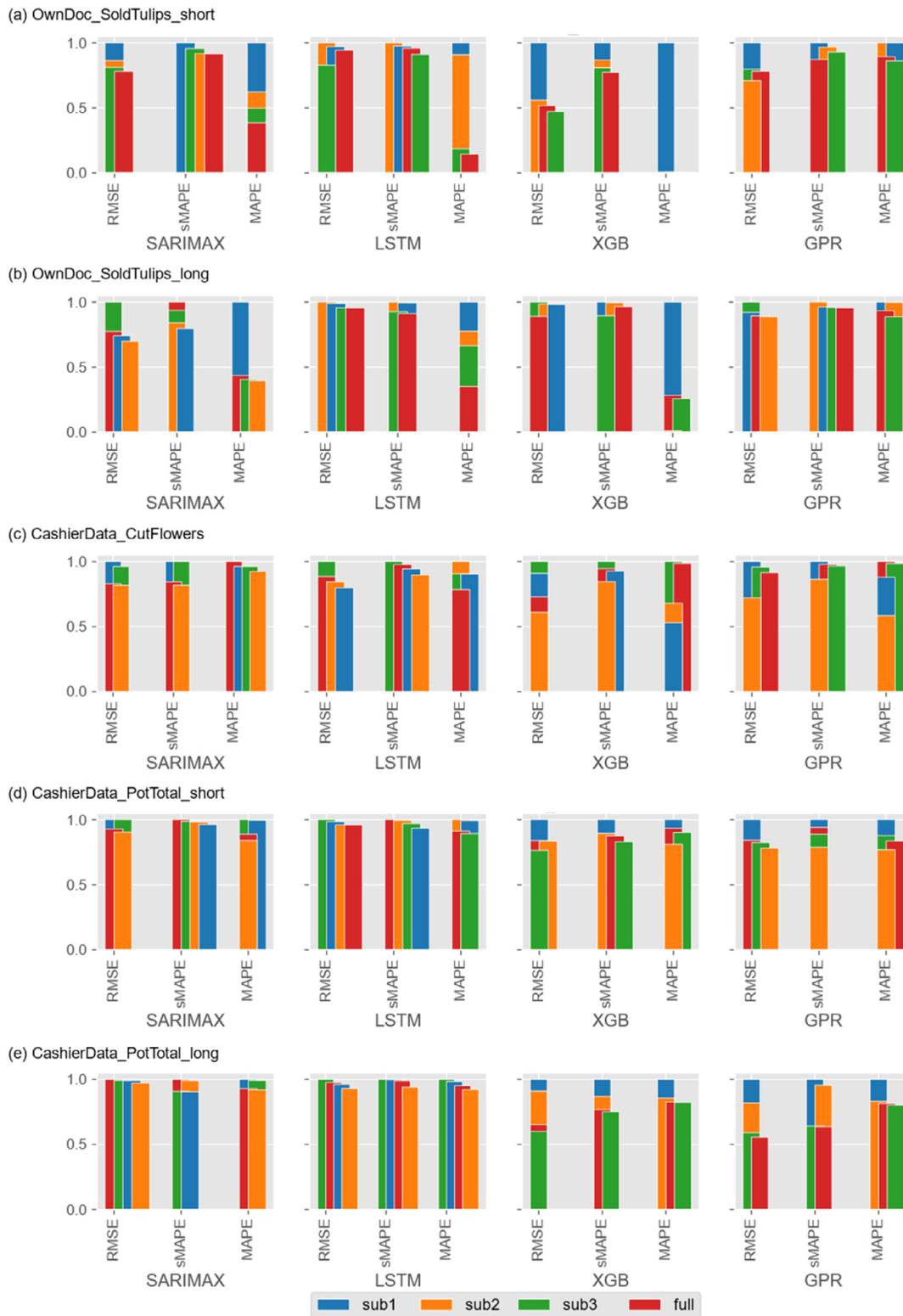


Fig. D.1. Top results per featureset for SARIMAX, LSTM, XGB, GPR (columns) on every dataset (rows). The values achieved with every featureset are scaled by the worst result. Smaller bars represent a better outcome with similar ones plotted with a small offset on the x-axis to ensure visibility. Bars might not be visible in case of a major difference to the worst result, e.g. XGB's MAPE for OwnDoc_SoldTulips_short.

References

- Arunraj, N. S., & Ahrens, D. (2015). A hybrid seasonal autoregressive integrated moving average and quantile regression for daily food sales forecasting. *International Journal of Production Economics*, 170, 321–335. <http://dx.doi.org/10.1016/j.ijpe.2015.09.039>.
- Arunraj, N. S., Ahrens, D., & Fernandes, M. (2016). Application of SARIMAX model to forecast daily sales in food retail industry. *International Journal of Operations Research and Information Systems*, 7(2), 1–21. <http://dx.doi.org/10.4018/IJORIS.2016040101>.
- Arunraj, N. S., Ahrens, D., Fernandes, M., & Müller, M. (2014). Time series sales forecasting to reduce food waste in retail industry. *Rotterdam*, <http://dx.doi.org/10.13140/RG.2.1.4829.1607>.
- Athanasopoulos, G., Hyndman, R. J., Song, H., & Wu, D. C. (2011). The tourism forecasting competition. *International Journal of Forecasting*, 27(3), 822–844. <http://dx.doi.org/10.1016/j.ijforecast.2010.04.009>.
- Behe, B. K., Getter, K. L., & Yue, C. (2012). Should you blame the weather? The influence of weather parameters, month, and day of the week on spring herbaceous plant sales in the U.S. midwest. *HortScience*, 47(1), 71–73. <http://dx.doi.org/10.21273/HORTSCI.47.1.71>.
- Bishop, C. M. (2009). *Pattern Recognition and Machine Learning* (8th ed.). Springer.
- Bojer, C. S., & Meldgaard, J. P. (2021). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2), 587–603. <http://dx.doi.org/10.1016/j.ijforecast.2020.07.007>.
- Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2016). *Wiley Series in Probability and Statistics, Time Series Analysis: Forecasting and Control* (5th ed.). Wiley.
- Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. Smola, C. Aggarwal, D. Shen, & R. Rastogi (Eds.), *KDD 2016: 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM Association for Computing Machinery, <http://dx.doi.org/10.1145/2939672.2939785>.
- Duan, Y., Li, G., Tien, J. M., & Huo, J. (2012). Inventory models for perishable items with inventory level dependent demand rate. *Applied Mathematical Modelling*, 36(10), 5015–5028. <http://dx.doi.org/10.1016/j.apm.2011.12.039>.
- Gardner, E. S. (2006). Exponential smoothing: The state of the art—Part II. *International Journal of Forecasting*, 22(4), 637–666. <http://dx.doi.org/10.1016/j.ijforecast.2006.03.005>.
- Grande, R. C., Walsh, T. J., Chowdhary, G., Ferguson, S., & How, J. P. (2017). Online regression for data with changepoints using Gaussian processes and reusable models. *IEEE Transactions on Neural Networks and Learning Systems*, 28(9), 2115–2128. <http://dx.doi.org/10.1109/TNNLS.2016.2574565>.
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Del Río, J. F., Wiebe, M., Peterson, P., Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <http://dx.doi.org/10.1038/s41586-020-2649-2>.
- Haselbeck, F., & Grimm, D. G. (2021). Evars-GPR: Event-triggered augmented refitting of Gaussian process regression for seasonal data. In S. Edelkamp, R. Möller, & E. Rueckert (Eds.), *Lecture Notes in Computer Science: vol. 12873, Ki 2021: 44th German Conference on AI, Virtual Event, September 27 - October 1, 2021, Proceedings (Vol. 12873)* (pp. 135–157). Springer, http://dx.doi.org/10.1007/978-3-030-87626-5_11.
- Hewamalage, H., Bergmeir, C., & Bandara, K. (2021). Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1), <http://dx.doi.org/10.1016/j.ijforecast.2020.06.008>.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55–67. <http://dx.doi.org/10.1080/00401706.1970.10488634>.
- Holt, C. C. (1957). *Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages*. Office of Naval Research Memorandum.
- Hong, T., Xie, J., & Black, J. (2019). Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. *International Journal of Forecasting*, 35(4), 1389–1399. <http://dx.doi.org/10.1016/j.ijforecast.2019.02.006>.
- Huber, J., & Stuckenschmidt, H. (2020). Daily retail demand forecasting using machine learning with emphasis on calendric special days. *International Journal of Forecasting*, 36(4), 1420–1438. <http://dx.doi.org/10.1016/j.ijforecast.2020.02.005>.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90–95. <http://dx.doi.org/10.1109/MCSE.2007.55>.
- Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and Practice (2. Auflage)*. OTexts.
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. <http://dx.doi.org/10.1016/j.ijforecast.2006.03.001>.
- Ivanov, D., Tsipoulanidis, A., & Schönberger, J. (2019). Demand forecasting. In D. Ivanov, A. Tsipoulanidis, & J. Schönberger (Eds.), *Springer Texts in Business and Economics. Global Supply Chain and Operations Management: A Decision-Oriented Introduction to the Creation of Value* (pp. 319–333). Springer, http://dx.doi.org/10.1007/978-3-319-94313-8_11.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2017). *An Introduction to Statistical Learning: With Applications in R* (8th ed.). Springer, <http://dx.doi.org/10.1007/978-1-4614-7138-7>.
- Jiao, E. X., & Chen, J. L. (2019). Tourism forecasting: A review of methodological developments over the last decade. *Tourism Economics*, 25(3), 469–492. <http://dx.doi.org/10.1177/1354816618812588>.
- Jolliffe, Ian T., & Cadima, Jorge (2016). Principal component analysis: A review and recent developments. *Phil. Trans. Ser. A Math. Phys. Eng. Sci.*, 374(2065), <http://dx.doi.org/10.1098/rsta.2015.0202>.
- Kolassa, Stephan (2021). Commentary on the M5 forecasting competition. *International Journal of Forecasting*, <http://dx.doi.org/10.1016/j.ijforecast.2021.08.006>, Advance online publication.
- Liu, X., & Ichise, R. (2017). Food sales prediction with meteorological data — A case study of a Japanese chain supermarket. In Y. Tan, H. Takagi, & Y. Shi (Eds.), *Lecture Notes in Computer Science: vol. 10387, Data Mining and Big Data: Second International Conference, DMBD 2017, Fukuoka, Japan, July 27 - August 1, 2017, Proceedings (Vol. 10387)* (pp. 93–104). Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-61845-6_10.
- Liu, B., Qi, Y., & Chen, K.-J. (2020). Sequential online prediction in the presence of outliers and change points: An instant temporal structure learning approach. *Neurocomputing*, 413, 240–258. <http://dx.doi.org/10.1016/j.neucom.2020.07.011>.
- Lloyd, J. R. (2014). Gefcom2012 hierarchical load forecasting: Gradient boosting machines and Gaussian processes. *International Journal of Forecasting*, 30(2), 369–374. <http://dx.doi.org/10.1016/j.ijforecast.2013.07.002>.
- Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., & Winkler, R. (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2), 111–153. <http://dx.doi.org/10.1002/for.3980010202>.
- Makridakis, S., Chatfield, C., Hibon, M., Lawrence, M., Mills, T., Ord, K., & Simmons, L. F. (1993). The M2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting*, 9(1), 5–22. [http://dx.doi.org/10.1016/0169-2070\(93\)90044](http://dx.doi.org/10.1016/0169-2070(93)90044).
- Makridakis, S., & Hibon, M. (2000). The M3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4), 451–476. [http://dx.doi.org/10.1016/S0169-2070\(00\)00057-1](http://dx.doi.org/10.1016/S0169-2070(00)00057-1).
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4), 802–808. <http://dx.doi.org/10.1016/j.ijforecast.2018.06.001>.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100, 000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. <http://dx.doi.org/10.1016/j.ijforecast.2019.04.014>.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2021). The M5 competition: Background, organization, and implementation. *International Journal of Forecasting*, <http://dx.doi.org/10.1016/j.ijforecast.2021.07.007>, Advance online publication.
- Matthews, A. G. de G., van der Wilk, M., Nickson, T., Fujii, K., Boukouvalas, A., León-Villagrà, P., & Hensman, J. (2017). Gpflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40), 1299–1304.
- McKinney, W. (2010). Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference* (pp. 56–61). SciPy, <http://dx.doi.org/10.25080/Majora-92bf1922-00a>.
- Ni, W., Tan, S. K., Ng, W. J., & Brown, S. D. (2012). Moving-window GPR for nonlinear dynamic system modeling with dual updating and dual preprocessing. *Industrial and Engineering Chemistry Research*, 51(18), 6416–6428. <http://dx.doi.org/10.1021/ie201898a>.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems, Vol. 32* (pp. 8024–8035). Curran Associates, Inc.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Petropoulos, F., Hyndman, R. J., & Bergmeir, C. (2018). Exploring the sources of uncertainty: Why does bagging for time series forecasting work? *European Journal of Operational Research*, 268(2), 545–554. <http://dx.doi.org/10.1016/j.ejor.2018.01.045>.
- Priyadarshi, R., Panigrahi, A., Routroy, S., & Garg, G. K. (2019). Demand forecasting at retail stage for selected vegetables: a performance analysis. *Journal of Modelling in Management*, 14(4), 1042–1063. <http://dx.doi.org/10.1108/JM2-11-2018-0192>.

- Rasmussen, C. E., & Williams, C. K. I. (2008). *Gaussian Processes for Machine Learning (3. Print). Adaptive Computation and Machine Learning*. MIT Press.
- Roberts, S., Osborne, M., Ebdem, M., Reece, S., Gibson, N., & Aigrain, S. (2013). Gaussian processes for time-series modelling. *Philosophical Transactions. Series A, Mathematical, Physical, and Engineering Sciences*, 371(1984), Article 20110550. <http://dx.doi.org/10.1098/rsta.2011.0550>.
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6), 386–408. <http://dx.doi.org/10.1037/h0042519>.
- Sankaran, S. (2014). Demand forecasting of fresh vegetable product by seasonal ARIMA model. *International Journal of Operational Research*, 20(3), 315. <http://dx.doi.org/10.1504/IJOR.2014.062453>.
- Santosa, F., & Symes, W. W. (1986). Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4), 1307–1330. <http://dx.doi.org/10.1137/0907087>.
- Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*.
- Seaman, B., & Bowman, J. (2021). Applicability of the M5 to forecasting at walmart. *International Journal of Forecasting*, <http://dx.doi.org/10.1016/j.ijforecast.2021.06.002>, Advance online publication.
- Shukla, M., & Jharkharia, S. (2011). ARIMA models to forecast demand in fresh supply chains. *International Journal of Operational Research*, 11(1), 1–18. <http://dx.doi.org/10.1504/IJOR.2011.040325>.
- Shumway, R. H., & Stoffer, D. S. (2000). Time series regression and ARIMA models. In R. H. Shumway, & D. S. Stoffer (Eds.), *Springer Texts in Statistics. Time Series Analysis and Its Applications* (pp. 89–212). Springer, http://dx.doi.org/10.1007/978-1-4757-3261-0_2.
- Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56), 1929–1958.
- Stepnicka, M., & Burda, M. (2017). On the results and observations of the time series forecasting competition CIF 2016. In *Fuzz-IEEE 2017: 2017 IEEE International Conference on Fuzzy Systems : 9-12 2017, Royal Continental Hotel, Naples, Italy* (pp. 1–6). IEEE, <http://dx.doi.org/10.1109/FUZZ-IEEE.2017.8015455>.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 58(1), 267–288. <http://dx.doi.org/10.1111/j.2517-6161.1996.tb02080.x>.
- Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1, 211–244. <http://dx.doi.org/10.1162/15324430152748236>.
- SciPy 1.0 Contributors (2020). SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17, 261–272. <http://dx.doi.org/10.1038/s41592-019-0686-2>.
- Waskom, M. L. (2021). Seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60), 3021. <http://dx.doi.org/10.21105/joss.03021>.
- Williams, C., & Rasmussen, C. (1996). Gaussian processes for regression. In D. Touretzky, M. C. Mozer, & M. Hasselmo (Eds.), *Advances in Neural Information Processing Systems (Vol. 8)*. MIT Press.
- Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3), 324–342. <http://dx.doi.org/10.1287/mnsc.6.3.324>.
- Yeo, I.-K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4), 954–959. <http://dx.doi.org/10.1093/biomet/87.4.954>.
- Zentralverband Gartenbau e. V (2020). *Jahresbericht 2020*.
- Zhang, G., Eddy Patuwo, B., & Y. Hu, M. (1998). Forecasting with artificial neural networks. *International Journal of Forecasting*, 14(1), 35–62. [http://dx.doi.org/10.1016/S0169-2070\(97\)00044-7](http://dx.doi.org/10.1016/S0169-2070(97)00044-7).
- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, 67(2), 301–320. <http://dx.doi.org/10.1111/j.1467-9868.2005.00503.x>.

Appendix **B**

Publication B

The subsequent pages show the original full version of the following publication:

PUBLICATION B:

Florian Haselbeck and Dominik G. Grimm (2021). EVARS-GPR: EVent-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data. *KI 2021: Advances in Artificial Intelligence*, 12873, 135–157. https://doi.org/10.1007/978-3-030-87626-5_11



EVARS-GPR: EVent-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data

Florian Haselbeck^{1,2}  and Dominik G. Grimm^{1,2,3} 

- ¹ TUM Campus Straubing for Biotechnology and Sustainability, Bioinformatics, Technical University of Munich, Schulgasse 22, 94315 Straubing, Germany
{florian.haselbeck, dominik.grimm}@hswt.de
- ² Weihenstephan-Triesdorf University of Applied Sciences, Bioinformatics, Petersgasse 18, 94315 Straubing, Germany
- ³ Department of Informatics, Technical University of Munich, Boltzmannstr. 3, 85748 Garching, Germany

Abstract. Time series forecasting is a growing domain with diverse applications. However, changes of the system behavior over time due to internal or external influences are challenging. Therefore, predictions of a previously learned forecasting model might not be useful anymore. In this paper, we present **EVent-triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data (EVARS-GPR)**, a novel online algorithm that is able to handle sudden shifts in the target variable scale of seasonal data. For this purpose, EVARS-GPR combines online change point detection with a refitting of the prediction model using data augmentation for samples prior to a change point. Our experiments on simulated data show that EVARS-GPR is applicable for a wide range of output scale changes. EVARS-GPR has on average a 20.8% lower RMSE on different real-world datasets compared to methods with a similar computational resource consumption. Furthermore, we show that our algorithm leads to a six-fold reduction of the averaged runtime in relation to all comparison partners with a periodical refitting strategy. In summary, we present a computationally efficient online forecasting algorithm for seasonal time series with changes of the target variable scale and demonstrate its functionality on simulated as well as real-world data. All code is publicly available on GitHub: <https://github.com/grimmlab/evars-gpr>.

Keywords: Gaussian process regression · Seasonal time series · Change point detection · Online time series forecasting · Data augmentation

1 Introduction

Time series forecasting is an emerging topic with applications in diverse domains, e.g. business, medicine or energy. These approaches make use of time series data, which describes a system behavior by a sequence of observations within a certain time period and try to predict future values. However, sudden changes of the system behavior over

time are common issues in time series analysis. These sudden changes can be either caused by external or internal influences, e.g. due to operational or strategic decisions. For instance, currently many sales forecasting systems are affected by the SARS-CoV-2 pandemic and energy demand predictions might be impeded by energetic optimizations of big consumers. Some but probably not all of the influential factors can be captured by features. Nevertheless, after a change of the generative data distribution, which reflects the relation between explanatory features and the target variable, predictions of a previously learned model might not be useful anymore. As a result, decisions based on these could cause damage such as a financial loss if e.g. an underestimated demand leads to missed sales [2, 13].

A common but computationally exhaustive approach to handle this problem is to periodically retrain a prediction model during the productive operation [13]. Furthermore, several methods combine change point detection (CPD), i.e. the problem of identifying a change of the generative data distribution, and Gaussian Process Regression (GPR). Some of them work offline and are therefore not suitable for changing data distributions during the online phase [9, 16]. Existing online approaches are either not event-triggered [21, 22], require a certain number of samples of a new generative distribution [12] or are based on *a priori* assumptions in terms of potentially changing time series properties [17]. Furthermore, none of them apply data augmentation (DA) on samples prior to a detected change point to reuse these augmented samples for model retraining.

In this work, we present **Event-triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data (EVARS-GPR)**, for which we provide an overview in Fig. 1. This novel online algorithm combines change point monitoring with a refitting of the prediction model using data augmentation. Compared to existing approaches, the main focus of our algorithm is on seasonal data with sudden changes of the target variable scale while values of explanatory features remain approximately equal, which is a common issue in seasonal time series forecasting. The data augmentation step is triggered after the detection of a change point and a deviation of the target variable scale compared to a certain threshold. This step updates known samples prior to a change point with new information on the changed target variable scale. Consequently, we gain potential useful data for the refitting of the prediction model. Hence, EVARS-GPR is event-triggered and as a result more efficient than a periodical refitting strategy. Furthermore, the algorithm reacts immediately after a detected change point and *a priori* assumptions on the output scale changes are not required. As prediction model we use Gaussian Process Regression (GPR), see Appendix A for an overview. GPR is a flexible and non-parametric Bayesian method including uncertainties of a prediction value, which seems profitable with regard to the practical use of forecasts. Moreover, we evaluate the integration of different approaches for online CPD and DA, two essential parts of EVARS-GPR. We further analyze EVARS-GPR using simulated data and evaluate the performance on real-world datasets including different comparison partners.

The remainder of this paper is organized as follows. In Sect. 2, we describe the related work. Afterwards, we provide the problem formulation in Sect. 3. Then, we outline EVARS-GPR in Sect. 4 followed by the experimental setup in Sect. 5. The experimental results are shown and discussed in Sect. 6, before we draw conclusions.

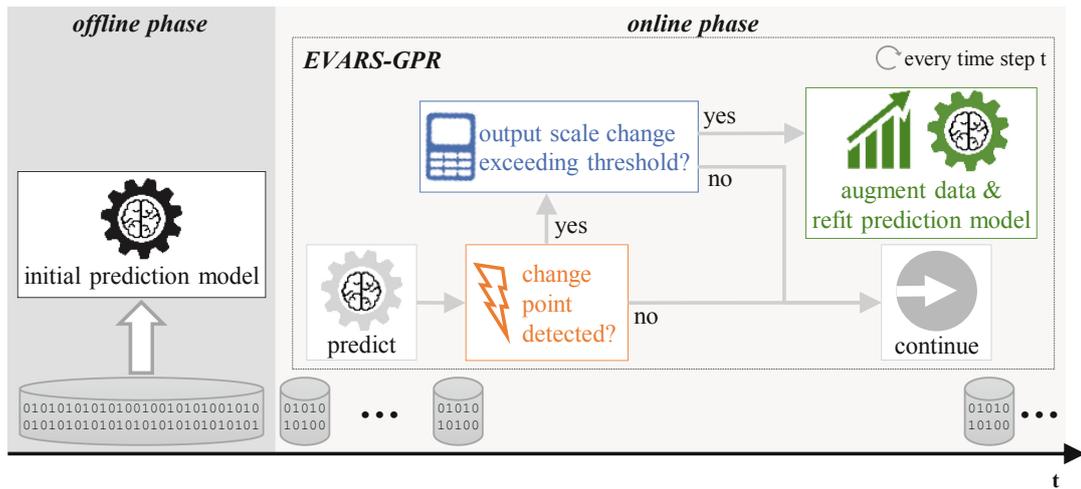


Fig. 1. Overview of EVARS-GPR during the online phase and the preconditions in the offline phase. The initial prediction model is trained offline. During the online phase, the prediction of the next target value is followed by a change point detection. If a change point is detected, the output scaling factor, which sets the target values of the current season in relation to previous seasons, is calculated. If the deviation between the current and last output scaling factor exceeds a threshold, then an augmented refitting of the prediction model is triggered. In case one of the two conditions is not fulfilled, EVARS-GPR continues using the current prediction model.

2 Related Work

Methods enabling GPR models to work with nonstationary data distributions can be divided into offline and online techniques. A common offline approach is to switch between kernel functions, e.g. by multiplication with sigmoid functions [9]. For some technical processes, multiple steady states can be determined. This enables the association of a corresponding model. For inference, the one associated with the current state is selected [16]. However, these approaches are limited to scenarios for which change points respectively steady states can be defined *a priori*. Furthermore, change points occurring abruptly at a single point can be treated as a hyperparameter of a nonstationary covariance function [11]. A further possibility of handling nonstationary data distributions is to augment the input using time-related functions. One option is the introduction of a forgetting factor, which leads to a lower influence of the information contained in older samples. Another common technique is to periodically update the hyperparameters of a GPR model using samples within a specified moving window [22]. A more elaborate approach is Moving-Window GPR (MWGPR). This method discards the oldest sample after a new one becomes available. Moreover, a dual preprocessing and dual updating strategy is performed. This introduces a recursive bias term, which depends on past model errors and is added to the model's prediction to get the final forecast value [21]. All these approaches have the drawback of losing potentially useful information from earlier samples even if the data distribution did not change [22]. GP-non-Bayesian clustering (GP-NBC) is focused on computational efficiency with the goal of making it suitable for resource-constrained environments, e.g. robotic platforms. Based on online-trained GP models, likelihood ratio tests are performed in order to determine whether a new candidate model or a previously stored one should be used in the further process. A

disadvantage of GP-NBC is that a certain number of new samples needs to be available to enable the training of a new model. This may lead to a delayed reaction after a change point occurred [12]. The INstant TEmporal structure Learning (INTEL) algorithm was recently proposed. First, a template model is learned using offline data. Then, a set of candidate models with varying hyperparameters due to assumptions of potential changes during the productive operation is constructed based on the template model. For the final prediction, all models are combined using weights that correspond to the likelihood of a new observation given each model. In its current implementation, INTEL is limited to univariate data. Furthermore, possible changes happening during the online phase need to be assumed *a priori* [17].

3 Problem Formulation

We define a multivariate time series $\mathcal{D} = \{\mathbf{x}_t, y_t\}^n$ as a sequence of n samples consisting of d -dimensional explanatory variables called covariates $\mathbf{x}_t \in \mathbb{R}^d$ and a corresponding target value $y_t \in \mathbb{R}$ at time step t . The target value at time step t , y_t , is drawn from a distribution $p_i(y|\mathbf{x}_t)$, i.e. it is dependent on the covariates \mathbf{x}_t . In this work, we consider seasonal data, meaning data that follows a certain periodicity of length n_{seas} . We assume that periodicity is present for the target variable y as well as for at least some of the covariates \mathbf{X} . Thus, the target variable at time step t can be decomposed in a seasonal component s_t and a residual r_t summing up all other effects: $y_t = s_t + r_t$. Based on its periodicity of length n_{seas} , the seasonal component at time step t is similar to those of previous seasons: $s_t \approx s_{t-k \cdot n_{seas}}$ with $k \in \mathbb{N} \setminus \{0\}$. The covariates \mathbf{x}_t respectively a subset $\boldsymbol{\chi}_t \subseteq \mathbf{x}_t$ of them can also be decomposed in a seasonal component $s_{\boldsymbol{\chi},t}$ and a residual $r_{\boldsymbol{\chi},t}$ into $\boldsymbol{\chi}_t = s_{\boldsymbol{\chi},t} + r_{\boldsymbol{\chi},t}$, with similar periodicity characteristics regarding $s_{\boldsymbol{\chi},t}$. The strength of the seasonal pattern, i.e. the influence of the seasonal component on the final value, might vary for different covariates and target variables.

With n_{off} samples of \mathcal{D} , a model M , here a Gaussian Process Regression, can be trained offline using cross-validation to determine the hyperparameter configuration that delivers predictions \hat{y} generalizing best to the true distribution $p_i(y|\mathbf{x})$. During the online phase, with a new input \mathbf{x}_t provided at every time step t , the model M is used to deliver a prediction for the target variable value \hat{y}_t based on \mathbf{x}_t . However, it is a common issue in time series forecasting that the generative distribution $p_i(y|\mathbf{x})$ our predictor M was trained on might change to another distribution $p_j(y|\mathbf{x})$. The time step at which such a shift happens is called a change point. In this work, we focus on output scale shifts, meaning that the value range of the target variable y changes. Therefore, with regard to the periodicity of the covariates \mathbf{X} and the target variable \mathbf{y} , a similar covariate vector \mathbf{x}_t corresponds to a different target variable y_t as the generative distribution changed. Consequently, the predictions produced by the previously trained model M might not be useful anymore. With EVARS-GPR, we address this problem by combining online change point monitoring of the target variable y and a refitting of the base model M using data augmentation in case a change point is detected. A list of symbols including those of subsequent sections is provided in Appendix B.

4 EVARS-GPR

EVARS-GPR is an online algorithm that is focused on changes resulting in an output scale shift of seasonal multivariate time series, as outlined in Sect. 3. In Fig. 1 and Algorithm 1, we give an overview of EVARS-GPR. Following the problem formulation, we assume an offline-trained model M , which we subsequently call the base model M_{base} . Prior to the online phase, the current prediction model $M_{current}$ is initialized with this offline-trained model M_{base} . As EVARS-GPR operates online, the main part starts with a new sample becoming available at time step t . As a first step, we retrieve the prediction of the next target value \hat{y}_t using the covariates \mathbf{x}_t as well as the current model $M_{current}$. Then, we run an online change point detection (CPD) algorithm, which is updated with the current target variable value y_t . In case we do not detect a shift of the generative distribution $p(y|\mathbf{x}_t)$, the current prediction model $M_{current}$ stays unchanged and the algorithm waits for the next time step $t + 1$. However, if we determine a change point at time step t , the remaining procedures of EVARS-GPR are triggered. First, as EVARS-GPR is focused on changes of the output scale in seasonal time series data, the output scaling factor η is determined. For that purpose, the target values y prior to the change point and within a window of size n_w are considered. These are set in relation to the target values y within the corresponding window of n_η previous seasons with a season length of n_{seas} :

$$\eta = \frac{1}{n_\eta} \sum_{k=1}^{n_\eta} \frac{\sum_{i=t-n_w}^t y_i}{\sum_{j=t-k \cdot n_{seas}-n_w}^{t-k \cdot n_{seas}} y_j} \quad (1)$$

The nominator of Eq. (1) includes current target values y_i prior to the change point, whereas the denominator conveys information on the corresponding period of a previous season. This ratio is averaged over the number of seasons taken into account to retrieve the output scaling factor. Online CPD is prone to false alarms due to outliers. For this reason and to limit the amount of refittings for efficiency, we set a minimum threshold π_η for the deviation between the current output scaling factor η and the output scaling factor of the last augmented refitting η_{old} . If this threshold is exceeded, the augmented refitting of the current model $M_{current}$ is triggered. First, we generate an augmented set of samples \mathcal{D}' based on the dataset prior to the change point at time step t . Thereby, we reuse known samples and update them with new information on the changed target variable scale. Consequently, we gain an augmented dataset \mathcal{D}' for the refitting of the current model $M_{current}$. Furthermore, the last output scaling factor η_{old} is stored. Subsequently, the refitted current model is used for the predictions of the target value. With a new sample arriving at the next time step $t + 1$, the whole cycle of predicting, change point monitoring, potential data augmentation and model refitting starts again.

The goal of CPD is to find abrupt changes in data, in the context of this work resulting in a shift of the scale of the target variable y . A CPD method should ensure a quick reaction to a change point. Considering a real time operation, computationally efficient CPD methods are advantageous. Beyond that, for EVARS-GPR, the CPD and the prediction methods are separated in order to enable the output scale-dependent, augmented model refitting. For these reasons, we excluded approaches such as GPTS-CP [24] and BOCPD-MS [15]. Based on the outlined criteria, we evaluated Bayesian

Online Change Point Detection (BOCPD) and ChangeFinder (CF). More information on these two methods can be found in Appendix C. In both cases, we deseasonalize data via seasonal differencing in order to prevent false alarms due to seasonal effects [2].

Besides online CPD, DA is an essential part of EVARS-GPR. For this work, we focus on computationally efficient approaches ensuring a real time operation and consider small datasets as well. Therefore, we excluded generative models such as TimeGAN [28] or C-RNN-GAN [20]. First, we augmented the original dataset consisting of all samples prior to a change point at time step t , $\mathcal{D}_{0:t}$, by scaling the original target variable vector $\mathbf{y}_{0:t}$. Thereby, we multiply the target variable vector $\mathbf{y}_{0:t}$ with the output scaling factor η and leave the covariates $\mathbf{x}_{0:t}$ unchanged, resulting in the augmented dataset $\mathcal{D}_{0:t}^\eta$. Considering the focus on shifts of the output scale, augmenting the dataset by scaling the target variable vector \mathbf{y} is a reasonable and efficient approach. Second, we used two virtual sample generation techniques for imbalanced regression: Random Oversampling with the introduction of Gaussian Noise (GN) [26] and SMOGN, which combines the former and the Synthetic Minority Oversampling Technique for Regression (SMOTER) [5, 27]. Both methods are outlined in Appendix D.

Algorithm 1. EVARS-GPR

Inputs: $M_{base}, \mathcal{D}_{0:n_{off}}$

Parameters: $n_\eta, n_{seas}, n_w, \pi_\eta$, CPD and DA parameters

Results: $\hat{\mathbf{y}}$

```

1:  $M_{current} = M_{base}$ 
2:  $\eta_{old} = 1$ 
3: for new sample at time step  $t$  do
4:   predict target value:  $\hat{\mathbf{y}}_t = M_{current}(\mathbf{x}_t)$ 
5:   perform online CPD:  $online\_cpd(\mathbf{y}_{0:t}, CPD\ parameters)$  ▷ App. 2
6:   if change point detected then
7:     calculate output scaling factor: ▷ Eq. (1)
8:        $\eta = calc\_output\_scaling\_factor(n_\eta, n_{seas}, n_w)$ 
9:       if  $|\eta - \eta_{old}| / \eta_{old} > \pi_\eta$  then
10:        augment data: ▷ App. 3
11:           $\mathcal{D}' = augment\_data(\mathcal{D}_{0:t}, DA\ parameters)$ 
12:          refit current model:  $refit(M_{current}, \mathcal{D}')$ 
13:           $\eta_{old} = \eta$ 
14:        end if
15:      end if
16:    end for

```

5 Experimental Setup

Subsequently, we will first describe the simulated data we used to determine the configuration of EVARS-GPR and to analyze its behavior. Afterwards, we outline the real-world datasets and the performance evaluation.

5.1 Simulated Data

EVARS-GPR is focused on seasonal data with changes regarding the target variable scale during the online phase. In order to configure and parametrize the algorithm as well as to analyze its behavior, we generated simulated data fulfilling these properties. Figure 2 visualizes a simplified example of simulated data to explain its configuration. As we observe in Fig. 2a, the target variable y follows a periodical pattern with a season length n_{seas} and an amplitude a . Between t_{start} and t_{end} during the online phase, we manipulate y by a multiplication with a factor δ , which results in a change of the output scale. The characteristics of this manipulation factor are visualized in Fig. 2b. At t_{start} , we begin with δ_{base} and increase δ by a slope of κ at every time step t , up to a maximum manipulation factor δ_{max} . Then, the manipulation factor δ stays constant until its sequential decrease is triggered in order to reach δ_{base} at t_{end} . To meet the properties specified in Sect. 3, the covariates x are also periodical. Both x and y can be modified with additive random noise in order to model the seasonality more realistic.

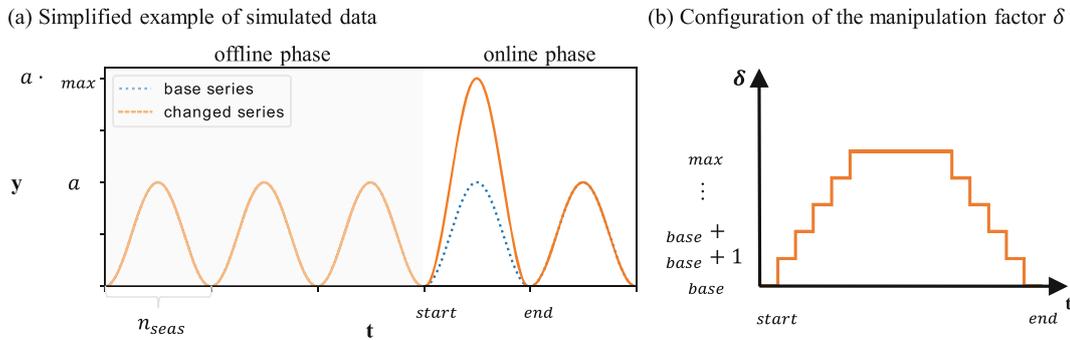


Fig. 2. (a) Visualization of a simplified example of simulated data with the base as well as the changed series, both with a season length of n_{seas} . Between t_{start} and t_{end} , the base series with its amplitude a is changed by multiplication with a manipulation factor δ . (b) Configuration of the manipulation factor δ . Starting from δ_{base} at t_{start} , the manipulation factor δ increases by a slope κ at every time step t . If a maximum manipulation factor δ_{max} is reached, δ stays constant. At t_{end} , the base factor δ_{base} is reached again after sequentially decreasing δ using κ .

In summary, the parameters n_{seas} , t_{start} , t_{end} , δ_{max} and κ enable us to simulate various settings of the output scale change. For instance, t_{start} and t_{end} modify the duration and time of occurrence. Furthermore, δ_{max} marks the maximum extent of the output scale change, whereas κ determines its increase at every time step t , thus the speed respectively abruptness. Based on this, we formulated 67 scenarios and evaluated the performance to select the online CPD and DA method for EVARS-GPR, see Sect. 6.1. Furthermore, the parametrization of EVARS-GPR is based on these scenarios, see Appendix E for an overview. For that purpose, we employed a random search with 100 different parameter settings for each combination of online CPD and DA method [3].

5.2 Real-World Datasets

We additionally evaluated EVARS-GPR on real-world datasets. Based on the algorithm's scope, we selected seasonal time series data, for which we provide more information in

Appendix F. For the horticultural sales prediction dataset *CashierData*¹, we observe a strong sales increase of potted plants (*PotTotal*) during the SARS-CoV-2 pandemic in 2020. Furthermore, we included the following common and publicly available datasets with changes of the output scale during the online phase: *DrugSales* [13], *VisitorNights* [13], *AirPassengers* [4] and *MaunaLoa* [10]. Beyond that, we used time series data without such changes to test the robustness of EVARS-GPR: *ChampagneSales* [18], *TouristsIndia* [7], *Milk* [19], *Beer* [13] and *USDeaths* [19]. We further applied mean imputation for missing values and added calendric as well as statistical features, e.g. lagged target variables, see Appendix F for an overview. Then, we used 80% of the data to determine the base model M_{base} , i.e. the configuration that leads to the best performance in a cross-validation setup. Thereby, we employed a random search over the model’s hyperparameters such as the kernel function as well as preprocessing parameters, e.g. whether to perform a principal component analysis [3]. Finally, we evaluated EVARS-GPR in an online setting for the remaining left out 20% of the data.

5.3 Evaluation

To evaluate the performance on a set of n samples, we used the Root Mean Squared Error (RMSE), which is defined as $RMSE = \sqrt{1/n \sum_{i=1}^n (y_i - \hat{y}_i)^2}$ with the true value y_i and the prediction \hat{y}_i . As the RMSE is scale-dependent, we further applied a scaling by the RMSE value achieved with M_{base} , subsequently called RMSE-ratio. Thus, the performances on simulated scenarios with different scales are comparable.

We included several comparison partners for the real-world datasets. M_{base} applies the offline-trained model during the whole online phase. Furthermore, we employed common but computationally exhaustive periodical refits of the prediction model, which trigger a retraining at every (*PR1*) respectively every second (*PR2*) time step [13]. Moving-Window GPR (MWGPR) is included as an additional computational resource demanding comparison partner, because a refit is needed at every time step t [21]. Moreover, we defined methods with a computational resource consumption similar to EVARS-GPR. These methods also react after a valid change point was detected, so the number of refittings and thus the resource consumption is similar. *CPD_scaled* scales the predictions of M_{base} using the output scaling factor η , whereas *CPD_retrain* triggers a refitting of the current prediction model $M_{current}$ using all original samples prior to a change point. *CPD_MW* also leads to a refitting, but only uses data of the season before the detected change point. For an estimate of the resource consumption, we measured the process-wide CPU time of EVARS-GPR and the computationally exhaustive methods on a machine with two 2.1 GHz *Intel Xeon Gold 6230R* CPUs (each with 26 cores and 52 threads) and a total of 756 GB of memory.

6 Experimental Results

In this section, we will first describe the behavior on simulated data. Afterwards, we outline the results on real-world datasets, before we discuss them.

¹ <https://github.com/grimmlab/HorticulturalSalesPredictions>.

6.1 Behavior on Simulated Data

We determined the configuration and parameters of EVARS-GPR based on simulated data. Using CF for online CPD and a scaling of the original dataset for DA lead to the lowest RMSE-ratio averaged over all scenarios (0.549). Thereby, we experienced that EVARS-GPR is sensitive to hyperparameters, e.g. the window size n_w , see Appendix E regarding the final values. We further analyzed EVARS-GPR on different output scale changes regarding the extent (maximum manipulation factor δ_{max}), speed (slope κ), time of occurrence and duration (both via start t_{start} respectively end index t_{end}). Results are visualized in Fig. 3. The performance of EVARS-GPR was in all scenarios at least equal to M_{base} and outperformed it in most of the cases. Figure 3a and 3b show that the advantage of EVARS-GPR was larger for longer periods with an output change. However, there was also an improvement for shorter durations. EVARS-GPR was beneficial for several extents and speeds of the shift as well as robust for constant scenarios ($\delta_{max} = 1$), see Fig. 3c and 3d. For smaller slopes κ , the improvement tended to decrease with a higher maximum manipulation factor δ_{max} . We observe in Fig. 3d that EVARS-GPR's benefit was mostly smaller for maximum manipulation factors d_{max} close to one. We included further scenarios in Appendix G, which show similar results.

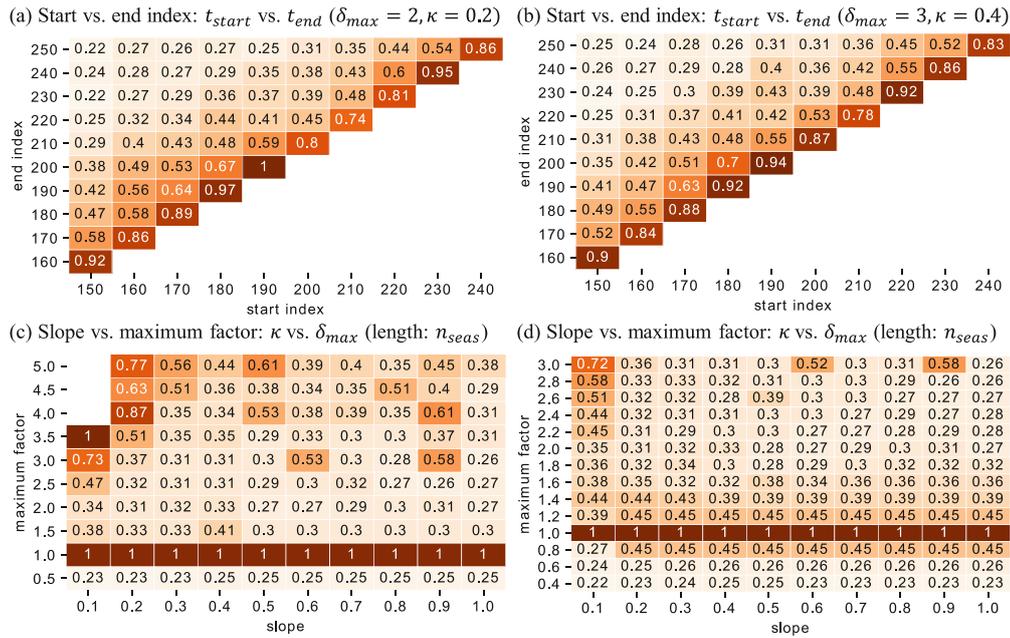


Fig. 3. Behavior on a variety of simulated data ($n_{seas} = 50$). Each box shows the result of the scenario parametrized with the values given on the x- and y- axis. Numbers are RMSE-ratios, lower values reflect a higher improvement compared to M_{base} . We analyzed the following factors of the output scale change: **(a)** time of occurrence and duration via start and end indices t_{start} respectively t_{end} , **(b)** same factors with a higher slope κ and maximum manipulation factor δ_{max} , **(c)** extent and speed of the change via κ and δ_{max} , **(d)** same factors on a finer grid for δ_{max} .

6.2 Results on Real-World Datasets

We further evaluated EVARS-GPR on several real-world datasets. In Table 1, we show the model performance in terms of RMSE compared to methods with a similar computational resource consumption for datasets with a changing target variable scale. As we observe, EVARS-GPR was superior on all datasets. Our algorithm outperformed M_{base} with an improvement of 37.9% averaged over all datasets and the second-best competing method among all datasets by 20.8%.

Table 1. Performance comparison based on RMSE for datasets with output scale changes during the online phase for EVARS-GPR and methods with a similar resource consumption. Numbers show the RMSE of the simulated online phase. The best results are printed bold.

	M_{base}	CPD_{scaled}	$CPD_{retrain}$	CPD_{MW}	$EVARS-GPR$
<i>CashierData</i>	1351.43	2266.57	1314.55	1683.11	1125.34
<i>DrugSales</i>	6.15	5.39	3.90	4.46	2.75
<i>AirPassengers</i>	171.58	101.61	108.79	174.31	93.88
<i>MaunaLoa</i>	34.37	32.01	31.22	33.50	27.96
<i>VisitorNights</i>	10.97	9.30	8.80	10.34	5.11

Beyond that, we compared EVARS-GPR to computationally exhaustive methods, for which we show the results as well as the process-wide CPU time in Table 2. EVARS-GPR outperformed all other methods with respect to the CPU time. In comparison to $PR2$, the method with the second lowest runtimes, the runtime of EVARS-GPR averaged over all datasets was more than six times lower.

Table 2. RMSE and CPU time compared to computationally exhaustive methods for datasets with output scale changes. Numbers show the RMSE and the CPU time averaged over ten runs. The best RMSE results and lowest CPU times excluding M_{base} are printed bold.

		M_{base}	$PR1$	$PR2$	$MWGPR$	$EVARS-GPR$
<i>CashierData</i>	RMSE	1351.43	1119.23	1185.82	1098.16	1125.34
	CPU time [s]	2.38	1443.80	741.71	1106.53	166.06
<i>DrugSales</i>	RMSE	6.15	2.44	2.54	2.86	2.75
	CPU time [s]	0.61	762.05	404.47	550.99	52.40
<i>AirPassengers</i>	RMSE	171.58	69.06	74.28	72.27	93.88
	CPU time [s]	0.93	587.70	303.31	514.53	34.53
<i>MaunaLoa</i>	RMSE	34.37	11.12	12.60	9.88	27.96
	CPU time [s]	3.60	27459.79	13790.61	19525.85	78.99
<i>VisitorNights</i>	RMSE	10.97	5.08	5.21	5.85	5.11
	CPU time [s]	0.35	40.85	22.28	33.87	6.24

It is not surprising that these comparison partners outperformed EVARS-GPR with respect to predictive power, however at the cost of computational runtime. Regarding *AirPassengers* and *MaunaLoa*, for which EVARS-GPR was outperformed in terms of RMSE, the CPU time of EVARS-GPR was 16 respectively 250 times more efficient. However, for *CashierData*, *DrugSales* and *VisitorNights*, the RMSE of EVARS-GPR was comparable to the leading ones, while being computationally much more efficient.

In Table 3, we see that EVARS-GPR was robust for datasets without an output scale change during the online phase as the results were identical to M_{base} .

Table 3. Robustness on datasets without output scale changes during the online phase. Numbers show the RMSE of the simulated online phase.

	<i>ChampagneSales</i>	<i>TouristsIndia</i>	<i>Milk</i>	<i>Beer</i>	<i>USDeaths</i>
M_{base}	1158.26	90707.30	15.16	16.88	276.72
<i>EVARS-GPR</i>	1158.26	90707.30	15.16	16.88	276.72

6.3 Discussion

We showed that EVARS-GPR is able to handle seasonal time series with changes of the target variable scale, both on simulated and real-world data. The performance on simulated data demonstrates a broad applicability regarding the time of occurrence, duration, speed and extent of the output scale change, with a higher advantage over M_{base} for longer durations. Shorter changes are more difficult to detect for online CPD methods, which is one reason for the lower improvement in such settings. Our results further indicate that EVARS-GPR can handle various speeds and extents of the output scale change, which can be seen as different abruptness levels. This applies both for increases as well as decreases. Experiments with smaller extents showed smaller improvements of EVARS-GPR, as it is more difficult to detect such changes. A similar effect can be observed for smaller speeds and larger extents of the output scale change. Nevertheless, EVARS-GPR was at least on par with M_{base} in all cases and outperformed it in most of the settings.

In addition, EVARS-GPR outperformed all methods with a similar computational resource consumption with respect to RMSE on real-world datasets, with a mean improvement of 20.8% compared to the second-best approaches. Regarding *AirPassengers* and *MaunaLoa*, the advantage of EVARS-GPR in terms of RMSE was 7.6% respectively 10.4%. For these datasets, the output scale changes at the detected change points were rather small. Consequently, the DA step did not enhance the performance that much, which might be a reason for the smaller improvements on RMSE in contrast to the other datasets. Furthermore, the difference to other periodical refitting strategies was largest for *AirPassengers* and *MaunaLoa*. This might indicate that not all possible change points were detected or that these datasets possess further data distribution shifts not resulting in an output scale change. With respect to the other three datasets, EVARS-GPR's results were comparable to the periodical refitting strategies, suggesting

that all relevant change points could be detected. Moreover, we showed EVARS-GPR's efficiency in comparison with periodical refitting strategies with a more than six-fold reduction of the averaged runtime in relation to *PR2*. This advantage of EVARS-GPR was even bigger for *AirPassengers* and *MaunaLoa* with a 16 respectively 250 times lower runtime compared to the best performer. Finally, EVARS-GPR was robust for datasets without changes of the target variable scale.

The online detection of change points is an essential part of EVARS-GPR, as wrong or missed detections might lead to a performance decrease. We addressed the problem of misdetections due to outliers with the introduction of a threshold for the output scaling factor. Nevertheless, EVARS-GPR would probably benefit a lot from improvements of the online CPD method. Moreover, we observed lower RMSE values for periodical refitting strategies, especially on a dataset with more samples (*MaunaLoa*). Thus, a combination of EVARS-GPR and a periodical refitting strategy with a lower frequency is an interesting approach for future research. This might result in a computationally efficient algorithm, which is additionally able to capture changing data distributions that do not result in a target variable scale. We further determined the parameters of the whole pipeline based on simulated data. This might not be the best strategy for all settings and real-world applications. However, as the simulated scenarios were diverse and reflected the scope of this work with output scale changes, this is a reasonable approach. Nevertheless, another way for parameter optimization is a further potential for future research. One possibility is to integrate this into the cross-validation performed offline by simulating different manipulations of the real-world data. Beyond that, EVARS-GPR is model-agnostic. Therefore, it seems interesting to transfer this approach to other prediction models, e.g. XGBoost, which is limited to prediction values within the target value range of the training set [8].

7 Conclusion

In this paper, we presented EVARS-GPR, a novel online time series forecasting algorithm that is able to handle sudden shifts in the target variable scale of seasonal data by combining change point monitoring with an augmented refitting of a prediction model. Online change point detection and data augmentation are essential components of EVARS-GPR, for which we evaluated different approaches based on simulated scenarios. Using the resulting configuration and parameterization, we showed on simulated data that EVARS-GPR is applicable for a wide range of output scale changes. Furthermore, EVARS-GPR had on average a 20.8% lower RMSE on different real-world datasets compared to methods with a similar computational resource consumption. Moreover, we demonstrated its computational efficiency compared to periodical refitting strategies with a more than six-fold reduction of the averaged runtime.

Acknowledgments. This work is supported by funds of the Federal Ministry of Food and Agriculture (BMEL) based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support program [grant number 2818504A18]. Furthermore, we acknowledge Maura John and Nikita Genze for fruitful discussions regarding the naming of the algorithm.

Competing Interests. All authors declare that they have no competing interests.

Appendix A: Gaussian Process Regression

With regard to the practical use of forecasts, the uncertainty of a prediction value seems profitable. Providing those by its definition is a main advantage of the nonparametric Bayesian method GPR. To explain this approach, we use the linear model that is defined as

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{w}, y = f(\mathbf{x}) + \epsilon \quad (2)$$

with \mathbf{x} being the input vector, \mathbf{w} the vector of weights, the function value $f(\mathbf{x})$ and observed target value y with additive noise ϵ assumed to follow a zero-mean Gaussian. Combined with the independence assumption of the observation values, we get the likelihood, which reflects how probable the observed target values \mathbf{y} are for the different inputs \mathbf{X} and weights \mathbf{w} :

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^j p(y_i|\mathbf{x}_i, \mathbf{w}) \quad (3)$$

As usual for a Bayesian formulation, we define a prior over the weights, for which we again choose a zero-mean Gaussian. With the defined prior and the likelihood based on the observed data, we can use Bayes' rule to get the posterior of the weights given the data:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|\mathbf{X})} \quad (4)$$

This is also called the maximum a posteriori estimate, which - provided the data - delivers the most likely set of weights \mathbf{w} . As $p(\mathbf{y}|\mathbf{X})$ is independent of \mathbf{w} , we can reformulate this equation expressing the posterior distribution with a Gaussian defined by a mean and covariance matrix:

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \sim N(\bar{\mathbf{w}}, \mathbf{A}^{-1}) \quad (5)$$

During inference, we marginalize out \mathbf{w} and as a result take the average based on all possible \mathbf{w} weighted by their posterior probability:

$$\begin{aligned} p(y_{Test}|\mathbf{x}_{Test}, \mathbf{X}, \mathbf{y}) &= \int p(y_{Test}|\mathbf{x}_{Test}, \mathbf{w})p(\mathbf{w}|\mathbf{X}, \mathbf{y})d\mathbf{w} \\ &= N\left(\frac{1}{\sigma^2}\mathbf{x}_{Test}^T\mathbf{A}^{-1}\mathbf{X}\mathbf{y}, \mathbf{x}_{Test}^T\mathbf{A}^{-1}\mathbf{x}_{Test}\right) \end{aligned} \quad (6)$$

Therefore, we do not only get an output value, but also an uncertainty. So far, we reached the Bayesian formulation of linear regression with its limited expressiveness. To overcome this constraint to linearity, we can project the inputs into a high-dimensional space and apply the linear concept there. This transformation can be accomplished using basis functions $\phi(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}^i$ leading to the following model with i weights \mathbf{w} :

$$f(x) = \phi(x)^T \mathbf{w} \quad (7)$$

Conducting the same derivation as shown above results in a similar outcome:

$$p(y_{Test}|\mathbf{x}_{Test}, \mathbf{X}, \mathbf{y}) = N\left(\frac{1}{\sigma^2}\phi(\mathbf{x}_{Test})^T\mathbf{A}^{-1}\Phi(\mathbf{X})\mathbf{y}, \phi(\mathbf{x}_{Test})^T\mathbf{A}^{-1}\phi(\mathbf{x}_{Test})\right) \quad (8)$$

The need of inverting the $i \times i$ matrix A possibly causes computational problems if the dimension of the feature space i becomes large. To solve this, we can reformulate the above using the so-called “kernel trick”. This leads to the formulation of a Gaussian Process, which is completely specified by its mean and covariance function:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (9)$$

$$m(\mathbf{x}) = E[f(\mathbf{x})] \quad (10)$$

$$k(\mathbf{x}, \mathbf{x}') = E[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \quad (11)$$

$k(\mathbf{x}, \mathbf{x}')$ consists of the covariance value between any two sample points \mathbf{x} and \mathbf{x}' resulting in a $n \times n$ matrix for a training set length of n . The assumption is that the similarity between samples reflects the strength of the correlation between their corresponding target values. Therefore, the function evaluation can be seen as a draw from a multivariate Gaussian distribution defined by $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$. Thus, Gaussian Processes are a distribution over functions rather than parameters, in contrast to Bayesian linear regression. For simplicity, the mean function is often set to zero or a constant value. There are many forms of kernel functions, which need to fulfill certain properties, e.g. being positive semidefinite and symmetric. Furthermore, they can be combined, e.g. by summation or multiplication. The choice of the covariance kernel function is a determining configuration of GPR and its parameters need to be optimized during training [15, 17].

Appendix B: List of Symbols

General Symbols

M	prediction model
t	current time step
\mathcal{D}	time series dataset
n	number of samples of the time series dataset \mathcal{D}
n_{off}	number of samples that are available during the offline phase
\mathbf{x}_t	covariate vector at time step t
d	dimensionality of the covariate vector \mathbf{x}_t
\mathbf{X}	covariate matrix including all covariates vectors
$\boldsymbol{\chi}_t$	subset of the covariate vector \mathbf{x}_t at time step t
$s_{\boldsymbol{\chi},t}$	seasonal component of the subset of the covariate vector $\boldsymbol{\chi}_t$ at time step t
$\mathbf{r}_{\boldsymbol{\chi},t}$	residual component of the subset of the covariate vector $\boldsymbol{\chi}_t$ at time step t
y_t	true target value at time step t
s_t	seasonal component of y_t at time step t
r_t	residual component of y_t at time step t
n_{seas}	length of one season
\hat{y}_t	predicted target value at time step t
$p(y \mathbf{x}_t)$	generative distribution of y .

EVARS-GPR

M_{base}	offline-trained base model
$M_{current}$	current prediction model
η	current output scaling factor
η_{old}	output scaling factor of last augmented refitting
n_w	window size for the calculation of the output scaling factor η
n_η	number of previous seasons considered for the calculation of the output scaling factor η
π_η	minimum threshold for the deviation between the current η and the last output scaling factor η_{old}
\mathcal{D}'	augmented set of samples.

Simulated Data

a	amplitude
n_{seas}	length of a season
t_{start}	start index of the output scale change
t_{end}	end index of the output scale change
δ	multiplicative manipulation factor for the output scale change
δ_{base}	starting manipulation factor
δ_{max}	maximum manipulation factor for the output scale change
κ	slope, i.e. increase per time step t , for the manipulation factor δ .

Appendix C: Online Change Point Detection

The goal of CPD is to find abrupt changes in data, in the context of this work resulting in a shift of the scale of the target variable y . Based on the criteria outlined in Sect. 4, we selected Bayesian Online Change Point Detection and ChangeFinder.

Bayesian Online Change Point Detection (BOCPD) is a common probabilistic technique. BOCPD assumes that a sequence of observations y_1, y_2, \dots, y_T can be divided into non-overlapping partitions ρ within which the data is i.i.d. from a distribution $p(y_t|\theta_\rho)$, with the parameters θ_ρ being i.i.d. as well. A central aspect of BOCPD is the definition of the run length at time step t , r_t , i.e. the time since the last change point. The posterior distribution of the run length r_t can be determined using Bayes' theorem with $y_t^{(r)}$ denoting the observations associated with r_t :

$$p(r_t|y_{1:t}) = \frac{\sum_{r_{t-1}} p(r_t|r_{t-1})p(y_t|r_{t-1}, y_t^{(r)})p(r_{t-1}, y_{1:t-1})}{\sum_{r_t} p(r_t, y_{1:t})} \quad (12)$$

The conditional prior $p(r_t|r_{t-1})$ is defined to be nonzero only for $r_t = 0$ and $r_t = r_{t-1} + 1$ making the algorithm computationally efficient:

$$p(r_t|r_{t-1}) = \begin{cases} H(r_{t-1} + 1) & \text{if } r_t = 0 \\ 1 - H(r_{t-1} + 1) & \text{if } r_t = r_{t-1} + 1 \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

$H(\tau) = \frac{p_{gap}(g=\tau)}{\sum_{i=\tau}^{\infty} p_{gap}(g=i)}$ is the so-called hazard function with the *a priori* probability distribution over the interval between between change points $p_{gap}(g)$. To apply BOCPD, a hazard function needs to be provided. With $p_{gap}(g)$ being a discrete exponential distribution with timescale λ , the hazard function is constant at $H(\tau) = 1/\lambda$, which is a common assumption. Finally, using the posterior distribution of the run length r_t , a change point can be determined [1, 2].

Another type of online CPD techniques suitable for our purposes are likelihood ratio methods, which declare a change point if the probability distributions before and after a candidate point differ significantly. ChangeFinder is a common approach of this kind, which employs a two-stage learning and smoothing strategy using Sequentially Discounting Auto-Regression (SDAR) model learning. In the first stage, we fit an SDAR model for each new sample at time step t to represent the statistical behavior of the data. The model parameters are updated sequentially with a reduction of the influence of older samples. Thereby, we obtain a sequence of probability densities p_1, p_2, \dots, p_t for each y_t . Based on these, we assign an outlier score to each data point, which is defined as $score(y_t) = -\log p_{t-1}(y_t)$. This enables the formulation of an auxiliary time series o_t by building moving averages of the outlier scores within a time window T for each time step t :

$$o_t = \frac{1}{T} \sum_{i=t-T+1}^t score(y_i) \quad (14)$$

After this first smoothing, the second stage starts. Thereby, another SDAR model is fitted using o_t , also resulting in a sequence of probability densities q_1, q_2, \dots, q_t . Finally, we get a change point score z_t after a second smoothing step within a time window T :

$$z_t = \frac{1}{T} \sum_{i=t-T+1}^t -\log q_{t-1}(o_i) \quad (15)$$

A higher value of z_t corresponds to a higher probability of a change point at time step t . Hence, a threshold π_{cf} at which a change point is declared, needs to be defined [2, 14, 25].

Appendix D: Data Augmentation

As outlined in Sect. 4, we used Virtual Sample Generation approaches for imbalanced regression besides a scaling of the original dataset for data augmentation. These methods are suitable for continuous target variables and small datasets. We therefore selected the two following approaches: Random Oversampling with the introduction of Gaussian Noise (GN) [26] and SMOGN, which combines the former and the Synthetic Minority Oversampling TEchnique for Regression (SMOTER) [5, 27]. Both methods start with the assignment of a relevance value to every sample (\mathbf{x}_i, y_i) of a dataset \mathcal{D}_{vsg} using a relevance function $\phi : y \rightarrow [0, 1]$. Based on these and a specified threshold π_{rel} , the dataset \mathcal{D}_{vsg} is splitted into a subset of normal and rare cases, \mathcal{D}_N respectively \mathcal{D}_R . We employed a relevance function, which proposes an inverse proportionality of the relevance value and the probability density function of y [23]. Therefore, extreme cases have a higher relevance value. Furthermore, we tested two compositions of \mathcal{D}_{vsg} . In the first case, \mathcal{D}_{vsg} was equal to the original dataset up to the change point at time step t , $\mathcal{D}_{0:t}$, and in the second one $\mathcal{D}_{0:t}$ and the output scaled dataset $\mathcal{D}_{0:t}^\eta$ were concatenated. Both GN and SMOGN then apply a Random Undersampling strategy for the normal cases \mathcal{D}_N , meaning that a specified share of these is randomly selected to get \mathcal{D}_{us} .

Furthermore, GN generates new samples \mathcal{D}_{os} based on the rare cases \mathcal{D}_R by adding Gaussian Noise to the target variable as well as the numeric covariates. Values for nominal attributes are randomly selected with a probability proportional to their frequency in the dataset. Finally, for GN, the undersampled and oversampled cases are concatenated to the augmented dataset $\mathcal{D}^{GN} = \{\mathcal{D}_{us}, \mathcal{D}_{os}\}$.

SMOGN instead employs two different oversampling techniques: GN and SMOTER. Prior to the sample generation, the k -nearest neighbors of a seed sample are determined. If a randomly selected k -nearest neighbor is within a specified maximum distance, SMOTER is performed resulting in a set of new samples $\mathcal{D}_{os}^{SMOTER}$. With SMOTER, values of numeric attributes are interpolated and nominal ones are randomly selected. The target value is determined by a weighted average with weights that are inversely proportional to the distance between the seed samples and the new generated one. In case the maximum distance is exceeded, GN is performed, leading to a second oversampling dataset \mathcal{D}_{os}^{GN} . The final augmented dataset for SMOGN therefore consists of three sets: $\mathcal{D}^{SMOGN} = \{\mathcal{D}_{us}, \mathcal{D}_{os}^{SMOTER}, \mathcal{D}_{os}^{GN}\}$ [5, 6, 23].

Appendix E: EVARS-GPR Parameters

Table 4 provides an overview of EVARS-GPR's parameters including all applied CPD and DA methods.

Table 4. Overview of EVARS-GPR's parameters with all analyzed methods CPD and DA.

Category	Parameter	Explanation
General parameters	scale_window(<i>_factor</i>)	Size of window prior to detected change point for calculation of output scaling factor, alternatively formulated as a factor of the season length
	scale_window_minimum	Minimum window size for output scaling factor
	scale_seasons	Number of seasons considered for output scaling factor
	scale_thr	Minimum threshold for the deviation between the current and last output scaling factor to trigger augmented refitting
CPD parameters	BOCPD	
	const_hazard(<i>_factor</i>)	Constant of the hazard function, alternatively formulated as a factor of the season length
	ChangeFinder	
	cf_r	Forgetting factor of the SDAR models
	cf_order	Order of the SDAR models
	cf_smooth	Window size for the smoothing step
	cf_thr_perc	Percentile threshold of the anomaly score during the offline phase to declare a change point
DA parameters	max_samples(<i>_factor</i>)	Maximum number of samples for DA, alternatively formulated as number of seasons
	GN	
	gn_operc	Oversampling percentage
	gn_uperc	Undersampling percentage
	gn_thr	Threshold to determine normal and rare values
	append	Specify if scaled dataset version is appended prior to sample generation
	SMOGN	
	smogn_relthr	Threshold to determine normal and rare values
	smogn_relcoef	Box plot coefficient for relevance function
	smogn_under_samp	Specify if undersampling is performed
	append	Specify if scaled dataset version is appended prior to sample generation

Based on the results over all 67 simulated scenarios, we selected CF for online CPD and a scaling of the original dataset for DA. Furthermore, our experiments yielded the following parameters: scale_window_factor = 0.1, scale_window_minimum = 2, scale_seasons = 2, scale_thr = 0.1, cf_r = 0.4, cf_order = 1, cf_smooth = 4, cf_thr_perc = 70. For efficiency, we set max_samples_factor = 10.

Appendix F: Real-World Datasets

Tables 5 and 6 show an overview of the real-world datasets used for evaluation as well as the additionally derived features.

Table 5. Overview of the used real-world datasets.

Dataset	Explanation	Samples
<i>Datasets with a changing output scale during the online phase</i>		
<i>CashierData</i>	Weekly sales of a horticultural retailer	195
<i>DrugSales</i>	Sales of antidiabetic drugs per month	204
<i>VisitorNights</i>	Visitor nights per quarter in millions in Australia	68
<i>AirPassengers</i>	Total number of US airline passengers per month	144
<i>MaunaLoa</i>	Monthly averaged parts per million of CO2 measured at Mauna Loa observatory, Hawaii	751
<i>Datasets without a changing output scale during the online phase</i>		
<i>ChampagneSales</i>	Sales of perrin freres champagne	105
<i>TouristsIndia</i>	Foreign tourist arrivals per quarter in India	48
<i>Milk</i>	Average milk production per cow and month	168
<i>Beer</i>	Australian monthly beer production	56
<i>USDeaths</i>	Accidental deaths in the US per month	72

Table 6. Overview of additional calendric and statistical features. Not all features are applicable for all datasets, e.g. due to the temporal resolution. Features are added to existing ones.

Category	Features	Explanation
Calendric features	Date based features	Hour, day of month, weekday, month, quarter
	Working day	Flag showing if the day is a working day
Statistical features	Lagged variables	Prior values of the target variable/features
	Seasonal lagged variables	Prior values of the preceding season
	Rolling statistics	Rolling mean and maximum within a window
	Seasonal rolling statistics	Seasonal rolling mean and maximum within a window
	Rolling weekday statistics	Rolling mean and maximum within a window calculated for each weekday

Appendix G: Further Simulated Scenarios

Figures 4, 5 and 6 show further simulated scenarios. Each box shows the result of the scenario parametrized with the value given on the x- and y- axis. Numbers are RMSE-ratios, lower values reflect a higher improvement compared to M_{base} .

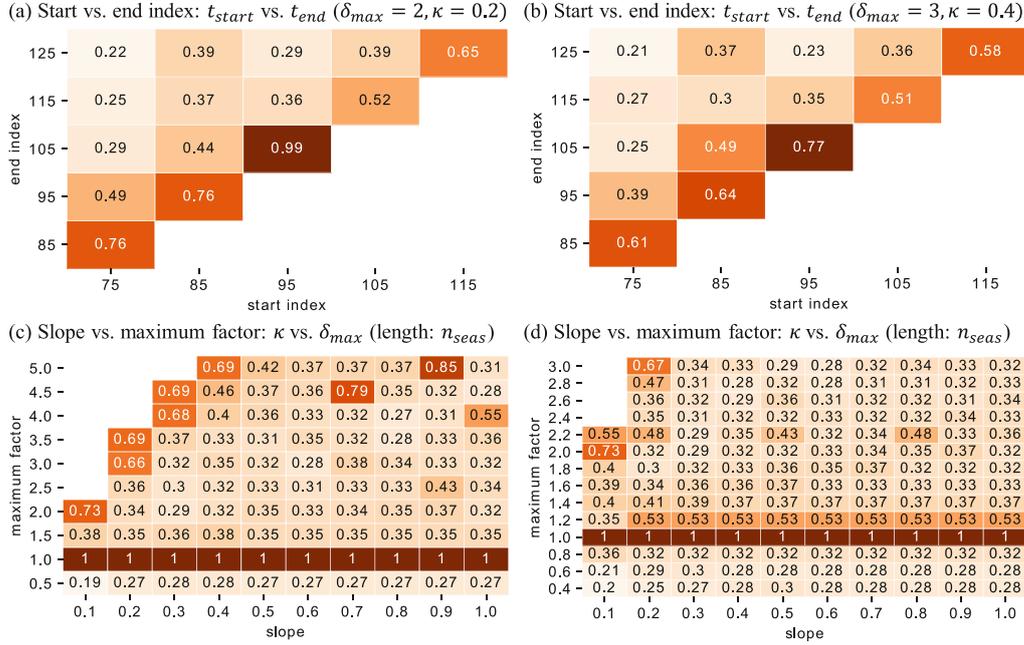


Fig. 4. Behavior on a variety of simulated data ($n_{seas} = 25$). (a) time of occurrence and duration via start and end indices t_{start} respectively t_{end} , (b) same factors with a higher slope κ and maximum manipulation factor δ_{max} , (c) extent and speed of the change via κ and δ_{max} , (d) same factors on a finer grid for δ_{max} .

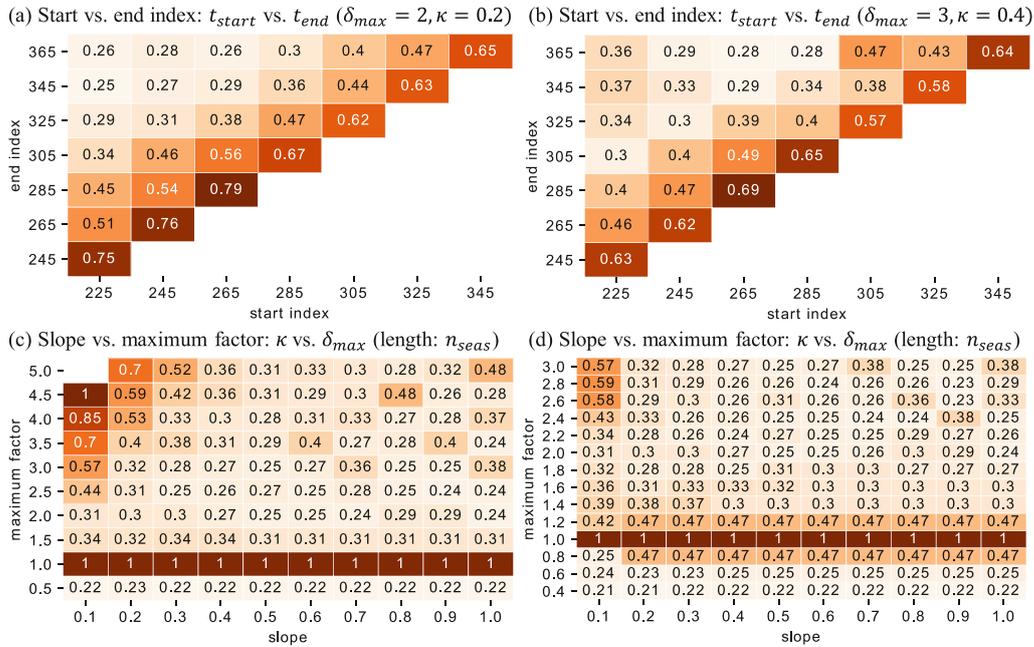


Fig. 5. Behavior on a variety of simulated data ($n_{seas} = 75$). (a) time of occurrence and duration via start and end indices t_{start} respectively t_{end} , (b) same factors with a higher slope κ and maximum manipulation factor δ_{max} , (c) extent and speed of the change via κ and δ_{max} , (d) same factors on a finer grid for δ_{max} .

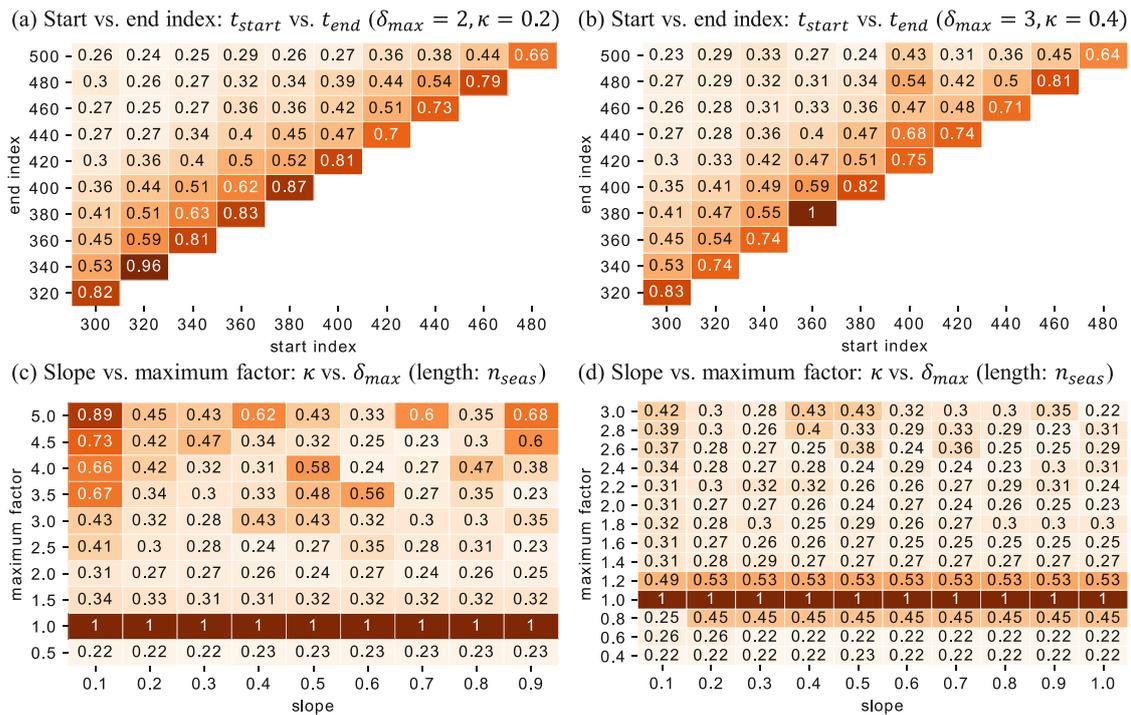


Fig. 6. Behavior on a variety of simulated data ($n_{seas} = 100$). (a) time of occurrence and duration via start and end indices t_{start} respectively t_{end} , (b) same factors with a higher slope κ and maximum manipulation factor δ_{max} , (c) extent and speed of the change via κ and δ_{max} , (d) same factors on a finer grid for δ_{max} .

References

1. Adams, R.P., MacKay, D.J.C.: Bayesian online changepoint detection (2007)
2. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. *Knowl. Inf. Syst.* **51**(2), 339–367 (2016). <https://doi.org/10.1007/s10115-016-0987-z>
3. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**(10), 281–305 (2012)
4. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: *Time Series Analysis. Forecasting and Control*. Wiley Series in Probability and Statistics. Wiley, Hoboken (2016)
5. Branco, P., Torgo, L., Ribeiro, R.P.: SMOGN: a pre-processing approach for imbalanced regression. In: Torgo, L., Krawczyk, B., Branco, P., Moniz, N. (eds.) *Proceedings of the First International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pp. 36–50. PMLR, Skopje (2017)
6. Branco, P., Torgo, L., Ribeiro, R.P.: Pre-processing approaches for imbalanced distributions in regression. *Neurocomputing* **343**, 76–99 (2019). <https://doi.org/10.1016/j.neucom.2018.11.100>
7. Chakrabarty, N.: Quarterly foreign tourist arrivals in India. *Determinants of Foreign Tourism Demand and Foreign Tourist Arrivals (2005–2016)*. <https://www.kaggle.com/navoneel/fta-data>. Accessed 14 May 2021
8. Chen, T., Guestrin, C.: XGBoost: A scalable tree boosting system. In: Krishnapuram, B., Shah, M., Smola, A., Aggarwal, C., Shen, D., Rastogi, R. (eds.) *KDD 2016. 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 785–794. Association for Computing Machinery Inc. (ACM), New York (2016). <https://doi.org/10.1145/2939672.2939785>

9. Duvenaud, D.: Automatic model construction with Gaussian processes. Doctoral thesis, University of Cambridge (2014)
10. Earth System Research Laboratory: CO2 PPM - Trends in Atmospheric Carbon Dioxide. Atmospheric Carbon Dioxide Dry Air Mole Fractions at Mauna Loa, Hawaii. <https://datahub.io/core/co2-ppm-daily#data>. Accessed 14 May 2021
11. Garnett, R., Osborne, M.A., Reece, S., Rogers, A., Roberts, S.J.: Sequential Bayesian prediction in the presence of changepoints and faults. *Comput. J.* **53**(9), 1430–1446 (2010). <https://doi.org/10.1093/comjnl/bxq003>
12. Grande, R.C., Walsh, T.J., Chowdhary, G., Ferguson, S., How, J.P.: Online regression for data with changepoints using gaussian processes and reusable models. *IEEE Trans. Neural Netwo. Learn. Syst.* **28**(9), 2115–2128 (2017). <https://doi.org/10.1109/TNNLS.2016.2574565>
13. Hyndman, R.J., Athanasopoulos, G.: *Forecasting: Principles and Practice*, 2nd edn. OTexts, Melbourne (2018)
14. Iwata, T., Nakamura, K., Tokusashi, Y., Matsutani, H.: Accelerating online change-point detection algorithm using 10 GbE FPGA NIC. In: Mencagli, G., et al. (eds.) *Euro-Par 2018*. LNCS, vol. 11339, pp. 506–517. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-10549-5_40
15. Knoblauch, J., Damoulas, T.: Spatio-temporal Bayesian on-line changepoint detection with model selection. In: Dy, J., Krause, A. (eds.) *Proceedings of the 35th International Conference on Machine Learning*, pp. 2718–2727. PMLR (2018)
16. Liu, Y., Chen, T., Chen, J.: Auto-switch Gaussian process regression-based probabilistic soft sensors for industrial multigrade processes with transitions. *Ind. Eng. Chem. Res.* **54**(18), 5037–5047 (2015). <https://doi.org/10.1021/ie504185j>
17. Liu, B., Qi, Y., Chen, K.-J.: Sequential online prediction in the presence of outliers and change points: an instant temporal structure learning approach. *Neurocomputing* **413**, 240–258 (2020). <https://doi.org/10.1016/j.neucom.2020.07.011>
18. Makridakis, S., Wheelwright, S.C.: *Forecasting Methods for Management*, 5th edn. Wiley, New York (1989)
19. Makridakis, S.G., Wheelwright, S.C., Hyndman, R.J.: *Forecasting. Methods and Applications*, 3rd edn. Wiley, Hoboken (1998)
20. Mogren, O.: C-RNN-GAN: A continuous recurrent neural network with adversarial training. In: *Constructive Machine Learning Workshop (CML) at NIPS 2016* (2016)
21. Ni, W., Tan, S.K., Ng, W.J., Brown, S.D.: Moving-window GPR for nonlinear dynamic system modeling with dual updating and dual preprocessing. *Ind. Eng. Chem. Res.* **51**(18), 6416–6428 (2012). <https://doi.org/10.1021/ie201898a>
22. Perez-Cruz, F., van Vaerenbergh, S., Murillo-Fuentes, J.J., Lazaro-Gredilla, M., Santamaria, I.: Gaussian processes for nonlinear signal processing: an overview of recent advances. *IEEE Signal Process. Mag.* **30**(4), 40–50 (2013). <https://doi.org/10.1109/MSP.2013.2250352>
23. Ribeiro, R.P.: *Utility-based regression*. Doctoral thesis, University of Porto (2011)
24. Saatçi, Y., Turner, R., Rasmussen, C.E.: Gaussian process change point models. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pp. 927–934. Omnipress, Madison (2010)
25. Takeuchi, J., Yamanishi, K.: A unifying framework for detecting outliers and change points from time series. *IEEE Trans. Knowl. Data Eng.* **18**(4), 482–492 (2006). <https://doi.org/10.1109/TKDE.2006.1599387>
26. Torgo, L., Ribeiro, R.: Utility-based regression. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) *PKDD 2007*. LNCS (LNAI), vol. 4702, pp. 597–604. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74976-9_63

27. Torgo, L., Ribeiro, R.P., Pfahringer, B., Branco, P.: SMOTE for regression. In: Correia, L., Reis, L.P., Cascalho, J. (eds.) EPIA 2013. LNCS (LNAI), vol. 8154, pp. 378–389. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40669-0_33
28. Yoon, J., Jarrett, D., van der Schaar, M.: Time-series generative adversarial networks. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems (NeurIPS 2019), vol. 32, pp. 5508–5518. Curran Associates, Inc. (2019)

Appendix C

Publication C

The subsequent pages show the original full version of the following publication:

PUBLICATION C:

Jan D. Hüwel*, **Florian Haselbeck***, Dominik G. Grimm and Christian Beecks (2022). Dynamically Self-adjusting Gaussian Processes for Data Stream Modelling. *KI 2022: Advances in Artificial Intelligence, 13404*, 96-114. https://doi.org/10.1007/978-3-031-15791-2_10

**Both authors contributed equally and share first authorship.*



Dynamically Self-adjusting Gaussian Processes for Data Stream Modelling

Jan David Hüwel¹ , Florian Haselbeck^{2,3} , Dominik G. Grimm^{2,3,4} ,
and Christian Beecks¹

¹ Department of Mathematics and Computer Science, University of Hagen,
Hagen, Germany

{jan.huwel, christian.beecks}@fernuni-hagen.de

² Technical University of Munich, Campus Straubing for Biotechnology
and Sustainability, Bioinformatics, Straubing, Germany

³ Weihenstephan-Triesdorf University of Applied Sciences, Bioinformatics,
Straubing, Germany

{florian.haselbeck, dominik.grimm}@hswt.de

⁴ Technical University of Munich, Department of Informatics, Garching, Germany

Abstract. One of the major challenges in time series analysis are changing data distributions, especially when processing data streams. To ensure an up-to-date model delivering useful predictions at all times, model reconfigurations are required to adapt to such evolving streams. For Gaussian processes, this might require the adaptation of the internal kernel expression. In this paper, we present dynamically self-adjusting Gaussian processes by introducing **Event-Triggered Kernel Adjustments** in Gaussian process modelling (ETKA), a novel data stream modelling algorithm that can handle evolving and changing data distributions. To this end, we enhance the recently introduced Adjusting Kernel Search with a novel online change point detection method. Our experiments on simulated data with varying change point patterns suggest a broad applicability of ETKA. On real-world data, ETKA outperforms comparison partners that differ regarding the model adjustment and its refitting trigger in nine respective ten out of 14 cases. These results confirm ETKA's ability to enable a more accurate and, in some settings, also more efficient data stream processing via Gaussian processes.

Keywords: Gaussian process · Time series modelling · Change point detection · Kernel search · Data stream modelling

1 Introduction

For many applications, accurate real-time analysis of data streams is essential to guarantee a constant workflow. In order to analyze data streams, incoming data

J.D. Hüwel and F. Haselbeck—Both authors contributed equally.

© The Author(s) 2022

R. Bergmann et al. (Eds.): KI 2022, LNAI 13404, pp. 96–114, 2022.

https://doi.org/10.1007/978-3-031-15791-2_10

points need to be incorporated into an online-generated data model. However, changing data distributions at so called change points are a common challenge in data stream modelling [4]. As a consequence, an outdated prediction model might impair a downstream application. For instance, this could lead to overstocking and missed sales in demand forecasting or power supply issues in case of smart grid systems [2, 11, 12]. Providing an up-to-date model is therefore a major objective when modelling data streams. However, identifying the correct time point for model reconfiguration is challenging. Simply adjusting the current model periodically bypasses this challenge, but it might lead to prolonged periods with inaccurate models or increased computational costs due to unnecessary reconfigurations. Because of these drawbacks, many algorithms aim to detect change points online and consequently trigger model adjustments [3, 18].

A Gaussian process (GP) is a stochastic process based on the Gaussian distribution and is commonly used as a non-parametric machine learning model [17]. GPs' probabilistic nature makes them excel at dealing with small and noisy datasets. To incorporate knowledge on the general behavior of the data, GPs use positive semi-definite covariance functions, often called kernels. If no prior knowledge about this behavior is available, an automatic kernel search can determine a fitting function for the given data [8]. However, this is usually a computationally expensive process and requires the optimization of numerous GP models. Recently, the Adjusting Kernel Search (AKS) [13] algorithm was introduced to accelerate this process on data streams. If multiple GP models are used to represent consecutive segments of a stream, it is often reasonable to assume that the models' covariance functions will be similar. AKS enables a search of similar kernels based on a given kernel expression and circumvents the construction of novel expressions from scratch.

In this paper, we enhance AKS with a novel GP-based change point detection (CPD) method in order to propose the **Event-Triggered Kernel Adjustments in Gaussian process modelling (ETKA)** algorithm. The major objective of this algorithm is to deliver an up-to-date GP model describing the current data behavior at all times. We evaluate ETKA based on simulated as well as real-world data and compare it to alternatives in CPD and model inference. Beyond that, we present multiple ways to further expand and improve this method.

The rest of the paper is structured as follows: In Sect. 2, we outline relevant literature about GPs, kernel search algorithms and CPD methods. Sect. 3 introduces the ETKA algorithm. In Sect. 4 we describe our experimental setup. Afterwards, in Sect. 5, we show and discuss the results, before we conclude our findings in Sect. 6.

2 Related Work

The research we present in this paper combines the fields CPD and GP-based data stream modelling. In this section, we briefly introduce relevant works from these fields. Due to a lack of space, we include a formal introduction of GPs and kernel search approaches in Appendix 1.

GPs are commonly used probabilistic machine learning models that mainly depend on their inherent kernel function. An appropriate kernel expression can be chosen by an expert based on previous knowledge about the data. Without such expert knowledge, automatic kernel search algorithms can be employed to find an optimal fit for the given data [6, 8, 14, 15]. In 2013, Duvenaud et al. [8] introduced such an algorithm for the first time, i.e. the Compositional Kernel Search (CKS). Lloyd et al. [15] expanded the method to the Automatic Bayesian Covariance Discovery (ABCD) by including change point kernels. Since both algorithms require the exact evaluation of numerous GP models per iteration, they are restricted to small to medium-sized datasets only.

The problem of scalability was addressed in different ways: Kim et al. [14] introduced the Scalable Kernel Composition (SKC) in 2018, which performs model selection via lower and upper bounds for the GPs' marginal likelihood instead of the exact evaluations. Berns et al. developed new approaches that use a prior segmentation of the data to accelerate the search [5] or perform the segmentation themselves [6]. We aim to expand this idea by performing a dynamic segmentation via online CPD during the modelling process. Recently, Hüwel et al. [13] introduced the Adjusting Kernel Search (AKS) algorithm, which exploits prior assumptions about the data without ascertaining the kernel function. More details about AKS can be found in Appendix 1.

Haselbeck et al. [10] developed EVARS-GPR, a framework to update a GP model online at certain change points. While this approach is restricted to output scale changes, it can be seen as a predecessor of this work due to its retraining of a GP at online-detected change points.

CPD approaches can be separated into offline and online methods. The former were extensively reviewed by Truong et al. [18]. In this paper, we focus on online methods to enable real-time model adjustments. Aminikhanghahi and Cook [3] provided an elaborate overview of available approaches in this area. One widely-used method is the Bayesian Online Change Point Detection [1]. It uses Bayes' rule to determine the number of observations since the last change point and a hazard function to predict a new one. Another commonly used method is the cumulative sum (CUSUM) [16]. It tracks an accumulated deviation score over multiple data points and detects a change point when that score exceeds a custom threshold. CUSUM's high potential for adjustments allows us to use this approach for ETKA.

3 ETKA

Previous applications of AKS employed periodic adjustments of the GP model [13]. This potentially leads to extended periods of incoming data points being processed with an outdated model. Contrariwise, unnecessarily frequent model reconfigurations cause increased computational costs. For this reason, we present the Event-Triggered Kernel Adjustments in Gaussian process Modelling (ETKA) algorithm, an enhancement of AKS with a novel GP-based online CPD approach.

The combination of a kernel search algorithm with a CPD method is obvious, but the nature of GPs results in specific requirements for an optimal CPD

method. Changes should not be detected in primary statistics of the incoming data, such as its mean or variance, as a GP model does not need to be adjusted to regular changes of that kind. Rather, we need to find changes in the abstract behavior and tendencies underlying the data. For example, if the periodicity changes, we want to adjust the model to find a new period length value. Aside from an accurate modelling, there are two additional requirements for CPD: the method should be simple, computationally efficient and easily comprehensible in order to maintain high explainability.

Within ETKA, we achieve these goals by employing a CUSUM approach [16] based on the current GP's performance. We hypothesize that data that differs from the current model's prediction for multiple points in a row signifies a change point. In this case, a kernel search using AKS is triggered to adjust the model to the novel data. The exact procedure of ETKA is explained below and presented in Algorithm 1.

Algorithm 1: The Event-Triggered Kernel Adjustment

Data: $D = (x_i, y_i)_{i=1, \dots, N}$, base kernel set B , window size w , tolerance δ , threshold ε , CKS iterations i_{CKS} , AKS iterations i_{AKS}

Result: change points CP , kernels \mathcal{K}

- 1 $CP \leftarrow \square$
- 2 $\mathcal{K} \leftarrow \square$
- 3 $s \leftarrow 0$
- 4 $k, \sigma^2 \leftarrow CKS(D_{1:w}, B, i_{CKS})$
- 5 **for** $i = w + 1, \dots, N$ **do**
- 6 $\hat{y}_i \leftarrow$
 $k(x_i, \mathbf{x}_{i-w, \dots, i-1}) [k(\mathbf{x}_{i-w, \dots, w-1}, \mathbf{x}_{i-w, \dots, w-1}) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y}_{i-w, \dots, w-1}$
- 7 $s \leftarrow \max(0, s + |y_i - \hat{y}_i| - \delta \cdot 2 \cdot \sigma)$
- 8 **if** $s > \varepsilon$ **then**
- 9 $s \leftarrow 0$
- 10 $\mathcal{K} \leftarrow \mathcal{K} \cup [k]$
- 11 $CP \leftarrow CP \cup [x_i]$
- 12 $k, \sigma^2 \leftarrow AKS(D_{i-w, \dots, w}, B, i_{AKS}, k)$

First, we construct a GP model using CKS on an initial window of w data points. By using this model with kernel $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, we make a prediction \hat{y}_i for the next value starting from $i = w + 1$ after this initial window of length w as follows [17]:

$$\hat{y}_i = k(x_i, \mathbf{x}_{i-w, \dots, i-1}) [k(\mathbf{x}_{i-w, \dots, w-1}, \mathbf{x}_{i-w, \dots, w-1}) + \sigma^2 \mathbf{I}]^{-1} \mathbf{y}_{i-w, \dots, w-1} \quad (1)$$

Then, we calculate the absolute deviation $|y_i - \hat{y}_i|$ between the observed value y_i and our prediction \hat{y}_i . Afterwards, the window slides one position further and the consecutive data points are used for the next prediction step employing the

current GP. The accumulating error is used together with the GP’s noise σ^2 and a tolerance factor $\delta \in \mathbb{R}$ to compute a change point score s :

$$s \leftarrow \max(0, s + |y_i - \hat{y}_i| - \delta \cdot 2 \cdot \sigma) \quad (2)$$

If this score surpasses a certain threshold $\varepsilon \in \mathbb{R}_{>0}$, a change point is detected at x_i and s is reset to zero. With this CPD approach, incoming data points need to be within the inner $100 \cdot \delta\%$ of the GP’s confidence interval in order to count as accurately predicted. The further a data point is outside this interval, the more it increases s and the faster a change point is detected. When the need for a model adjustment is triggered due to a detected change point, the GP’s kernel is adjusted with AKS on the current window w . Then, the procedure with the CUSUM-based CPD and a potential trigger of AKS continues with this updated model. In settings where other kernel search methods are considered more useful, this step can easily be substituted with the corresponding approach.

4 Experimental Setup

In this section, we present the experimental settings that we employed to produce the results shown in Sect. 5. All experiments were conducted on an 11th generation Intel Core i9-11900H processor with 8 cores à 2.50 GHz. For reproducibility, we published all code and data on GitHub¹.

4.1 Simulated Data

For the development and evaluation of ETKA under controlled and predefined settings, we generated artificial datasets². The simulations are based on univariate time series of length n with values $\mathbf{b} \in \mathbb{R}^n$ that follow a periodicity of length n_{per} and have an amplitude of size a . Furthermore, we consider the multiplicative components linear trend $\mathbf{l} \in \mathbb{R}^n$ and random noise $\boldsymbol{\eta} \in \mathbb{R}^n$. The size of the linear trend \mathbf{l} is defined by the coefficient m of a linear model. The random noise $\boldsymbol{\eta}$ is sampled from a normal distribution with a mean value of 1 and a standard deviation s . These components enable the simulation of time series data $\mathbf{y} \in \mathbb{R}^n$ with

$$y_i = b_i \cdot l_i \cdot \eta_i \quad (3)$$

where b_i is the value of the base signal \mathbf{b} at index i , l_i the factor of the linear trend \mathbf{l} at index i and η_i the factor of the random noise $\boldsymbol{\eta}$ at index i . A factor can be left out to simulate that a component is not present. Finally, we can generate change points by fading between time series \mathbf{y} of different properties. The abruptness of the change is adjustable via a fading window w_{fade} . We used this framework to model for instance changes of the period length (parameter n_{per} of \mathbf{b}), the output scale (parameter a of \mathbf{b}), the size of the linear trend

¹ <https://github.com/JanHuewel/ETKA>.

² <https://github.com/Nike-Inc/timeseries-generator>.

(parameter m of \boldsymbol{l}) or the noise level (parameter s of $\boldsymbol{\eta}$). We both considered time series with single and multiple change points, all having a length of 2000 data points. An overview of all simulation settings can be found in Table 3 in Appendix 2.

4.2 Real-World Data

We further included 14 real-world datasets from various domains, see Table 4 in the appendix. Besides the different domains, the datasets show a wide range regarding the number of samples reaching from 180 (*Call centre*) up to 7718 (*Airquality*). Examining the minimum (min) and maximum (max) target values of each time series, we further observe that many datasets have a large value range. Beyond that, several datasets have a standard deviation (std) that is rather high relative to their value range, indicating a strong variation in the time series. In summary, this collection of common time series datasets with varying characteristics and domains enables a broad evaluation of ETKA.

4.3 Evaluation

As comparison partners in our experiments, we included alternatives to ETKA in both main aspects: the choices when to adjust the model and how to adjust the model. Hence, we compare the previously described CPD-based approach to data-agnostic periodic model adjustments (*PER AKS*). These model adjustments are done three times in equidistant intervals within the dataset after the initial kernel search. Furthermore, only the data in the current window is used, disregarding everything before that. By doing so, we examine the concept of CPD-driven adjustments at the cost of frequent model predictions needed to enable CPD. Furthermore, as mentioned in Sect. 2, different kernel search algorithms exist [6, 8, 14, 15]. For this proof of concept, we included a hyperparameter optimization (no change of the GP’s kernel expression) after a detected change point as a comparison partner (*CPD HPO*). This approach also performs its retraining on the current window exclusively.

The two evaluation criteria we consider in this work are runtime and the mean absolute error of the prediction. Regarding the former, we measured the total runtime for processing each dataset, excluding the initial model construction as it is identical for all comparison partners. With respect to the prediction performance, we calculated the mean absolute error $\frac{1}{N-w} \sum_{i=w+1}^N |y_i - \hat{y}_i|$ on every step after the initial model construction. Finally, we set the results in relation to all comparison partners. Thereby, a negative value represents a shorter runtime respective more accurate modelling of ETKA, i.e. an improvement.

For all experiments, we use a window size w equivalent to 20% of the whole dataset. We allow kernels consisting of up to three base kernels and employ one iteration of adjustments when using AKS. The base kernel set B consists of the periodic, the linear and the squared exponential kernel. Before the initial kernel search with CKS, the data is rescaled using a Z-normalization.

5 Experimental Results

In this section, we provide an overview of our experimental results, both on simulated as well as on real-world data, and discuss our findings.

5.1 Simulated Data

As outlined in Sect. 4.1, we conducted experiments based on simulated data to evaluate ETKA under controlled and predefined settings. We further considered different configurations of the online CPD integrated in ETKA. These differ with respect to the defining parameters, i.e. the tolerance factor δ and the threshold ε . With the former, one can control how strict ETKA’s CPD is, i.e. a higher value increases the tolerance range for deviations between real and prediction values. A change point is declared, if the threshold ε is exceeded, so a higher value allows higher change point scores and leads to a less sensitive CPD.

We show detailed results with absolute evaluation values for all simulated datasets and CPD configurations in Figs. 2 and 3 in Appendix 3. In Table 1, we provide a summary of these results. Besides the absolute evaluation values of ETKA, all results are shown in relation to the comparison partners *PER AKS* and *CPD HPO*. The table shows averaged values over all simulated datasets and its standard deviations. As *CPD HPO* also differs based on δ and ε , the values of the relative comparison with ETKA cannot be compared across CPD settings, but give an impression which algorithm is in advantage for the specific CPD.

We observe that ETKA constantly outperforms both comparison partners with respect to the prediction error. The predictions of ETKA tend to be more accurate with a more sensitive CPD. In comparison with *PER AKS*, ETKA shows the largest improvement for $\delta = 0.5$ and $\varepsilon = 5.0$. With respect to *CPD HPO*, ETKA’s top result is achieved for $\delta = 0.5$ as well, but with $\varepsilon = 7.0$. In terms of the absolute prediction error of ETKA, we see that the best overall result is achieved for $\delta = 0.5$ and $\varepsilon = 5.0$. Regarding the runtime, *CPD HPO* is in all cases more computationally efficient, while ETKA is faster or at least on a par with *PER AKS*. As expected, we observe that the runtime of ETKA decreases with a less sensitive CPD due to higher values for δ and ε . This decrease is larger for an increasing δ with a constant ε than the other way round.

Assessing Fig. 2 in Appendix 3 for the best performing CPD configuration with $\delta = 0.5$ and $\varepsilon = 5.0$, we observe that ETKA outperforms its comparison partners on the majority of the simulated datasets (*PER AKS* is best in 7 and *CPD HPO* in 4 out of 24 simulation settings). Furthermore, in cases for which ETKA does not deliver the best outcome, it is generally close to the top performer. We further see that results for lower noise levels tend to be better (scenarios with a variable noise are also less noisy than those with $s = 0.05$ as the maximum of s is 0.05 for them).

The main goal of ETKA is an up-to-date GP model at all times. For that reason, the prediction error is more important than the runtime. Consequently, we set δ to 0.5 and ε to 5.0 at the cost of longer runtimes for the experiments on real-world data.

Table 1. Summary of the results on simulated data. The table shows the results of ETKA in terms of the prediction error respective the runtime, as well as both evaluation values of ETKA in relation to *PER AKS* and *CPD HPO*. All results are given for four different configurations regarding the CPD with different tolerance factors δ and thresholds ε . Each cell shows the mean and standard deviation over all simulated datasets. For the absolute ETKA results, a smaller value is better, both for the prediction error and the runtime. For the relative values, a negative value reflects an improvement by ETKA. All cases for which ETKA outperforms its comparison partners are highlighted in bold.

δ	ε	ETKA results		Runtime vs.		Prediction error vs.	
		Prediction error	Runtime [s]	<i>PER AKS</i>	<i>CPD HPO</i>	<i>PER AKS</i>	<i>CPD HPO</i>
0.5	5.0	0.1403 (± 0.1120)	1336 (± 385)	+0.96% ($\pm 28.70\%$)	+114.71% ($\pm 105.65\%$)	-16.68% ($\pm 23.42\%$)	-16.37% ($\pm 20.73\%$)
0.5	7.0	0.1413 (± 0.1117)	1121 (± 316)	-10.79% ($\pm 28.88\%$)	+92.25% ($\pm 80.69\%$)	-13.95% ($\pm 23.33\%$)	-21.40% ($\pm 23.13\%$)
0.7	5.0	0.1472 (± 0.1137)	922 (± 288)	-24.86% ($\pm 29.07\%$)	+56.84% ($\pm 58.70\%$)	-6.40% ($\pm 32.91\%$)	-21.02% ($\pm 24.28\%$)
0.7	7.0	0.1522 (± 0.1110)	919 (± 317)	-28.21% ($\pm 24.83\%$)	+49.07% ($\pm 47.07\%$)	-1.27% ($\pm 33.12\%$)	-20.17% ($\pm 22.00\%$)

5.2 Real-World Data

Besides simulated data, we evaluated ETKA on real-world data from different domains (see Sect. 4.2). An overview of the results with absolute evaluation values is shown in Fig. 4 in Appendix 3. In Table 2, we provide the comparison of ETKA with *PER AKS* and *CPD HPO*. On average, ETKA outperforms the comparison partners in terms of the prediction error by 2.73% and 6.19%, respectively. Considering the individual datasets, ETKA is more accurate than *PER AKS* and *CPD HPO* in 9 out of 14 respective 10 out of 14 cases. For two respectively three cases, ETKA is only slightly outperformed. Both comparison partners deliver better predictions than ETKA for *Unemployment*. Beyond that, *PER AKS* is the top performer for *Internet* and *Gas production*.

We observe the largest improvement of ETKA over both comparison partners in terms of the prediction error for *Airline*, which we show in Fig. 1a. ETKA detected two change points, so less refittings than for *PER AKS* were employed. Furthermore, the reconfigurations that ETKA performed are closer to the actual changes in the dataset, which leads to advantages regarding the prediction error. For *Gas production* shown in Fig. 1b, ETKA performs significantly better than *CPD HPO*, but is outperformed by *PER AKS*. By chance, the periodical refitting points of *PER AKS* are accurate, leading to better predictions.

Regarding the runtime overall, both *PER AKS* and *CPD HPO* are more efficient, with the latter requiring the lowest runtimes as expected. Furthermore, this disadvantage of ETKA is clearer for the real-world data than it was for simulated data. However, when focusing on the datasets for which the runtime of ETKA is more than 100% higher than for *PER AKS* (*Wheat*, *Internet*, *Radio* and *Airquality*), we observe lower prediction errors in three out of four cases.

Table 2. Results overview on real-world data. The table shows the results of ETKA in terms of runtime respective prediction error in relation to *PER AKS* and *CPD HPO*. A negative value reflects an improvement by ETKA, both for the runtime and the prediction error. All cases for which ETKA outperforms its comparison partners are highlighted in bold.

Dataset	Runtime vs.		Prediction error vs.	
	<i>PER AKS</i>	<i>CPD HPO</i>	<i>PER AKS</i>	<i>CPD HPO</i>
<i>Solar irradiance</i>	+62.51%	+918.00%	+1.90%	− 1.31%
<i>Mauna Loa</i>	− 86.96%	− 0.91%	− 1.07%	+0.00%
<i>Airline</i>	+0.79%	+2059.33%	− 21.46%	− 26.37%
<i>Wheat</i>	+161.54%	+1327.28%	− 2.83%	+2.15%
<i>Temperature</i>	− 31.83%	+159.92%	− 0.57%	− 6.66%
<i>Internet</i>	+134.97%	+183.77%	+12.51%	− 19.76%
<i>Call centre</i>	+48.67%	+1811.77%	− 2.39%	− 1.96%
<i>Radio</i>	+228.26%	+2458.84%	− 19.30%	− 7.51%
<i>Gas production</i>	+16.29%	+531.39%	+8.66%	− 23.77%
<i>Sulphuric</i>	+23.14%	+631.26%	− 6.32%	− 11.77%
<i>Unemployment</i>	− 1.81%	+760.48%	+7.21%	+15.44%
<i>Births</i>	− 38.92%	+79.1%	− 2.47%	− 1.46%
<i>Wages</i>	− 59.78%	+129.56%	+1.27%	+0.41%
<i>Airquality</i>	+130.22%	+37.77%	− 13.44%	− 4.19%
Summary	+41.93% (± 91.77%)	+792.03% (± 819.75%)	− 2.73% (± 9.84%)	− 6.19% (± 11.16%)

In comparison with *PER AKS*, *Radio* and *Airquality* are within the three datasets with the largest improvement on the prediction error. *Internet*, which already was noticeable with respect to the prediction error, is also problematic regarding the runtime.

5.3 Discussion

With respect to synthetic data, we overall observe a broad applicability of ETKA. Despite different change point patterns and noise levels, ETKA mostly outperforms its comparison partners. We further observe that ETKA does not perform worse in case of multiple changes in comparison with single changes. This indicates that the CPD within ETKA delivers the intended results and is applicable on data with multiple change points. Simulation settings *B* and *C* (see Table 3 and Fig. 2 in the Appendix) lead to the worst results for all three noise levels, while ETKA performs best for four of these six datasets. Setting *B* triggers a slow change of the periodicity. In contrast, settings *E* and *H* contain abrupt shifts of the periodical length, for which all three prediction models show lower prediction errors. Furthermore, the slow change of the amplitude *a* for setting *C* is problematic, whereas an abrupt shift of *a* for setting *H* is captured better

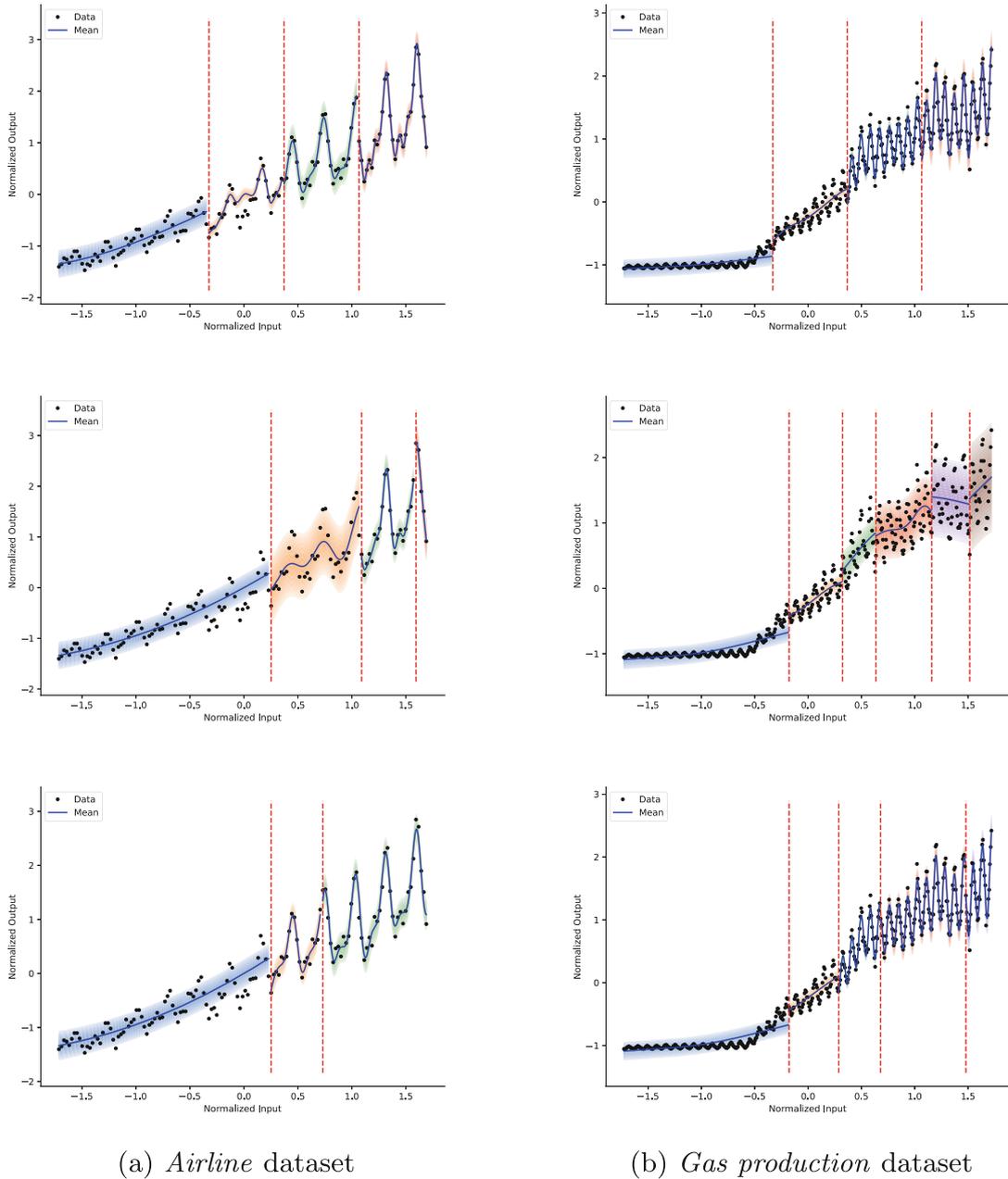


Fig. 1. Comparison of the model predictions and segmentation by the three different approaches for *Airline* and *Gas production*. The top-most plot shows the results of *PER AKS*, the second row contains *CPD HPO*'s results and the final row *ETKA*'s outcome. Each plot shows the observed data, mean prediction of the GP model as well as the confidence intervals. Points at which the model is refitted are marked with a red vertical line. The confidence interval of the GP is shown in separate colors for each segment for visual clarity.

by all methods. This lets us assume that slow changes are problematic for all three prediction models. A potential explanation for this phenomenon for both CPD-based approaches (*ETKA* and *CPD HPO*) is that abrupt changes increase the change point score s faster. Consequently, this might lead to quicker model adjustments. In contrast to abrupt changes, slow shifts can lead to prolonged inaccuracies without triggering the model adjustment.

On real-world data, ETKA still delivered the best performance in terms of the mean absolute error. However, it comes at the cost of higher runtimes compared to both alternative approaches. The primary goal of ETKA is to deliver an up-to-date model at all times. As we therefore chose a rather sensitive CPD configuration based on the results on simulated data, higher runtimes were expected. A notable exception is the *Mauna Loa* dataset, for which ETKA had the lowest runtime. The prediction error comparison to *CPD HPO* let us infer that no change point was detected. Furthermore, the slightly lower prediction error of ETKA in comparison with *PER AKS* indicates that model adjustments are not beneficial for *Mauna Loa*. Hence, it might be valid that ETKA does not detect any change points. On the other datasets, the runtime difference varies greatly, while the performance improvement is, albeit not constant, more stable. For *Airline*, *Radio* and *Airquality*, we observe the highest improvement in terms of the prediction error. As outlined, ETKA detected less change points more accurately than its comparison partners for the former. In contrast to the improved prediction error, *Radio* and *Airquality* lead to higher runtimes for ETKA. Both datasets contain several change points, which were detected by ETKA and lead to multiple model adjustments. Consequently, ETKA delivered more accurate GP models, however at the cost of computational resources. Periodical refittings highly depend on coincidence regarding an appropriate timing of model reconfigurations, as for instance observed for *Gas production*. Not depending on a by chance well chosen time for refitting is a big advantage of ETKA’s CPD-based approach. With respect to *Internet*, we observe both a higher runtime as well as prediction error for ETKA in comparison with *PER AKS*. This is probably caused by poor results of the initial kernel search using CKS, indicating data that would require a higher complexity of the kernel expression. For such data, AKS and consequently ETKA might intuitively be advantageous as its search is based on the current kernel expression instead of restarting from scratch.

In settings focused on fast processing, ETKA has multiple options to trade potential prediction performance for lowered computational costs, e.g. by employing a smaller window size w and a reduced set of base kernels. In contrast, more iterations of AKS per adjustment or a more sensitive CPD can increase the model quality at the cost of longer processing times. The effect of the CPD configuration can be seen in our results on simulated data, cf. Table 1. As expected, settings with higher values for δ and ϵ lead to lower runtimes at the cost of a higher prediction error.

ETKA’s performance is highly dependent on the integrated CPD and consequently on its parameters. For this study, the CPD parameters were determined using simulated data. Despite having generated a broad variety of change point patterns, this might not lead to the optimal parameters for all settings. Therefore, future work enabling a parameter determination based on the processed dataset could improve ETKA and the evaluation of other state-of-the-art CPD approaches is an interesting point for future research. Beyond that, for severe changes of the data behavior, a kernel search from scratch might be more efficient than AKS. A classification of detected change points could enable ETKA to always choose the most appropriate kernel search approach. A further

potential improvement could be a dynamically determined window size w in contrast to the fixed value we applied. This could be beneficial both in terms of efficiency and prediction performance.

6 Conclusion

In this paper, we enhanced AKS, a recently-introduced kernel search algorithm for GPs with a novel CUSUM-based CPD approach. The resulting algorithm, ETKA, offers the ability to automatically deliver an up-to-date GP model for data streams. In our experiments, ETKA proved to be broadly applicable to data of different behavior and noise levels. Compared to intuitive alternatives, ETKA delivered improved predictions. Especially on simulated data, the results were significantly better. On real-world datasets, the improvement was smaller. Overall, ETKA reached its main goal of an up-to-date model at all times and is therefore especially well suitable for applications for which an accurate modelling is paramount.

Funding Information. This research was supported by the research training group “Dataninja” (Trustworthy AI for Seamless Problem Solving: Next Generation Intelligence Joins Robust Data Analysis) funded by the German federal state of North Rhine-Westphalia.

The project is supported in parts by funds of the Federal Ministry of Food and Agriculture (BMEL), Germany based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support program [grant number 2818504A18, DGG].

Appendix 1: Background

In the following, we formally introduce the foundations of ETKA, namely GPs and the AKS algorithm.

A GP is a non-parametric probabilistic machine learning model [9, 17]. A model $GP(m, k)$ is uniquely defined by its mean function $m : \mathbb{R} \rightarrow \mathbb{R}$ and its covariance function or kernel $k : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$. While the mean function can often be set to constant zero [17], the kernel contains assumptions about the GP’s behavior. There are various kernels that are well understood and describe specific patterns, like the periodic \mathcal{K}_{PER} and the linear kernel \mathcal{K}_{LIN} . These simple kernels will be referred to as base kernels throughout this paper.

Kernels often depend on parameters such as a lengthscale or a period length. These parameters are referred to as hyperparameters of the GP model and can be (locally) optimized for given data. One possible measure of performance for such optimization is the GP’s log marginal likelihood $\mathcal{L}(GP(m, k), \mathbf{D})$ on the data $\mathbf{D} = (x_i, y_i)_{i=1, \dots, N}$ [17]. While there exist alternative measures [8, 17], we will use the log marginal likelihood for model optimization in our experiments.

Individual base kernels can be combined to more complex kernel expression via addition or multiplication [8]. This way, a kernel that optimally describes the data’s behavior can be constructed by experts. Alternatively, an algorithmic

approach to find such an optimal kernel expression automatically can be employed. An example for this type of algorithm is CKS [8], which is depicted in Algorithm 2.

Algorithm 2: Pseudocode of the Compositional Kernel Search

Data: $D = (x_i, y_i)_{i=1, \dots, N}$, base kernel set B , max iterations i_{max}
Result: Kernel expression K

- 1 $K \leftarrow \operatorname{argmax}_{b \in B} (\mathcal{L}(GP(0, b), D))$
- 2 **for** $i = 1, \dots, i_{max}$ **do**
- 3 $C \leftarrow \operatorname{AddViaAddition}(K, B)$
- 4 $C \leftarrow C \cup \operatorname{AddViaMultiplication}(K, B)$
- 5 $C \leftarrow C \cup \operatorname{ReplaceKernel}(K, B)$
- 6 $K \leftarrow \operatorname{argmax}_{c \in C} (\mathcal{L}(GP(0, c), D))$

In each iteration, the algorithm adjusts the current best kernel expression K given a set of base kernels B by

1. adding any base kernel to any subexpression of K ,
2. multiplying any base kernel to any subexpression of K or
3. replacing any base kernel in K with a different base kernel from B .

The abstract functions *AddViaAddition*, *AddViaMultiplication* and *ReplaceKernel* in Algorithm 2 correspond to these three options. The best performing candidate from the thus generated set is used as the basis for the next iteration. This way, the algorithm can build arbitrarily complex kernels at the cost of multiple model optimizations and evaluations per iteration. Since the candidate generation can not lower the kernel's complexity, any searches for new kernels need to start from scratch.

Our goal is to model consecutive segments of potentially infinite time series. We utilize the AKS algorithm [13] as it has inherent advantages in this specific setting. In particular, the AKS algorithm is able to adjust a given kernel to fit new data instead of starting from zero. This is accomplished by adding a fourth possibility in the candidate generation: the removal of a base kernel from the current expression. This procedure is depicted Algorithm 3.

Algorithm 3: Pseudocode of the Adjusting Kernel Search

Data: $D = (x_i, y_i)_{i=1, \dots, N}$, base kernel set B , max iterations i_{max} , starting kernel K_0
Result: Kernel expression K

- 1 $K \leftarrow K_0$
- 2 **for** $i = 1, \dots, i_{max}$ **do**
- 3 $C \leftarrow \operatorname{AddViaAddition}(K, B)$
- 4 $C \leftarrow C \cup \operatorname{AddViaMultiplication}(K, B)$
- 5 $C \leftarrow C \cup \operatorname{ReplaceKernel}(K, B)$
- 6 $C \leftarrow C \cup \operatorname{RemoveKernel}(K)$
- 7 $K \leftarrow \operatorname{argmax}_{c \in C} (\mathcal{L}(GP(0, c), D))$

It has been shown before that AKS can lead to a faster modelling process than CKS [13]. Especially in high-complexity models, the computational cost of constructing a kernel from zero are much higher than a few iterations of adjustment. However, for low-complexity models, the larger set of candidates in AKS can lead to longer processing compared to CKS, even if fewer iterations are needed.

Appendix 2: Simulated and Real-World Data Characteristics

Table 3. Overview of simulated datasets. Artificial datasets were generated at three different noise levels. For the third setting, the standard deviation s changed for each η_i . Various scenarios with single and multiple changes of the time series components as well as several configurations regarding the type of the change were simulated. We considered both abrupt as well as slower occurring changes.

		$\mathbf{y_1}$			w_{fade}	$\mathbf{y_2}$			w_{fade}	$\mathbf{y_3}$			w_{fade}	$\mathbf{y_4}$		
		\mathbf{b}		\mathbf{l}		\mathbf{b}		\mathbf{l}		\mathbf{b}		\mathbf{l}		\mathbf{b}		\mathbf{l}
		n_{per}	a	m		n_{per}	a	m		n_{per}	a	m		n_{per}	a	m
η with $s = 0.01$	$A_{0.01}$	100	0.1	–	500	100	0.1	1								
	$B_{0.01}$	100	0.1	–	[600, 650]	200	0.1	–								
	$C_{0.01}$	100	0.1	–	[1000, 1050]	100	0.2	–								
	$D_{0.01}$	100	0.1	1	[500, 550]	100	0.1	5								
	$E_{0.01}$	100	0.1	1	500	200	0.1	2	1250	200	0.1	–				
	$F_{0.01}$	100	0.1	–	500	100	0.2	–	1250	200	0.2	–				
	$G_{0.01}$	100	0.1	4	500	100	0.1	1	1250	100	0.1	–0.1	1500	100	0.1	0.5
	$H_{0.01}$	100	0.1	2	500	200	0.1	0.5	1250	200	0.2	0.5	1500	100	0.1	0.2
η with $s = 0.05$	$A_{0.05}$	100	0.1	–	500	100	0.1	1								
	$B_{0.05}$	100	0.1	–	[600, 650]	200	0.1	–								
	$C_{0.05}$	100	0.1	–	[1000, 1050]	100	0.2	–								
	$D_{0.05}$	100	0.1	1	[500, 550]	100	0.1	5								
	$E_{0.05}$	100	0.1	1	500	200	0.1	2	1250	200	0.1	–				
	$F_{0.05}$	100	0.1	–	500	100	0.2	–	1250	200	0.2	–				
	$G_{0.05}$	100	0.1	4	500	100	0.1	1	1250	100	0.1	–0.1	1500	100	0.1	0.5
	$H_{0.05}$	100	0.1	2	500	200	0.1	0.5	1250	200	0.2	0.5	1500	100	0.1	0.2
η_i with $s_{max} = 0.05$	A_{var}	100	0.1	–	500	100	0.1	1								
	B_{var}	100	0.1	–	[600, 650]	200	0.1	–								
	C_{var}	100	0.1	–	[1000, 1050]	100	0.2	–								
	D_{var}	100	0.1	1	[500, 550]	100	0.1	5								
	E_{var}	100	0.1	1	500	200	0.1	2	1250	200	0.1	–				
	F_{var}	100	0.1	–	500	100	0.2	–	1250	200	0.2	–				
	G_{var}	100	0.1	4	500	100	0.1	1	1250	100	0.1	–0.1	1500	100	0.1	0.5
	H_{var}	100	0.1	2	500	200	0.1	0.5	1250	200	0.2	0.5	1500	100	0.1	0.2

Table 4. Overview of the real-world data for our experiments. We considered 14 datasets from various domains and show some characteristics below. Most datasets were obtained from Lloyd et al. [15]. The *Airquality* data was published by De Vito et al. [7].

Dataset	Length	Mean	Std	Min	Max
<i>Solar irradiance</i>	391	1365.82	0.24	1365.52	1366.68
<i>Mauna Loa</i>	702	352.30	26.18	313.21	407.65
<i>Airline</i>	144	280.30	119.97	104.00	622.00
<i>Wheat prices</i>	370	107.88	67.21	11.00	381.00
<i>Temperature</i>	1000	11.21	4.14	-0.80	26.30
<i>Internet</i>	1000	46355.45	22058.93	13486.74	125058.79
<i>Call centre</i>	180	492.50	189.54	161.00	872.00
<i>Radio</i>	240	8.08	2.44	4.30	13.50
<i>Gas production</i>	476	21415.27	18678.34	1646.00	66600.00
<i>Sulphuric production</i>	462	131.34	41.26	42.00	228.00
<i>Unemployment</i>	408	520.28	261.22	122.00	1350.00
<i>Births</i>	1000	248.53	42.06	136.00	366.00
<i>Wages</i>	735	8.76	8.18	2.15	49.99
<i>Airquality</i>	7718	246.90	212.98	2.00	1479.00

Appendix 3: Results Overviews

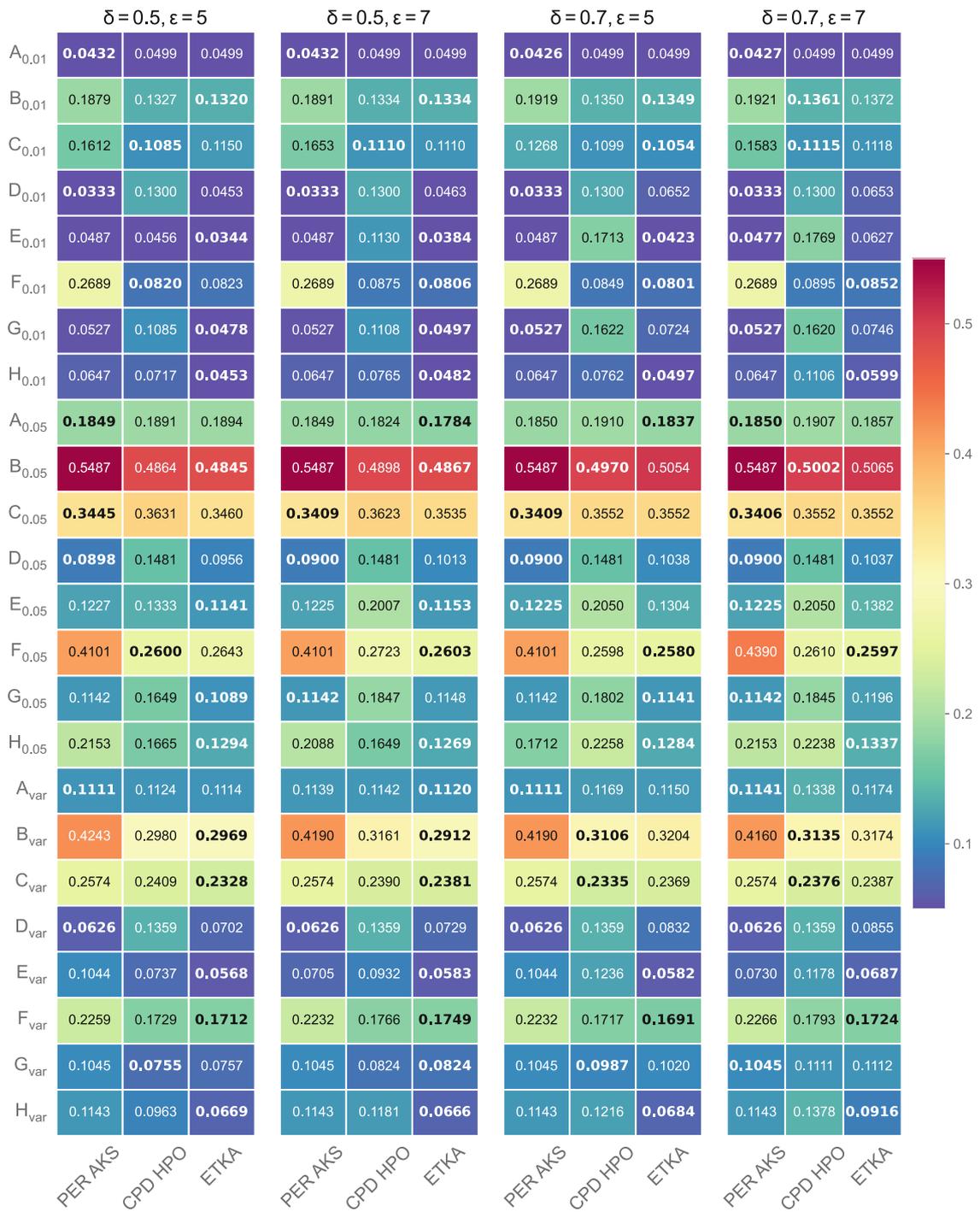


Fig. 2. Results overview in terms of the prediction error on all simulated data and CPD configurations. Each cell shows the result for the model given on the horizontal axis and the simulated dataset given on the vertical axis. Smaller values reflect a better performance. The best performing model for each CPD configuration and dataset is highlighted in bold.

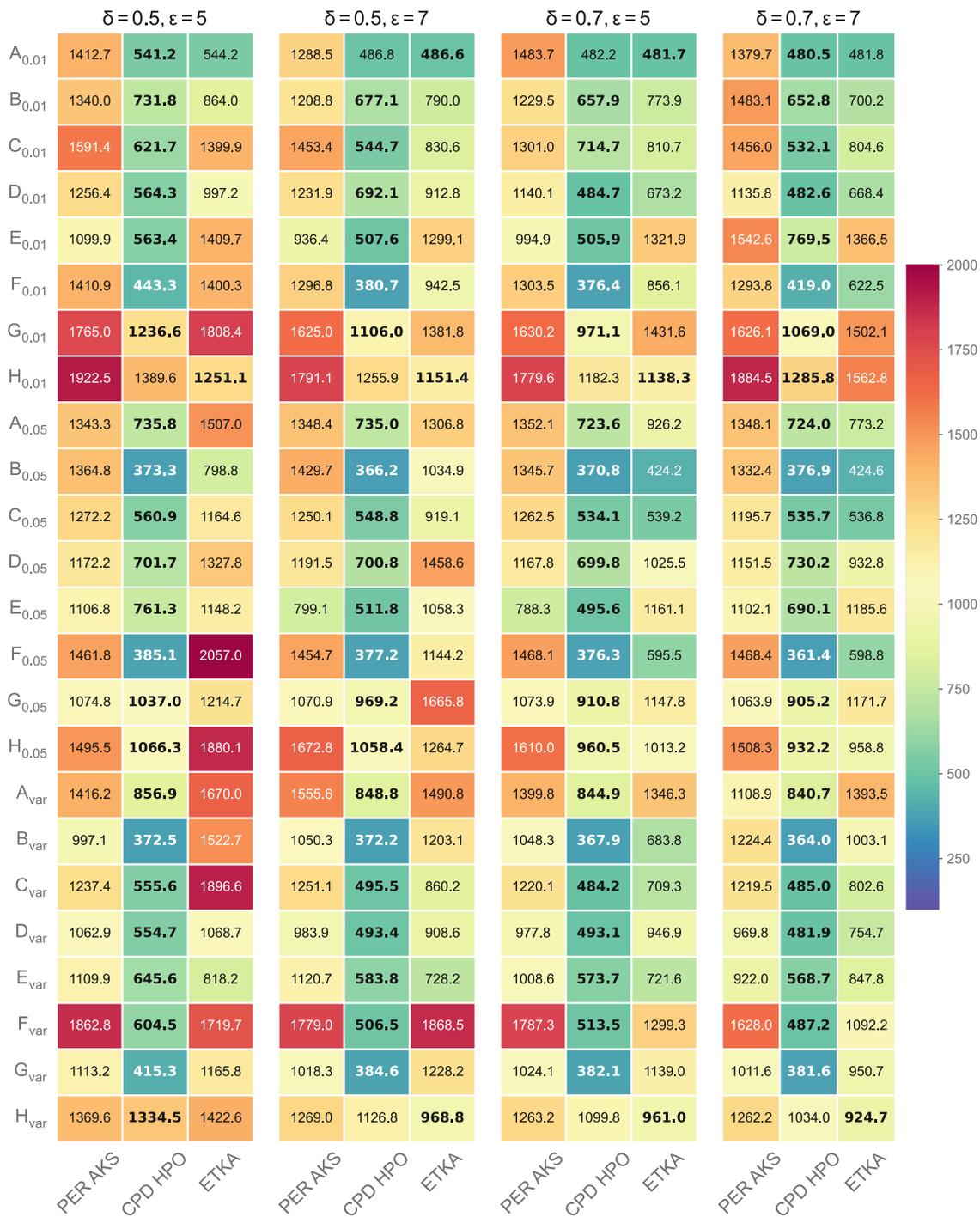


Fig. 3. Results overview in terms of runtime on all simulated data and CPD configurations. Each cell shows the result for the model given on the horizontal axis and the simulated dataset given on the vertical axis. Smaller values are considered better. The most efficient model for each CPD configuration and dataset is highlighted in bold.

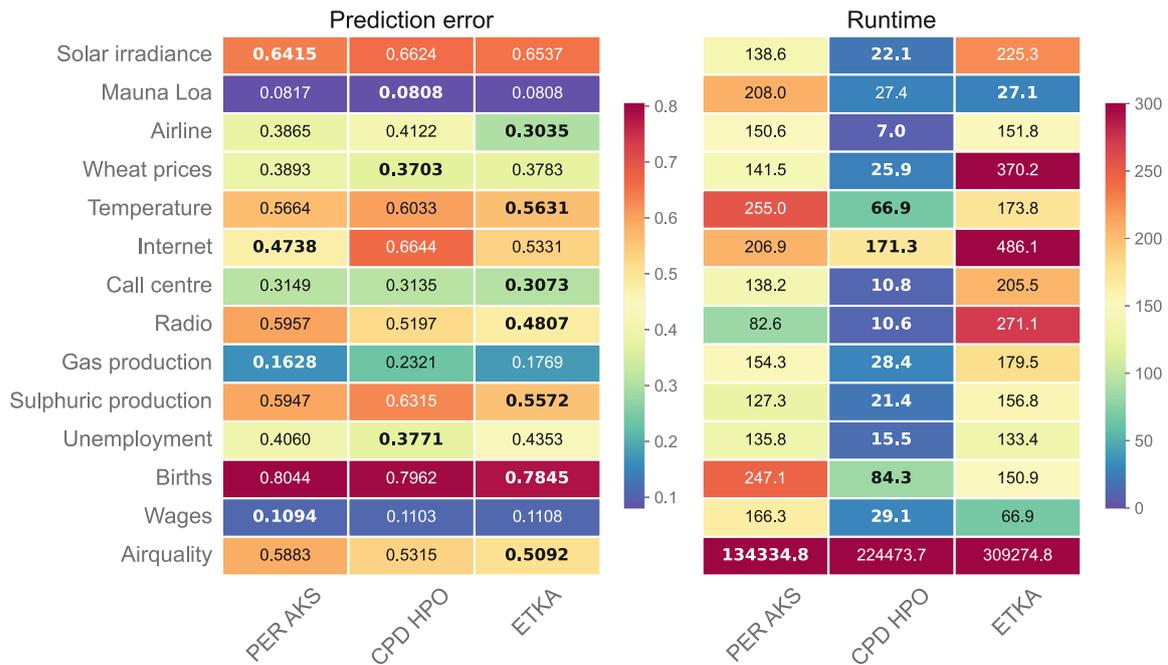


Fig. 4. Results overview for real-world data. Each cell shows the result for the model given on the horizontal axis and the dataset given on the vertical axis. Smaller values reflect a better performance. The best performing respective most efficient model for each dataset is highlighted in bold.

References

1. Adams, R.P., MacKay, D.J.C.: Bayesian online changepoint detection (2007)
2. Alagu Dharshini, M.P., Antelin Vijila, S.: Survey of machine learning and deep learning approaches on sales forecasting. In: 2021 4th International Conference on Computing and Communications Technologies (ICCCT), pp. 59–64 (2021). <https://doi.org/10.1109/ICCCT53315.2021.9711878>
3. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. *Knowl. Inf. Syst.* **51**(2), 339–367 (2016). <https://doi.org/10.1007/s10115-016-0987-z>
4. Bahri, M., Bifet, A., Gama, J., Gomes, H.M., Maniu, S.: Data stream analysis: foundations, major tasks and tools. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **11**(3), e1405 (2021)
5. Berns, F., Beecks, C.: Automatic gaussian process model retrieval for big data. In: CIKM, pp. 1965–1968. ACM (2020)
6. Berns, F., Schmidt, K., Bracht, I., Beecks, C.: 3cs algorithm for efficient gaussian process model retrieval. In: ICPR, pp. 1773–1780. IEEE (2020)
7. De Vito, S., Massera, E., Piga, M., Martinotto, L., Di Francia, G.: On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario. *Sens. Actuators B: Chem.* **129**(2), 750–757 (2008). <https://doi.org/10.1016/j.snb.2007.09.060>
8. Duvenaud, D., Lloyd, J.R., Grosse, R.B., Tenenbaum, J.B., Ghahramani, Z.: Structure discovery in nonparametric regression through compositional kernel search. In: ICML (3). *JMLR Workshop and Conference Proceedings*, vol. 28, pp. 1166–1174. JMLR.org (2013)

9. Ghahramani, Z.: Probabilistic machine learning and artificial intelligence. *Nature* **521**(7553), 452–459 (2015)
10. Haselbeck, F., Grimm, D.G.: EVARS-GPR: EVent-triggered augmented refitting of gaussian process regression for seasonal data. In: Edelkamp, S., Möller, R., Rueckert, E. (eds.) *KI 2021. LNCS (LNAI)*, vol. 12873, pp. 135–157. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87626-5_11
11. Haselbeck, F., Killinger, J., Menrad, K., Hannus, T., Grimm, D.G.: Machine learning outperforms classical forecasting on horticultural sales predictions. *Mach. Learn. Appl.* **7**, 100239 (2022). <https://doi.org/10.1016/j.mlwa.2021.100239>
12. Hernandez, L., et al.: A survey on electric power demand forecasting: future trends in smart grids, microgrids and smart buildings. *IEEE Commun. Surv. Tutorials* **16**(3), 1460–1495 (2014). <https://doi.org/10.1109/SURV.2014.032014.00094>
13. Hüwel, J.D., Berns, F., Beecks, C.: Automated kernel search for gaussian processes on data streams. In: *IEEE BigData*, pp. 3584–3588. IEEE (2021)
14. Kim, H., Teh, Y.W.: Scaling up the automatic statistician: scalable structure discovery using gaussian processes. In: *AISTATS. Proceedings of Machine Learning Research*, vol. 84, pp. 575–584. PMLR (2018)
15. Lloyd, J., Duvenaud, D., Grosse, R., Tenenbaum, J., Ghahramani, Z.: Automatic construction and natural-language description of nonparametric regression models. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28 (2014)
16. Page, E.S.: Continuous inspection schemes. *Biometrika* **41**(1/2), 100–115 (1954)
17. Rasmussen, C.E., Williams, C.K.I.: *Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning*, MIT Press, Cambridge (2006)
18. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. *Sig. Process.* **167**, 107299 (2020)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter’s Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter’s Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.



Appendix D

Publication D

The subsequent pages show the original full version of the following publication:

PUBLICATION D:

Josef Eiglsperger*, **Florian Haselbeck*** and Dominik G. Grimm (2023). ForeTiS: A comprehensive time series forecasting framework in Python. *Machine Learning with Applications*, 12, 100467. <https://doi.org/10.1016/j.mlwa.2023.100467>

**Both authors contributed equally and share first authorship.*



Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

ForeTiS: A comprehensive time series forecasting framework in Python

Josef Eiglsperger^{a,b,1}, Florian Haselbeck^{a,b,1}, Dominik G. Grimm^{a,b,c,*}^a Technical University of Munich, Campus Straubing for Biotechnology and Sustainability, Bioinformatics, Schulgasse 22, Straubing, 94315, Germany^b Weihenstephan-Triesdorf University of Applied Sciences, Bioinformatics, Petersgasse 18, Straubing, 94315, Germany^c Technical University of Munich, TUM School of Computation, Information and Technology (CIT), Boltzmannstraße 3, Garching, 85748, Germany

ARTICLE INFO

Dataset link: <https://github.com/grimmlab/ForeTiS>

Keywords:

Time series forecasting
Regression
Machine learning
Scientific computing

ABSTRACT

Summary: Time series forecasting is a research area with applications in various domains, nevertheless without yielding a predominant method so far. We present ForeTiS, a comprehensive and open source Python framework that allows rigorous training, comparison, and analysis of state-of-the-art time series forecasting approaches. Our framework includes fully automated yet configurable data preprocessing and feature engineering. In addition, we use advanced Bayesian optimization for automatic hyperparameter search. ForeTiS is easy to use, even for non-programmers, requiring only a single line of code to apply state-of-the-art time series forecasting. Various prediction models, ranging from classical forecasting approaches to machine learning techniques and deep learning architectures, are already integrated. More importantly, as a key benefit for researchers aiming to develop new forecasting models, ForeTiS is designed to allow for rapid integration and fair benchmarking in a reliable framework. Thus, we provide a powerful framework for both end users and forecasting experts.

Availability: ForeTiS is available at <https://github.com/grimmlab/ForeTiS>. We provide a setup using Docker, as well as a Python package at <https://pypi.org/project/ForeTiS/>. Extensive online documentation with hands-on tutorials and videos can be found at <https://foretis.readthedocs.io>.

1. Introduction

Time series forecasting is a research area with diverse applications, such as predicting product demand, energy consumption, or health status. Several forecasting competitions, including classical forecasting and machine learning (ML) techniques, have not resulted in a dominant method, although recent publications show advantages for ML-based approaches (Bojer & Meldgaard, 2021; Haselbeck et al., 2022; Huber & Stuckenschmidt, 2020; Jiao & Chen, 2019; Makridakis et al., 2020, 2022). Despite the need to re-evaluate different prediction models for each time series forecasting task, no comprehensive time series forecasting framework is publicly available.

As Meisenbacher et al. (2022) conclude in a recent review paper, existing frameworks such as Bauer et al. (2020), Züfle and Kounev (2020), and Martínez et al. (2019) cover only parts of the typical time series forecasting pipeline consisting of data preprocessing, feature engineering, hyperparameter optimization, and model selection. In addition, many packages such as statsmodels (Seabold & Perktold, 2010), scikit-learn (Pedregosa et al., 2011), or PyTorch (Paszke et al., 2019) are focused on a particular type of prediction model and are not explicitly designed for time series forecasting. pyWATTS has recently

been published as a Python framework aimed at automating the time series forecasting workflow (Heidrich et al., 2021). However, its use requires expert knowledge, and several lines of code must be written to run any model. Furthermore, it does not include hyperparameter optimization and mainly provides wrappers for third-party libraries. The low code machine learning library PyCaret (Moez, 2023) recently included time series forecasting but similar to sktime (Löning et al., 2019) also requires several lines of code to run a comparative study and is not available as a command line tool. PyCaret does not include deep learning frameworks, whereas sktime is focused on Keras without providing inherited general functionalities. Beyond that, sktime does not leverage Bayesian optimization for hyperparameter search.

In this paper, we present ForeTiS, a comprehensive and open source Python framework that allows for rigorous training, comparison, and analysis of different time series forecasting approaches, covering the entire time series forecasting workflow. Unlike existing frameworks, ForeTiS is easy to use, requiring only a single-line command to apply state-of-the-art time series forecasting. In addition, data preprocessing and feature engineering are configurable and fully automated, as is hyperparameter search, for which we use advanced Bayesian optimization. In terms of forecasting approaches, our

* Corresponding author at: Technical University of Munich, Campus Straubing for Biotechnology and Sustainability, Bioinformatics, Schulgasse 22, Straubing, 94315, Germany.

E-mail addresses: josef.eiglsperger@hswt.de (J. Eiglsperger), florian.haselbeck@hswt.de (F. Haselbeck), dominik.grimm@hswt.de (D.G. Grimm).

¹ These authors have contributed equally to this work and share first authorship.

<https://doi.org/10.1016/j.mlwa.2023.100467>

Received 7 March 2023; Received in revised form 4 April 2023; Accepted 6 April 2023

Available online 10 April 2023

2666-8270/© 2023 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

framework already offers three classical forecasting models and eleven ML-based methods, ranging from classical ML to modern deep learning (DL) architectures. In addition, ForeTiS is designed to facilitate the development and benchmarking of new prediction models. In summary, ForeTiS is unique compared to existing frameworks in several ways. First, ForeTiS is easy to install as a Python package and as a command line tool using Docker. Second, ForeTiS is the only framework that covers and fully automates the whole time series forecasting pipeline, already including various prediction models and only requiring a single line of code to run a comparative study. Hence, ForeTiS is applicable in a user-friendly manner, also for users that are not forecasting experts. In addition, ForeTiS employs advanced Bayesian optimization for a fully-automated hyperparameter search. More importantly, in contrast to other frameworks, ForeTiS supports the development of new prediction models by already implementing general functionalities for a variety of model types and enabling quick benchmarking.

Overall, ForeTiS is a powerful tool for improved time series forecasting. The framework simplifies the use of time series forecasting for end users while providing a reliable framework for researchers to develop new forecasting models. To support novice users, we also provide comprehensive online documentation supplemented by various hands-on tutorials and videos. In the following, we will first outline the user benefits of ForeTiS before providing information on the implementation of its main features. We then present a case study using three publicly available datasets with different characteristics and draw conclusions.

2. User benefits

ForeTiS provides several benefits to end users and researchers that want to develop new time series forecasting models. These benefits are outlined below.

2.1. Easy to use

We designed ForeTiS to provide user-friendly time series forecasting so that even non-programmers can benefit from our work. The command line interface of ForeTiS requires only a single-line command

to perform a comprehensive, reproducible study over several freely selectable forecasting models. To start the pipeline, users only need to provide a CSV file containing the data and specify dataset-specific settings via a configuration file. All data preprocessing, feature engineering, and hyperparameter optimization are fully automated, as described in Section 3. Regarding the prediction models, the user can choose from a wide range of pre-integrated models, ranging from classical forecasting methods over ML-based approaches to modern DL techniques. In addition, ForeTiS supports result analysis, e.g., via a plotting function. Our online documentation provides a hands-on tutorial on how to run ForeTiS: <https://tinyurl.com/TutorialRunDocker>. To simplify the installation, we provide a setup using Docker (Merkel, 2014), which we also explain step-by-step in our documentation at https://foretis.readthedocs.io/en/latest/install_docker.html. In addition, we released ForeTiS as a Python package, which requires only one function call to start the whole time series prediction pipeline, see <https://tinyurl.com/UseTutorialPythonPackage>.

2.2. Easy to customize

In addition to its straightforward application, ForeTiS can be customized in several ways. A variety of arguments can be passed to ForeTiS to allow the user to customize the pipeline, such as the imputation method for missing values or data splitting settings. In addition, ForeTiS can be configured using the dataset-specific configuration file. In this configuration file, the user can, for example, specify items from the provided CSV file for which statistical features should be generated. ForeTiS prevents pipeline failures due to mismatches in a user's configuration by catching them and assisting the user in resolving them with meaningful error messages. The following video tutorial shows two case studies in which we explain how to create a dataset-specific configuration file: <https://tinyurl.com/TutorialCaseStudies>. Beyond that, users can even customize an already integrated prediction model, e.g., in terms of hyperparameter ranges, as shown here: <https://tinyurl.com/TutorialAdjustModels>.

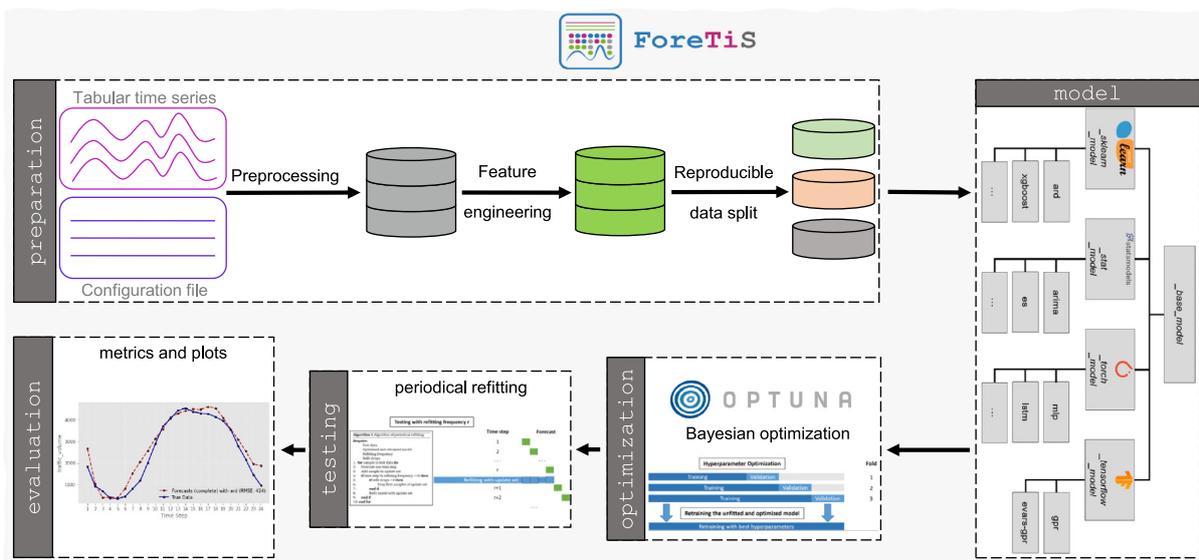


Fig. 1. Overview of the structure of ForeTiS: In preparation, we summarize the fully automated and configurable data preprocessing and feature engineering. In model, we have already integrated several time series forecasting models from which the user can choose. Furthermore, the design of this module allows for easy integration of new prediction models. We use state-of-the-art Bayesian optimization with the Python package Optuna for automated hyperparameter optimization. With the testing module, we allow the user to test different fitting procedures. Finally, we provide several methods to analyze the results in evaluation. To start the optimization pipeline, users only need to provide a CSV file containing the data and a configuration file that allows the pipeline to be customized. This design allows end users to apply time series forecasting with just a single line of code. In addition, we support researchers who want to develop new forecasting methods with quick integration into a reliable framework and benchmarking against existing approaches.

2.3. Easy to extend

ForeTiS allows advanced users to integrate new prediction models into the framework quickly. This process is especially facilitated for models based on the widely used Python packages `scikit-learn` (Pedregosa et al., 2011), `statsmodels` (Seabold & Perktold, 2010), `PyTorch` (Paszke et al., 2019), and `TensorFlow` (Abadi et al., 2016). A forecasting expert can focus on improving a new forecasting model by using our reliable and tested framework that includes fully automated data preprocessing, feature engineering, and hyperparameter search. As an additional benefit, the implemented model can be easily benchmarked against a wide range of already integrated forecasting models. Thus, ForeTiS accelerates the development of new prediction models while reducing risks and testing efforts. As a further plus, ForeTiS is available as a Python package, allowing the framework or parts of it to be integrated into existing code.

3. Implementation

ForeTiS is structured according to the common time series forecasting pipeline. In Fig. 1, we provide an overview of the main packages of our framework along the typical workflow. In the following, we outline the implementation of the main features.

3.1. Data preparation

In preparation, we summarize the fully automated yet configurable data preprocessing and feature engineering. First, the data is transformed and stored in a unified format to enable consistent handling. Since many prediction models cannot handle missing values, ForeTiS offers three imputation methods, namely mean, k-nearest-neighbors, and iterative imputation. We have also integrated Principal Component Analysis for dimensionality reduction, which can be disabled by the user or automatically selected during optimization.

In terms of feature engineering, we consider calendrical and statistical features. In addition to date-based features, such as the day of the week or month, we include special event countdowns. Special events can influence the target variable, e.g., product sales before Christmas or flower sales before Valentine’s Day. Therefore, a user can define such special events in the configuration file, for which the pipeline will determine a countdown. For statistical features, we consider lagged versions of configurable dataset items, i.e., past values, since, for example, past sales may influence future demand. For seasonal time series, a feature that applies to many time series when seasonality is considered a periodic system behavior, the values of previous seasons may be predictive. To account for this, we also include seasonally lagged features. However, calculating these features leads to missing values at the beginning of the dataset as values from previous seasons are not available. For these samples, imputation may be problematic

due to a long period without information. To reduce data loss when dropping these samples, we dynamically set the number of previous seasons considered for these seasonally lagged features depending on the length of the dataset. In addition, ForeTiS computes statistical information such as averages of past information within a user-defined window. Then we encode the non-numerical information and resample the dataset to a weekly resolution if specified in the configuration file. Finally, we drop features with constant values.

We have implemented three types of data splits for the user to choose from: train-validation-test, cross-validation with a separate test set, and time series cross-validation with an independent test set. Because many time series prediction models require a chronological order of samples, time series cross-validation with a separate test set is the default data split of ForeTiS, and the use of the other data splits is disabled for such models. In the upper part of Fig. 2, we visualize time series cross-validation using three folds. The size of the validation and test sets can be set as a percentage of the entire dataset or a certain number of seasons in a seasonal time series.

To conserve computing resources and ensure identical data, we perform all data preparation steps once and save the result in an HDF5 file for reuse. This procedure ensures comparable and reproducible results, even on different machines. Furthermore, with regular expressions specified in the configuration file, users can define subsets of features related to feature names. By default, ForeTiS starts a separate optimization pipeline for the entire featureset and these subsets. Alternatively, when running ForeTiS, the user can select the featureset to use within the optimization pipeline.

3.2. Forecasting models and extendability

We already included several time series forecasting methods in model. Besides uni- and multivariate classical forecasting methods, e.g., Exponential Smoothing (Winters, 1960), we provide several ML-based approaches. These ML methods range from classical regression models, such as Elastic Net (Zou & Hastie, 2005), over the ensemble learner XGBoost (Chen & Guestrin, 2016), to modern DL-based architectures, e.g., Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997). We also use probabilistic models, such as Gaussian Process Regression (GPR) (Williams & Rasmussen, 1995) and Bayesian implementations of neural network-based architectures (Denker & LeCun, 1990). These provide prediction uncertainties that may be useful for downstream tasks. In addition, we have integrated the recently published EVARS-GPR approach, which handles sudden changes in the data distribution and adjusts the prediction model accordingly (Haselbeck & Grimm, 2021). We summarize the integrated forecasting approaches in Table 1. Due to the lack of a universal evaluation metric for time series forecasting, it is common to evaluate performance against baseline methods (Hyndman & Koehler, 2006). For

Table 1

Overview of classical forecasting and machine learning methods already integrated into ForeTiS: We show the prediction models with the abbreviations we use in ForeTiS. We also specify whether a prediction model is multivariate, i.e., whether it includes features in addition to the time series. In addition, we have implemented probabilistic models that provide prediction uncertainties.

Category	Prediction model	Abbreviation	Multivariate	Probabilistic
Classical forecasting	Exponential Smoothing	ES		
	(Seasonal) Autoregressive Integrated Moving Average	(S)ARIMA		
	(Seasonal) Autoregressive Integrated Moving Average with eXogenous factors	(S)ARIMAX	×	
Machine learning	Elastic Net Regression	ElasticNet	×	
	Lasso Regression	LassoReg	×	
	Ridge Regression	RidgeReg	×	
	eXtreme Gradient Boosting	XGB	×	
	Automatic Relevance Determination Regression	ARD	×	×
	Bayesian Ridge Regression	BayesRidge	×	×
	Gaussian Process Regression	GPR	×	×
	Event-triggered Augmented Refitting of for Seasonal Data	EVARS-GPR	×	×
	(Bayesian) Long Short-Term Memory network	(Bayes)LSTM	×	(×)
	(Bayesian) Multilayer Perceptron	(Bayes)MLP	×	(×)

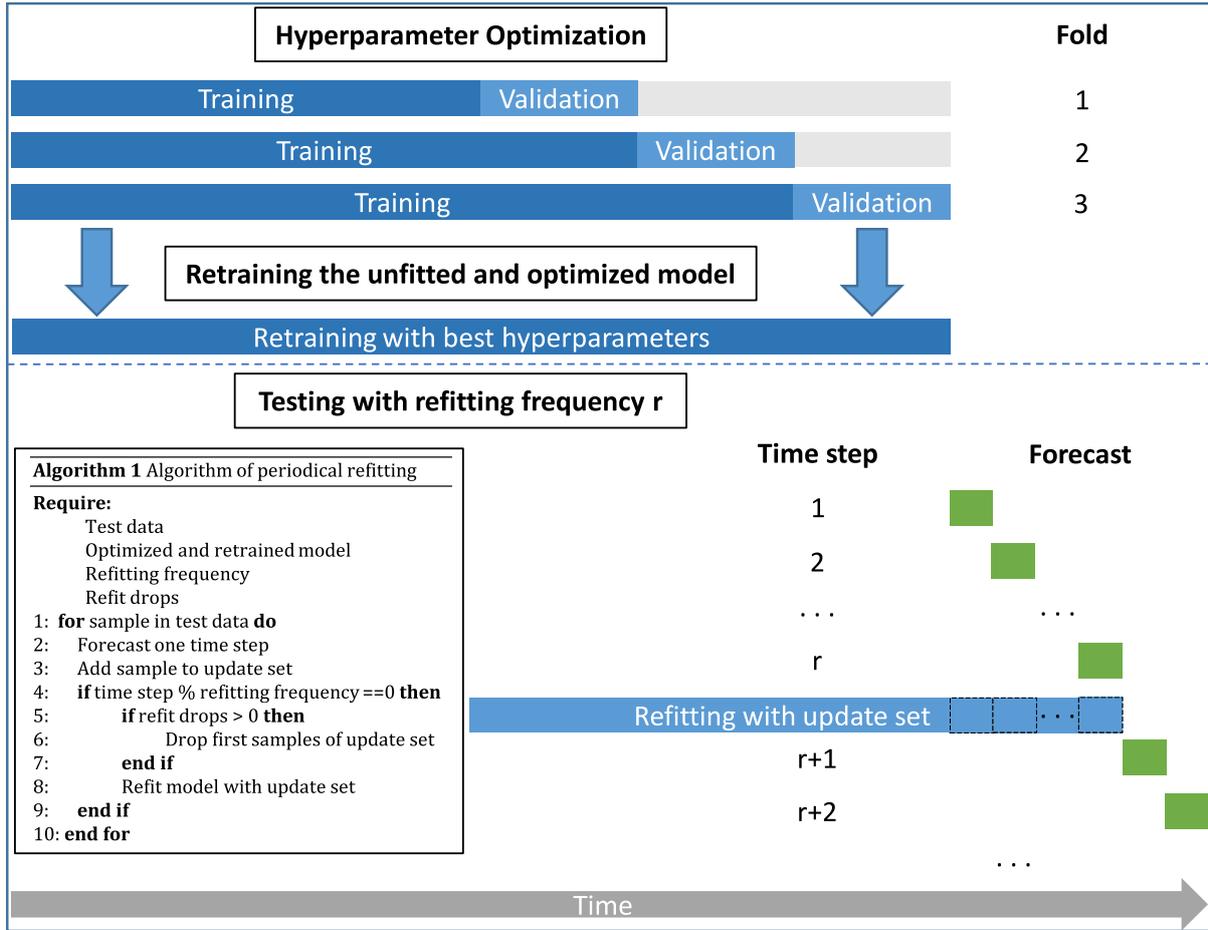


Fig. 2. Scheme for the hyperparameter optimization, exemplarily shown with a three-fold time series cross-validation and a separate test set. Based on the results averaged over the three validation sets, hyperparameters are determined that are used to retrain the prediction model using the entire dataset. The refit frequency r defines the number of predicted samples before the model is refitted with new samples. A user-defined amount of previous data is used for this refitting, dropping the oldest samples in the training set.

this purpose, ForeTiS provides the baselines listed in Table 2. Detailed explanations of all time series forecasting and baseline methods are included in our documentation: <https://foretis.readthedocs.io/en/latest/models.html>.

Table 2
Baseline methods with a predicted value \hat{y}_t and a true value y_t at time step t , a seasonal period m , and a window size w .

Average Historical	$\hat{y}_t = \frac{1}{t-1} \sum_{i=1}^{t-1} y_i$
Average Moving/Random Walk	$\hat{y}_t = \frac{1}{w} \sum_{i=t-w}^{t-1} y_i$
Average Seasonal	$\hat{y}_t = \frac{1}{m} \sum_{i=t-m}^{t-1} y_i$
Average Seasonal Lag	$\hat{y}_t = \frac{1}{w} \sum_{i=t-m-w}^{t-m-1} y_i$

ForeTiS also allows for quick integration of new forecasting models. We have created a base class in which we have already defined general methods useful for all prediction models and abstract methods needed by each child class. For the commonly used packages `scikit-learn`, `statsmodels`, `PyTorch`, and `TensorFlow`, we already implemented most of the mandatory methods, for instance, the training loops. To create a new prediction model based on one of these widely used programming libraries, a user only needs to implement two methods: the actual prediction model and the hyperparameters with the corresponding ranges. This easy-to-extend design allows model developers to focus on the model design while relying on thoroughly

tested surrounding code. Our online documentation provides a detailed tutorial on integrating a new prediction model, including a video, at <https://tinyurl.com/TutorialIntegrateNewModel>. Another benefit of using ForeTiS when developing a new forecasting model is the ability to quickly and fairly benchmark against multiple comparison partners, as the same data preparation and hyperparameter search is used.

3.3. Hyperparameter optimization

We achieve an automatic hyperparameter search by using state-of-the-art Bayesian optimization via the Python package `Optuna` (Akiba et al., 2019). Unlike grid and random search, Bayesian optimization uses information from the performance of previously tested parameter choices to suggest new parameter candidates (Snoek et al., 2012; Turner et al., 2021). Thus, only promising parameter values are considered further, potentially improving efficiency and leading to better-performing hyperparameter choices. To further improve efficiency, we also applied a pruning of selected parameter sets that did not confirm a good performance on intermediate cross-validation results. In Fig. 2, we visualize the hyperparameter search using a three-fold time series cross-validation. The best-performing hyperparameters are selected based on the results averaged over the three validation sets, and we obtain the final model after retraining on the entire training and validation data.

Table 3

Overview of the datasets used for our case study. In addition to a description, we show statistics for each dataset. All datasets differ in terms of the application domain, the number of samples, and the number of features. *MetroInterstateTraffic* contains hourly data, the other two show daily information.

Dataset	Description	Samples	Features	Time resolution	Season length	Mean	Standard deviation	Max
<i>MetroInterstateTraffic</i>	Traffic volume, weather & holidays	52551	7	Hourly	24	3291	1985	7280
<i>DailyDelhiClimate</i>	Weather forecasting for Delhi, India	1462	3	Daily	365	6.80	4.56	42.22
<i>BikeSharing</i>	Bikes rented & rental period information	731	9	Daily	365	4504	1937	8714

3.4. Testing and model refitting

A well-known problem in time series forecasting are sudden changes in system behavior that are not captured by features, e.g., a construction site on an access road when predicting stationary retail sales or an unexpected breakdown at a large factory when predicting energy consumption. A common technique to account for these changing data distributions is to refit a prediction model using new data periodically but keeping the optimized hyperparameters fixed. Therefore, to simulate the productive operation of a time series forecasting application, we integrate periodical refittings on test data in *ForeTiS*. The user can configure the amount of historical data considered and the frequency of the refittings. Fig. 2 outlines this procedure for a refitting frequency r . At each time step $1 + n \cdot r$, we predict the subsequent r samples of the test data and refit the prediction model using the updated refit set. In addition, *ForeTiS* allows selecting refit frequency 0, leading to a single refitting at the beginning of the test data using the configured amount of previous data. Further, *ForeTiS* predicts the whole test set without a refitting, referred to as ‘complete’. The EVARS-GPR algorithm mentioned above can be considered an alternative to this periodic refitting scheme, as it adjusts the prediction model based on change point detection (Haselbeck & Grimm, 2021). In addition, the easily extensible design described above allows the integration of other methods that address this challenge, such as dynamically self-adjusting GPR (Hüwel et al., 2022).

3.5. Evaluation

We include the typical evaluation metrics (root) mean squared error (MSE and RMSE), r^2 score, explained variance, and the (scaled) mean absolute percentage error (MAPE and sMAPE). We save each optimization trial’s results, configuration, and runtime to a CSV file for debugging purposes. If a prediction model provides feature importances, e.g., XGBoost, we store these for result analyses. *ForeTiS* further supports results analysis by providing a plot function to visualize the predictions and actual values on the test data, see for instance Fig. 3. Additionally, we save the optimized prediction model for reuse on new data.

4. Case study

To demonstrate the broad applicability and flexibility of *ForeTiS*, we show below a case study using three publicly available datasets, namely *MetroInterstateTraffic*, *DailyDelhiClimate*, and *BikeSharing*. Descriptions and statistics of these datasets can be seen in Table 3. All datasets differ in their application domain and the number of samples and features. Furthermore, *MetroInterstateTraffic* has an hourly resolution, while the other three datasets contain daily information. For each dataset, we ran 200 optimization trials. In addition, we varied certain arguments to show the customizability of *ForeTiS*. In our documentation, we provide a step-by-step guide and a video explaining the pipeline configurations and setup: <https://tinyurl.com/TutorialCaseStudies>.

In the following, we outline some specific results from this case study, demonstrating how *ForeTiS* can be used to assess refitting frequencies, featuresets, and prediction models. Fig. 3 shows an example plot generated by *ForeTiS* on the results for *MetroInterstateTraffic*. Similar plots are automatically generated for each prediction

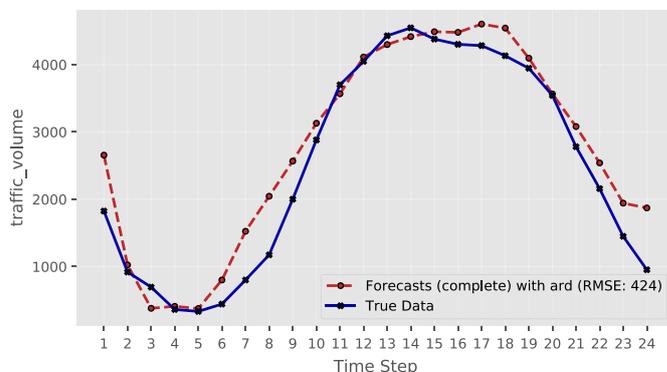


Fig. 3. An example plot of *ForeTiS* showing prediction results and true data for *MetroInterstateTraffic*. On the horizontal axis, we plot the time step within the test set, while we plot the target variable, here the traffic volume, on the vertical axis. The predicted values are plotted in a red dashed line, and the true values are shown in a blue solid line. The legend gives the refitting frequency, the prediction model, and the RMSE obtained.

model, refitting frequency, and featureset included in the optimization pipeline.

The effect of periodic refittings can be seen in Table 4, which compares results using different refitting frequencies for XGBoost on *BikeSharing* and for ARD on *MetroInterstateTraffic*. With respect to *BikeSharing*, which is a rather short dataset, we do not observe a difference in performance when we drop samples for the final model retraining after hyperparameter optimization (refitting frequency ‘complete’ vs. ‘0’). However, regular model refittings are beneficial for this dataset, as XGBoost achieves the best performance when the prediction model is refitted every time step. On the other hand, for the relatively long dataset *MetroInterstateTraffic*, we see a significant drop in performance when we only use the most recent training data for refitting (‘0’). Thus, there is a trade-off between the computational resources required for model retraining and the potential loss of information if we limit the amount of previous data used for refitting. Regarding the refitting frequency, we observe that the behavior differs between the datasets, as less frequent retraining improves the performance for *MetroInterstateTraffic*. *ForeTiS* can indicate this behavior, so the user can choose the most promising refitting frequency for the specific dataset in a production setup.

Table 4

Comparison of different refitting frequencies for XGBoost on *BikeSharing* and ARD on *MetroInterstateTraffic*. The full training set is used for retraining in the refitting frequency ‘complete’, while in the ‘0’ refitting frequency, only the most recent seasons are used for a single retraining at the beginning of the test data.

Dataset	RMSE on refitting frequency				
	complete	0	1	2	3
<i>BikeSharing</i>	905	905	827	831	837
<i>MetroInterstateTraffic</i>	424	1853	1783	1708	1613

Another important functionality of *ForeTiS* is the automated feature engineering. In Table 5, we compare the performance of XGBoost using different featuresets for *BikeSharing*. We observe that XGBoost performs best with the full featureset, with a smaller performance drop when considering only calendrical features compared to weather-related ones. As outlined, *ForeTiS* allows the creation of feature subsets to determine the best-performing one in a comparative study.

Table 5

Comparison of the performance of XGBoost using different featuresets for *BikeSharing*. For a fair comparison, we set the refitting frequency to 0.

	RMSE for featureset	
	Full	Weather
905	1614	1898

Finally, we demonstrate the quick benchmarking of *ForeTiS* for *DailyDelhiClimate*, assuming we want to evaluate the multivariate model GPR against the univariate classical forecasting approach SARIMA. After running both models with a single line of code, we see in Table 6 that GPR outperforms SARIMA on both refitting frequencies we evaluated. Also, the average runtime for each optimization trial is longer for SARIMA. With the aid of such results, a user can decide which algorithm to use for a productive operation, considering the prediction performance and required computational resources.

Table 6

Benchmarking GPR – using the full featureset – and univariate SARIMA on *DailyDelhiClimate*. GPR outperforms SARIMA on both refitting frequencies regarding RMSE and is computationally less exhaustive with respect to the average runtime of one optimization trial.

Prediction model	RMSE refitting frequency		Average runtime [s]
	1	2	
SARIMA	3.67	3.85	211495
GPR	3.43	3.42	251

5. Conclusion

We present *ForeTiS*, a Python framework that covers the entire time series forecasting workflow and allows for performing a comparative study with a single line of code. As outlined, the preprocessing, feature engineering, and optimization pipeline is fully automated yet customizable by the user. We have already implemented various prediction models, ranging from classical forecasting models over machine learning-based approaches to modern deep learning techniques from which users can freely choose. Besides this benefit, especially for non-programmers, *ForeTiS* simplifies developing and benchmarking new prediction models. This design allows for continuous improvements of *ForeTiS* and invites other researchers to collaborate. We further provide a comprehensive online documentation with several hands-on tutorials and videos to guide novice users.

CRedit authorship contribution statement

Josef Eiglsperger: Conceptualization, Methodology, Software, Investigation, Visualization, Writing – original draft. **Florian Haselbeck:** Conceptualization, Methodology, Software, Writing – original draft. **Dominik G. Grimm:** Conceptualization, Methodology, Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code is publicly available at <https://github.com/grimmla/b/ForeTiS>.

Funding

The project is supported by funds of the Federal Ministry of Food and Agriculture (BMEL) based on a decision of the Parliament of the Federal Republic of Germany via the Federal Office for Agriculture and Food (BLE) under the innovation support program [grant number 2818504A18].

References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. <http://dx.doi.org/10.48550/ARXIV.1603.04467>, URL <https://arxiv.org/abs/1603.04467>.
- Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 2623–2631). New York, NY, USA: Association for Computing Machinery, <http://dx.doi.org/10.1145/3292500.3330701>.
- Bauer, A., Züfle, M., Grohmann, J., Schmitt, N., Herbst, N., & Kounev, S. (2020). An Automated Forecasting Framework based on Method Recommendation for Seasonal Time Series. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (pp. 48–55). Edmonton AB Canada: ACM, <http://dx.doi.org/10.1145/3358960.3379123>, URL <https://dl.acm.org/doi/10.1145/3358960.3379123>.
- Bojer, C. S., & Meldgaard, J. P. (2021). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2), 587–603. <http://dx.doi.org/10.48550/arXiv.2009.07701>, URL <https://ideas.repec.org/a/eee/intfor/v37y2021i2p587-603.html>.
- Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In B. Krishnapuram, M. Shah, A. Smola, C. Aggarwal, D. Shen, & R. Rastogi (Eds.), *Proceedings of the 22nd ACM SIGKDD International conference on knowledge discovery and data mining* (pp. 785–794). New York, NY: ACM, <http://dx.doi.org/10.1145/2939672.2939785>.
- Denker, J., & LeCun, Y. (1990). Transforming neural-net output levels to probability distributions. In R. Lippmann, J. Moody, & D. Touretzky (Eds.), *Advances in neural information processing systems*, vol. 3. Morgan-Kaufmann, URL <https://proceedings.neurips.cc/paper/1990/file/7eac532570ff6858afid2723755ff790-Paper.pdf>.
- Haselbeck, F., & Grimm, D. G. (2021). EVARS-GPR: Event-triggered augmented refitting of Gaussian process regression for seasonal data. In S. Edelkamp, R. Möller, & E. Rueckert (Eds.), *KI 2021: Advances in artificial intelligence* (pp. 135–157). Cham: Springer International Publishing.
- Haselbeck, F., Killinger, J., Menrad, K., Hannus, T., & Grimm, D. G. (2022). Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions. *Machine Learning with Applications*, 7, Article 100239. <http://dx.doi.org/10.1016/j.mlwa.2021.100239>, URL <https://linkinghub.elsevier.com/retrieve/pii/S2666827021001201>.
- Heidrich, B., Bartschat, A., Turowski, M., Neumann, O., Phipps, K., Meisenbacher, S., Schmieler, K., Ludwig, N., Mikut, R., & Hagenmeyer, V. (2021). pyWATTS: Python Workflow Automation Tool for Time Series. <http://dx.doi.org/10.48550/arXiv.2106.10157>, URL <http://arxiv.org/abs/2106.10157>.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. <http://dx.doi.org/10.1162/neco.1997.9.8.1735>, arXiv:<https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf>.
- Huber, J., & Stuckenschmidt, H. (2020). Daily retail demand forecasting using machine learning with emphasis on calendric special days. *International Journal of Forecasting*, 36(4), 1420–1438. <http://dx.doi.org/10.1016/j.ijforecast.2020.02.005>, URL <https://EconPapers.repec.org/RePEc:eee:intfor:v:36:y:2020:i:4:p:1420-1438>.
- Hüwel, J. D., Haselbeck, F., Grimm, D. G., & Beecks, C. (2022). Dynamically self-adjusting Gaussian processes for data stream modelling. In R. Bergmann, L. Malburg, S. C. Rodermund, & I. J. Timm (Eds.), *KI 2022: Advances in artificial intelligence* (pp. 96–114). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-031-15791-2_10.
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. <http://dx.doi.org/10.1016/j.ijforecast.2006.03.001>, URL <https://linkinghub.elsevier.com/retrieve/pii/S0169207006000239>.
- Jiao, E. X., & Chen, J. L. (2019). Tourism forecasting: A review of methodological developments over the last decade. *Tourism Economics*, 25(3), 469–492. <http://dx.doi.org/10.1177/1354816618812588>.
- Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., & Király, F. J. (2019). Sktime: A unified interface for machine learning with time series. In *Workshop on systems for ML a NeurIPS 2019*.

- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. <http://dx.doi.org/10.1016/j.ijforecast.2019.04.014>, URL <https://www.sciencedirect.com/science/article/pii/S0169207019301128>.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364. <http://dx.doi.org/10.1016/j.ijforecast.2021.11.013>, URL <https://www.sciencedirect.com/science/article/pii/S0169207021001874>.
- Martínez, F., Frías, M. P., Pérez, M. D., & Rivera, A. J. (2019). A methodology for applying k-nearest neighbor to time series forecasting. *Artificial Intelligence Review*, 52(3), 2019–2037. <http://dx.doi.org/10.1007/s10462-017-9593-z>, URL <http://link.springer.com/10.1007/s10462-017-9593-z>.
- Meisenbacher, S., Turowski, M., Phipps, K., Rätz, M., Müller, D., Hagenmeyer, V., & Mikut, R. (2022). Review of automated time series forecasting pipelines. *WIREs Data Mining and Knowledge Discovery*, 12(6), <http://dx.doi.org/10.1002/widm.1475>, URL <https://onlinelibrary.wiley.com/doi/10.1002/widm.1475>.
- Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239).
- Moez, A. (2023). PyCaret: An open source, low-code machine learning library in Python. URL <https://www.pycaret.org> PyCaret version 3.0.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*, vol. 32 (pp. 8024–8035). Curran Associates, Inc., URL <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with python. In *9th Python in science conference*.
- Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International conference on neural information processing systems - volume 2* (pp. 2951–2959). Red Hook, NY, USA: Curran Associates Inc..
- Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., & Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In H. J. Escalante, & K. Hofmann (Eds.), *Proceedings of Machine Learning Research: vol. 133, Proceedings of the NeurIPS 2020 Competition and demonstration track* (pp. 3–26). PMLR, URL <https://proceedings.mlr.press/v133/turner21a.html>.
- Williams, C., & Rasmussen, C. (1995). Gaussian processes for regression. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural information processing systems*, vol. 8. MIT Press, URL <https://proceedings.neurips.cc/paper/1995/file/7cce53cf90577442771720a370c3c723-Paper.pdf>.
- Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3), 324–342. <http://dx.doi.org/10.1287/mnsc.6.3.324>.
- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B.*, 67, 301–320. <http://dx.doi.org/10.1111/j.1467-9868.2005.00503.x>.
- Züfle, M., & Kounev, S. (2020). A Framework for Time Series Preprocessing and History-based Forecasting Method Recommendation. (pp. 141–144). <http://dx.doi.org/10.15439/2020F101>, URL <https://annals-csis.org/proceedings/2020/drp/101.html>.

List of Symbols

General Symbols

p	probability distribution
\mathcal{D}	dataset containing features and corresponding target values
$n \in \mathbb{N}$	number of samples, i.e., pairs of feature vectors and corresponding target values, in the dataset \mathcal{D}
$t \in \mathbb{N}$	time step
$m \in \mathbb{N}$	number of features
$\mathbf{x}_t \in \mathbb{R}^m$	feature vector at time step t
$\mathbf{X} \in \mathbb{R}^{n \times m}$	feature matrix containing feature vectors \mathbf{x}_t of the whole dataset, also referred to as external factors
$y_t \in \mathbb{R}$	target value at time step t
$\mathbf{y} \in \mathbb{R}^n$	target variable time series containing all target values y_t of the whole dataset in chronological order
$\hat{y}_t \in \mathbb{R}$	predicted target value at time step t
$h \in \mathbb{N}$	forecast horizon

$n_{seas} \in \mathbb{N}$	seasonal length
$n_w \in \mathbb{N}$	window size
$\varepsilon \in \mathbb{R}^{dim}$	unobserved error term (dimension $dim \in \mathbb{N}$ depending on context)
$I \in \mathbb{R}^{dim \times dim}$	identity matrix (dimension $dim \in \mathbb{N}$ depending on context)
$j \in \mathbb{N}$	number of folds for cross-validation

Classical Time Series Forecasting Symbols

$l_t \in \mathbb{R}$	level component at time step t of Exponential Smoothing
$v_t \in \mathbb{R}$	trend component at time step t of Exponential Smoothing
$c_t \in \mathbb{R}$	seasonal component at time step t of Exponential Smoothing
$\alpha \in [0, 1]$	smoothing parameter for the level component of Exponential Smoothing
$\beta \in [0, 1]$	smoothing parameter for the trend component of Exponential Smoothing
$\rho \in]0, 1]$	damping parameter for damped trend of Exponential Smoothing
$\gamma \in [0, 1]$	smoothing parameter for the seasonal component of Exponential Smoothing
$\lfloor x \rfloor$	floor function giving the largest integer value less than or equal to x
$d \in \mathbb{N}$	differencing degree of ARIMA and related prediction models
$\mathbf{y}^d \in \mathbb{R}^n$	d times differenced time series
$p_{AR} \in \mathbb{N}$	order of the autoregressive term of ARIMA and related prediction models

$q_{MA} \in \mathbb{N}$	order of the moving average term of ARIMA and related prediction models
$\varepsilon_{wn,t} \in \mathbb{R}$	white noise error term at time step t of ARIMA and related prediction models
$\mu \in \mathbb{R}$	intercept of ARIMA and related prediction models
$\lambda_i \in \mathbb{R}$	model parameters for the autoregressive part of ARIMA and related prediction models
$\chi_i \in \mathbb{R}$	model parameters for the moving average part of ARIMA and related prediction models
$P_{AR} \in \mathbb{N}$	order of the seasonal autoregressive term of SARIMA and SARIMAX
$D \in \mathbb{N}$	seasonal differencing degree of SARIMA and SARIMAX
$Q_{MA} \in \mathbb{N}$	order of the seasonal moving average term of SARIMA and SARIMAX

Machine Learning Symbols

$p(\mathcal{D} \mid \boldsymbol{w})$	likelihood term
$p(\boldsymbol{w})$	<i>prior</i> for model parameters \boldsymbol{w}
$p(\boldsymbol{w} \mid \mathcal{D})$	<i>posterior</i> probability
$p(\mathcal{D})$	probability distribution of observed data \mathcal{D}
$\boldsymbol{w} \in \mathbb{R}^{dim}$	model parameters (dimension $dim \in \mathbb{N}$ depending on context)
$\boldsymbol{w}^* \in \mathbb{R}^{dim}$	point estimates of the model parameters that explain the observed data best (dimension $dim \in \mathbb{N}$ depending on context)
$\boldsymbol{X}^\dagger \in \mathbb{R}^{n \times m+1}$	matrix containing a vector of ones and the feature matrix \boldsymbol{X}

$w_k \in \mathbb{R}$	model coefficient of linear regression
$w_0 \in \mathbb{R}$	intercept and bias term, respectively
$\Omega(\boldsymbol{w}) \in \mathbb{R}$	penalty term for regularization
$\kappa \in \mathbb{R}_{>0}$	coefficient of the penalty term for regularization
$\ \boldsymbol{x}\ _1$	L1-norm of a vector \boldsymbol{x}
$\ \boldsymbol{x}\ _2^2$	L2-norm of a vector \boldsymbol{x}
$\zeta \in [0, 1]$	weighting factor between L1- and L2-norm of Elastic Net
$\boldsymbol{o}_i \in \mathbb{R}^{n_{i,o}}$	output vector of a neural network at layer i
$n_{i,o} \in \mathbb{N}$	number of neurons in layer i of a neural network
$n_{i,in} \in \mathbb{N}$	number of inputs at layer i of a neural network
$\boldsymbol{a}: \mathbb{R}^{n_{i,o}} \rightarrow \mathbb{R}^{n_{i,o}}$	activation function of a neural network
$\boldsymbol{W}_i \in \mathbb{R}^{n_{i,in} \times n_{i,o}}$	weight matrix of neural network at layer i
$\boldsymbol{x}_{i,in} \in \mathbb{R}^{n_{i,in}}$	input vector of neural network at layer i
$\boldsymbol{b}_i \in \mathbb{R}^{n_{i,o}}$	bias vector of neural network at layer i
$\sigma^2 \in \mathbb{R}_{>0}$	variance for a Gaussian distribution
$\nu \in \mathbb{R}_{>0}$	precision for a Gaussian distribution
$\boldsymbol{Y} \in \mathbb{R}^{dim \times dim}$	precision matrix for ARD (dimension $dim \in \mathbb{N}$ depending on size of weight vector \boldsymbol{w})
$\boldsymbol{C} \in \mathbb{R}^{dim \times dim}$	covariance matrix for <i>prior</i> over model parameters in mathematical description of GPR, $dim = m$ before mapping to feature space and $dim = f$ afterwards

$\mathbf{A}^{-1} \in \mathbb{R}^{dim \times dim}$	covariance matrix for <i>posterior</i> in mathematical description of GPR, $dim = m$ before mapping to feature space and $dim = f$ afterwards
$\bar{\mathbf{w}} \in \mathbb{R}^m$	mean for <i>posterior</i> in mathematical description of GPR before mapping to feature space
$\mathbf{x}_{test} \in \mathbb{R}^m$	test input vector in mathematical description of GPR
$y_{test} \in \mathbb{R}$	test output variable in mathematical description of GPR
$\phi: \mathbb{R}^m \rightarrow \mathbb{R}^f$	function mapping from m -dimensional space into f -dimensional feature space in mathematical description of GPR
$\Phi \in \mathbb{R}^{f \times n}$	feature matrix mapped to feature space in mathematical description of GPR
$\phi(\mathbf{x}_{test}) \in \mathbb{R}^f$	test input vector mapped to feature space in mathematical description of GPR
$\mathbf{K} \in \mathbb{R}^{dim1 \times dim2}$	covariance matrix based on pairwise evaluation of input samples, with $dim1 \in \mathbb{N}$ and $dim2 \in \mathbb{N}$ depending on the size of the input samples in mathematical description of GPR
$k(\mathbf{x}, \mathbf{x}')$	kernel with vector \mathbf{x} in training set and vector \mathbf{x}' in test set in mathematical description of GPR
$\psi(\mathbf{x})$	mapping function for kernel trick in mathematical description of GPR
$n^* \in \mathbb{N}$	number of test samples in mathematical description of GPR
$\mathbf{X}^* \in \mathbb{R}^{n^* \times m}$	matrix of test samples in mathematical description of GPR
$\mathbf{y}^* \in \mathbb{R}^{n^*}$	target vector of the test samples in mathematical description of GPR
$\bar{\mathbf{y}}^* \in \mathbb{R}^{n^*}$	mean of predictive distribution of GPR
$\mathbf{COV}(\mathbf{y}^*) \in \mathbb{R}^{n^* \times n^*}$	covariance of predictive distribution of GPR
f_{GP}	a GP can be seen as a distribution over functions f_{GP}

$\mathcal{GP}(\mathbf{mf}, k)$	GP defined by mean function \mathbf{mf} and kernel k
\mathbf{mf}	mean function of a GP
k_{Lin}	linear kernel
$z \in \mathbb{R}$	offset of a linear kernel
k_{SE}	squared exponential kernel
$\iota \in \mathbb{R}_{>0}$	lengthscale of a kernel
k_{RQ}	rational quadratic kernel
$\tau \in \mathbb{R}_{>0}$	scale mixture parameter of rational quadratic kernel
k_{Mat}	Matérn kernel
$\nu \in \mathbb{R}_{>0}$	parameter controlling the smoothness of the Matérn kernel
K_ν	modified Bessel function of the Matérn kernel
\mathcal{L}_{lml}	log marginal likelihood
\mathcal{K}	set of kernel expression
\mathcal{B}	set of base kernels
k_{base}	base kernel
k_c	composed kernel expression
k_{new}	new kernel expression
$i_{CKS} \in \mathbb{N}$	maximum number of iterations for CKS
$i_{AKS} \in \mathbb{N}$	maximum number of iterations for AKS
k_0	starting kernel for AKS

Hyperparameter Optimization Symbols

$\theta \in \mathbb{R}^{n_h}$	hyperparameter set
$n_h \in \mathbb{N}$	number of hyperparameters
$n_t \in \mathbb{N}$	number of hyperparameter sets
$\Theta \in \mathbb{R}^{n_h \times n_t}$	hyperparameter space
$\mathcal{L}_v \in \mathbb{R}$	validation loss
$\theta^* \in \mathbb{R}^{n_h}$	hyperparameter set yielding the best result
\mathcal{H}	historical observations during hyperparameter optimization process
$n_{finished} \in \mathbb{N}$	number of already tested hyperparameter combinations before starting hyperparameter optimization process
\mathcal{M}_i	surrogate model
\mathcal{M}_0	initial surrogate model
$ov \in \mathbb{R}$	objective value for hyperparameter optimization
$of: \mathbb{R}^{n_h} \rightarrow \mathbb{R}$	objective function
$\theta^+ \in \mathbb{R}^{n_h}$	hyperparameter set currently leading to best result during hyperparameter optimization
$i: \mathbb{R}^{n_h} \rightarrow \mathbb{R}_{\geq 0}$	improvement function
$EI(\theta)$	expected improvement

Change Point Detection Symbols

p_t^{CF}	probability density at time step t of ChangeFinder
------------	--

$\mathbf{y}^{CF} \in \mathbb{R}^t$	vector of smoothed outlier scores of ChangeFinder
$\mathbf{z}^{CF} \in \mathbb{R}^t$	vector of change point scores of ChangeFinder
$r_t \in \mathbb{N}$	run length at time step t of BOCPD
$\mathbf{y}_t^r \in \mathbb{R}^{r_t}$	observations associated with run length r_t of BOCPD
$H(t')$	hazard function of BOCPD
p_{gap}	probability distribution of interval between change points of BOCPD
$p_s \in]0, 1]$	success probability of a geometric distribution

Symbols in Chapter Results

M_{base}	initial prediction model of EVARS-GPR and EVARS-GPR+
$M_{current}$	current prediction model of EVARS-GPR and EVARS-GPR+
\mathcal{D}_{off}	dataset containing samples collected offline, i.e., before the online phase, of EVARS-GPR and EVARS-GPR+
$t_{off} \in \mathbb{N}$	last time step of the offline phase of EVARS-GPR and EVARS-GPR+
\mathcal{D}'	augmented dataset of EVARS-GPR and EVARS-GPR+
$cp \in \{False, True\}$	boolean showing whether a change point was detected in EVARS-GPR and EVARS-GPR+
$\eta \in \mathbb{R}$	output scaling factor of EVARS-GPR and EVARS-GPR+
$n_\eta \in \mathbb{N}$	number of previous seasons to consider for calculation of output scale change of EVARS-GPR and EVARS-GPR+
$\pi_\eta \in \mathbb{R}_{\geq 0}$	threshold for output scale change of EVARS-GPR and EVARS-GPR+ to declare an output scale-related change point

θ_{CF}	parameters of ChangeFinder included in EVARS-GPR and EVARS-GPR+
$\eta_{old} \in \mathbb{R}$	output scaling factor of last augmented refitting of EVARS-GPR and EVARS-GPR+
$t_{last} \in \mathbb{N}$	time step of last augmented refitting of EVARS-GPR and EVARS-GPR+
$s \in \mathbb{R}_{\geq 0}$	change point score of ETKA
$\delta \in \mathbb{R}$	tolerance factor for change point score calculation of ETKA
$\epsilon_{ETKA} \in \mathbb{R}_{> 0}$	threshold for change point score s to declare a change point in ETKA

List of Acronyms

- ABCD** Automatic Bayesian Covariance Discovery. 11, 68
- AKS** Adjusting Kernel Search. 17, 32–34, 54, 55, 67–70, 75, 154
- AR** autoregressive. 22, 23
- ARD** Automatic Relevance Determination. 26, 152
- ARIMA** Autoregressive Integrated Moving Average. 4, 8, 21–23, 150, 151
- ARIMAX** Autoregressive Integrated Moving Average with eXogenous factors. 9
- BOCPD** Bayesian Online Change Point Detection. 11, 40, 50, 156
- CKS** Compositional Kernel Search. 11, 32–34, 54, 67, 154
- CPD** change point detection. 2, 12, 17, 21, 39, 40, 49–51, 54, 55, 57, 64–69, 73, 75
- CUSUM** cumulative sum. 40, 54, 67
- DL** deep learning. 20, 57, 62, 70
- ETKA** Event-Triggered Kernel Adjustments in Gaussian Process modeling. 54, 55, 67–69, 75, 76
- EVARS-GPR** EVent-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data. 48–53, 55, 57, 64–70, 75, 164, 165
- GAN** Generative Adversarial Network. 73

- GP** Gaussian Process. 11, 12, 29–31, 37, 54, 55, 153, 154, 163
- GP-NBC** GP-non-Bayesian clustering. 12, 69
- GPR** Gaussian Process Regression. 2, 11, 12, 15, 17, 21, 26, 27, 30, 39, 44, 46, 48, 53, 54, 57, 60, 65, 69, 75, 152, 153
- INTEL** INstant TEmporal structure Learning. 12, 69
- k-NN** k-Nearest Neighbor. 4, 9, 46
- LASSO** Least Absolute Shrinkage and Selection Operator. 24
- LSTM** Long Short-Term Memory. 9, 23, 25, 26, 34, 44, 46, 60, 73
- MA** moving average. 22, 23
- MAE** mean absolute error. 38, 55
- MAP** maximum a posterior. 26
- MAPE** mean absolute percentage error. 38, 39, 46, 60
- ML** machine learning. 2, 4, 5, 7–10, 12–14, 16, 19, 21, 23, 25, 27, 34, 35, 44, 46, 47, 57, 59–61, 72, 73
- MLP** Multilayer Perceptron. 9, 23–26, 44, 73
- MSE** mean squared error. 38
- PCA** Principal Component Analysis. 5, 46, 56, 71
- RMSE** root mean squared error. 38, 46, 50, 52, 53, 64–66, 165
- SARIMA** Seasonal Autoregressive Integrated Moving Average. 8, 9, 23, 44, 47, 60, 151
- SARIMAX** Seasonal Autoregressive Integrated Moving Average with eXogenous factors. 8, 23, 44, 47, 151
- SDAR** sequentially discounting autoregressive. 40
- sMAPE** symmetric mean absolute percentage error. 38, 39, 46, 60

SVR Support Vector Regression. 9

VAE Variational Auto-Encoder. 73

List of Figures

1.1	Examples of time series patterns: All examples include additive noise. (a) Seasonal behavior. (b) Negative trend pattern without seasonality. (c) Seasonal pattern combined with positive linear trend. (d) Multi-seasonal behavior.	3
1.2	A common procedure in a time series forecasting project: After collecting the raw target time series, the data must be cleaned and missing values may need to be imputed. Then, in the case of a multivariate prediction model, feature engineering must be performed. Using this dataset, model selection, including hyperparameter optimization and model training, can be performed to finally select a model with the best hyperparameter configuration based on the results on validation data, see Section 2.4 for more details. Finally, we can evaluate the generalization ability of the model on unseen data.	4
2.1	Examples of kernels k: For each kernel, we visualize the kernel $k(\cdot, 1)$ in the upper part of the plot as well as samples drawn from a GP with the respective kernel. The x-axes have the same range for all plots. (a) Examples of base kernels. (b) Examples of combined kernels created by adding or multiplying base kernels.	31

- 2.2 **Time series cross-validation, Grid and Random Search:** (a) Visualization of a time series cross-validation with a separate test set shown for three folds used for training and validation. (b) and (c): Visualization of Grid and Random Search, see Section 2.4.2. We show nine trials for optimizing two parameters, of which one is important, i.e., influencing the prediction result significantly. With Grid Search, only three distinct places of the important parameter are tested, whereas Random Search tests a different place in each trial. Figures (b) and (c) are similar to (Bergstra & Bengio, 2012). 35
- 3.1 **Datasets, featuresets, and pre-processing methods for our study:** This figure contains parts of Figures 1 and 2 of the original publication and is shown for better readability (Haselbeck et al., 2022). (a) Overview of the five horticultural retail sales datasets based on two data sources. (b) Composition of the four featuresets that we used in addition to the univariate case as well as the missing value imputation and dimensionality reduction methods we considered for each featureset. 45
- 3.2 **Overview of EVARS-GPR:** This figure is similar to Figure 1 of the original publication and is shown for better readability (Haselbeck & Grimm, 2021). During the offline phase, the initial prediction model is determined. With every new sample available during the online phase, the next target value is first predicted before an online change point detection algorithm runs. If a change point is detected, the algorithm checks whether the extent of the output scale change compared to previous seasons exceeds a threshold. In case this condition is also fulfilled, the current prediction model is refitted using augmented existing data. EVARS-GPR continues with the current prediction model if one of these conditions is not met. 49
- 3.3 **Overview of the main components of ForeTiS:** This figure is similar to Figure 1 of the original publication and is shown for better readability (Eiglsperger et al., 2023). The submodule `preparation` contains the data pre-processing and feature engineering methods. We designed `model` to allow for easy integration of further prediction models and already included several ones. For hyperparameter search, we leverage Bayesian optimization via the Python package `Optuna`. Our testing module enables to test different refitting schemes, of which the results can be analyzed using `evaluation`. 57

List of Tables

3.1 **Overview of results in terms of RMSE on real-world data:** The unpublished extension EVARS-GPR+ is compared to EVARS-GPR, M_{base} without any refitting during the online phase, and the computationally exhaustive approaches PR1, PR2, and MWGPR. PR1 and PR2 refit the prediction model in every and every second time step, respectively. Moving-Window GPR (MWGPR) performs a model refitting every time step, discarding the oldest sample and a recursive bias term for model correction (Ni et al., 2012). We show results for the real-world datasets used in the EVARS-GPR publication, half of which does not contain an output scale change. The best result for each dataset is highlighted in bold. For EVARS-GPR+, we show the percentage change in RMSE compared to EVARS-GPR. Dataset sources: (Haselbeck et al., 2022): *CashierData*; (Hyndman & Athanasopoulos, 2021): *DrugSales*, *VisitorNights*, and *Beer*; (Box et al., 2016): *AirPassengers*; (Earth System Research Laboratory, 2021): *MaunaLoa*; (Makridakis & Wheelwright, 1989): *ChampagneSales*; (Chakrabarty, 2021): *TouristsIndia*; (Makridakis et al., 1998): *Milk*, and *USDeaths*. 52

Bibliography

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., . . . Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv preprint*. <https://doi.org/10.48550/ARXIV.1603.04467>
- Adams, R. P. & MacKay, D. J. C. (2007). Bayesian Online Changepoint Detection. *arXiv preprint*. <https://doi.org/10.48550/arXiv.0710.3742>
- Ahmad, T., Zhang, H. & Yan, B. (2020). A review on renewable energy and electricity requirement forecasting models for smart grid and buildings. *Sustainable Cities and Society*, 55, 102052. <https://doi.org/10.1016/j.scs.2020.102052>
- Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. (2019). Optuna: A Next-generation Hyperparameter Optimization Framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2623–2631. <https://doi.org/10.1145/3292500.3330701>
- Alsharef, A., Aggarwal, K., Sonia, Kumar, M. & Mishra, A. (2022). Review of ML and AutoML Solutions to Forecast Time-Series Data. *Archives of Computational Methods in Engineering*, 29(7), 5297–5311. <https://doi.org/10.1007/s11831-022-09765-0>
- Aminikhanghahi, S. & Cook, D. J. (2017). A Survey of Methods for Time Series Change Point Detection. *Knowledge and information systems*, 51, 339–367. <https://doi.org/10.1007/s10115-016-0987-z>
- Arunraj, N. S. & Ahrens, D. (2015). A hybrid seasonal autoregressive integrated moving average and quantile regression for daily food sales forecasting. *International Journal of Production Economics*, 170, 321–335. <https://doi.org/10.1016/j.ijpe.2015.09.039>
- Arunraj, N. S., Ahrens, D. & Fernandes, M. (2016). Application of SARIMAX Model to Forecast Daily Sales in Food Retail Industry. *International Journal of Operations*

- Research and Information Systems*, 7(2), 1–21. <https://doi.org/10.4018/IJORIS.2016040101>
- Athanasopoulos, G., Hyndman, R. J., Song, H. & Wu, D. C. (2011). The tourism forecasting competition. *International Journal of Forecasting*, 27(3), 822–844. <https://doi.org/10.1016/j.ijforecast.2010.04.009>
- Bauer, A., Züfle, M., Grohmann, J., Schmitt, N., Herbst, N. & Kounev, S. (2020). An Automated Forecasting Framework based on Method Recommendation for Seasonal Time Series. *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, 48–55. <https://doi.org/10.1145/3358960.3379123>
- Benidis, K., Rangapuram, S. S., Flunkert, V., Wang, Y., Maddix, D., Turkmen, C., Gasthaus, J., Bohlke-Schneider, M., Salinas, D., Stella, L., Aubet, F.-X., Callot, L. & Januschowski, T. (2022). Deep Learning for Time Series Forecasting: Tutorial and Literature Survey. *ACM Computing Surveys*, 55(6), 1–36. <https://doi.org/10.1145/3533382>
- Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. (2011). Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, 24, 2546–2554.
- Bergstra, J. & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(10), 281–305.
- Bhatnagar, A., Kassianik, P., Liu, C., Lan, T., Yang, W., Cassius, R., Sahoo, D., Arpit, D., Subramanian, S., Woo, G., Saha, A., Jagota, A. K., Gopalakrishnan, G., Singh, M., Krithika, K. C., Maddineni, S., Cho, D., Zong, B., Zhou, Y., ... Wang, H. (2021). Merlion: A Machine Learning Library for Time Series. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2109.09265>
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
- Blundell, C., Cornebise, J., Kavukcuoglu, K. & Wierstra, D. (2015). Weight Uncertainty in Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 37, 1613–1622.
- Bojer, C. S. & Meldgaard, J. P. (2021). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2), 587–603. <https://doi.org/10.1016/j.ijforecast.2020.07.007>
- Böse, J.-H., Flunkert, V., Gasthaus, J., Januschowski, T., Lange, D., Salinas, D., Schelter, S., Seeger, M. & Wang, Y. (2017). Probabilistic Demand Forecasting at Scale. *Proceedings of the VLDB Endowment*, 10(12), 1694–1705. <https://doi.org/10.14778/3137765.3137775>

- Box, G. E. P. & Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. Holden-Day.
- Box, G. E. P., Jenkins, G. M., Reinsel, G. C. & Ljung, G. M. (2016). *Time series analysis: Forecasting and control* (5th ed.). Wiley.
- Brockwell, P. J. & Davis, R. A. (2016). *Introduction to Time Series and Forecasting* (3rd ed.). Springer. <https://doi.org/10.1007/978-3-319-29854-2>
- Brown, R. G. (1956). Exponential Smoothing for Predicting Demand. *Arthur D. Little Inc.*
- Bundesministerium für Ernährung und Landwirtschaft. (2021). Der Gartenbau in Deutschland: Auswertung des Gartenbaumoduls der Agrarstrukturerhebung 2016. <https://www.bmel.de/SharedDocs/Downloads/DE/Broschueren/Gartenbauerhebung.html>
- Chakrabarty, N. (2021). Quarterly Foreign Tourist Arrivals in India: Determinants of Foreign Tourism Demand and Foreign Tourist Arrivals 2005-2016. Retrieved May 14, 2021, from <https://www.kaggle.com/navoneel/fta-data>
- Chen, T. & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Deb, C., Zhang, F., Yang, J., Lee, S. E. & Shah, K. W. (2017). A review on time series forecasting techniques for building energy consumption. *Renewable and Sustainable Energy Reviews*, 74, 902–924. <https://doi.org/10.1016/j.rser.2017.02.085>
- Ditzler, G., Roveri, M., Alippi, C. & Polikar, R. (2015). Learning in Nonstationary Environments: A Survey. *IEEE Computational Intelligence Magazine*, 10(4), 12–25. <https://doi.org/10.1109/MCI.2015.2471196>
- Doganis, P., Alexandridis, A., Patrinos, P. & Sarimveis, H. (2006). Time series sales forecasting for short shelf-life food products based on artificial neural networks and evolutionary computing. *Journal of Food Engineering*, 75(2), 196–204. <https://doi.org/10.1016/j.jfoodeng.2005.03.056>
- Duan, Y., Li, G., Tien, J. M. & Huo, J. (2012). Inventory models for perishable items with inventory level dependent demand rate. *Applied Mathematical Modelling*, 36(10), 5015–5028. <https://doi.org/10.1016/j.apm.2011.12.039>
- Duvenaud, D. (2014). Automatic model construction with Gaussian processes (Doctoral thesis). University of Cambridge. <https://doi.org/10.17863/CAM.14087>
- Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B. & Ghahramani, Z. (2013). Structure Discovery in Nonparametric Regression through Compositional Kernel

- Search. *Proceedings of the 30th International Conference on Machine Learning*, 1166–1174.
- Earth System Research Laboratory. (2021). CO2 PPM - Trends in Atmospheric Carbon Dioxide: Atmospheric Carbon Dioxide Dry Air Mole Fractions at Mauna Loa, Hawaii. Retrieved May 14, 2021, from <https://datahub.io/core/co2-ppm-daily#data>
- Eiglsperger, J. (2022). Synthesizing Time-Series Data with Generative Adversarial Networks (Master's thesis). Technical University of Munich, Campus Straubing for Biotechnology and Sustainability.
- Eiglsperger, J., Haselbeck, F. & Grimm, D. G. (2023). ForeTiS: A comprehensive time series forecasting framework in Python. *Machine Learning with Applications*, 12, 100467. <https://doi.org/10.1016/j.mlwa.2023.100467>
- Erhardt, W., Götz, E., Bödeker, N. & Seybold, S. (Eds.). (2014). Zander Handwörterbuch der Pflanzennamen (19th ed.). Ulmer.
- Erickson, N., Mueller, J., Shirkov, A., Zhang, H., Larroy, P., Li, M. & Smola, A. (2020). AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2003.06505>
- Feurer, M., Klein, A., Eggenberger, K., Springenberg, J., Blum, M. & Hutter, F. (2015). Efficient and Robust Automated Machine Learning. *Advances in Neural Information Processing Systems*, 28, 2962–2970.
- Fildes, R., Ma, S. & Kolassa, S. (2022). Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4), 1283–1318. <https://doi.org/10.1016/j.ijforecast.2019.06.004>
- Flores, B. E. (1986). A pragmatic view of accuracy measurement in forecasting. *Omega*, 14(2), 93–98. [https://doi.org/10.1016/0305-0483\(86\)90013-7](https://doi.org/10.1016/0305-0483(86)90013-7)
- Gama, J. (2012). A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, 1, 45–55. <https://doi.org/10.1007/s13748-011-0002-6>
- Gardner, E. S. (2006). Exponential smoothing: The state of the art - Part II. *International Journal of Forecasting*, 22(4), 637–666. <https://doi.org/10.1016/j.ijforecast.2006.03.005>
- Garnett, R. (2023). Bayesian Optimization. Cambridge University Press. <https://doi.org/10.1017/9781108348973>
- Garnett, R., Osborne, M. A. & Roberts, S. J. (2009). Sequential Bayesian Prediction in the Presence of Changepoints. *Proceedings of the 26th International Conference on Machine Learning*, 345–352. <https://doi.org/10.1145/1553374.1553418>

- Golubovic, K. (2022). An Online Machine Learning Framework for Changing Distributions in Time Series Data (Master's thesis). Technical University of Munich, Faculty of Informatics.
- Goodfellow, I., Bengio, Y. & Courville, A. (2016). Deep Learning. MIT Press.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative Adversarial Nets. *Advances in Neural Information Processing Systems*, 27, 2672–2680.
- Grande, R. C., Walsh, T. J., Chowdhary, G., Ferguson, S. & How, J. P. (2017). Online Regression for Data With Changepoints Using Gaussian Processes and Reusable Models. *IEEE transactions on neural networks and learning systems*, 28(9), 2115–2128. <https://doi.org/10.1109/TNNLS.2016.2574565>
- Guyon, I. & Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research*, 3, 1157–1182.
- Haselbeck, F. & Grimm, D. G. (2021). EVARS-GPR: EVent-Triggered Augmented Refitting of Gaussian Process Regression for Seasonal Data. *KI 2021: Advances in Artificial Intelligence*, 12873, 135–157. https://doi.org/10.1007/978-3-030-87626-5_11
- Haselbeck, F., Killinger, J., Menrad, K., Hannus, T. & Grimm, D. G. (2022). Machine Learning Outperforms Classical Forecasting on Horticultural Sales Predictions. *Machine Learning with Applications*, 7, 100239. <https://doi.org/10.1016/j.mlwa.2021.100239>
- Heidrich, B., Bartschat, A., Turowski, M., Neumann, O., Phipps, K., Meisenbacher, S., Schmieder, K., Ludwig, N., Mikut, R. & Hagenmeyer, V. (2021). pyWATTS: Python Workflow Automation Tool for Time Series. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2106.10157>
- Herzen, J., Lässig, F., Piazzetta, S. G., Neuer, T., Tafti, L., Raille, G., Pottelbergh, T. V., Pasięka, M., Skrodzki, A., Huguenin, N., Dumonal, M., Kosciusz, J., Bader, D., Gusset, F., Benheddi, M., Williamson, C., Kosinski, M., Petrik, M. & Grosch, G. (2022). Darts: User-Friendly Modern Machine Learning for Time Series. *Journal of Machine Learning Research*, 23(124), 1–6.
- Hewamalage, H., Bergmeir, C. & Bandara, K. (2021). Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1), 388–427. <https://doi.org/10.1016/j.ijforecast.2020.06.008>
- Hobensack, M., Song, J., Scharp, D., Bowles, K. H. & Topaz, M. (2023). Machine learning applied to electronic health record data in home healthcare: A scoping review.

- International Journal of Medical Informatics*, 170, 104978. <https://doi.org/10.1016/j.ijmedinf.2022.104978>
- Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Hoerl, A. E. & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>
- Holt, C. C. (1957). Forecasting seasonals and trends by exponentially weighted moving averages. *Office of Naval Research Memorandum*.
- Hong, T., Xie, J. & Black, J. (2019). Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. *International Journal of Forecasting*, 35(4), 1389–1399. <https://doi.org/10.1016/j.ijforecast.2019.02.006>
- Huber, J. (2019). Data-driven decision support for perishable goods (Doctoral thesis). University of Mannheim.
- Huber, J., Gossmann, A. & Stuckenschmidt, H. (2017). Cluster-based hierarchical demand forecasting for perishable goods. *Expert Systems with Applications*, 76, 140–151. <https://doi.org/10.1016/j.eswa.2017.01.022>
- Huber, J. & Stuckenschmidt, H. (2020). Daily retail demand forecasting using machine learning with emphasis on calendric special days. *International Journal of Forecasting*, 36(4), 1420–1438. <https://doi.org/10.1016/j.ijforecast.2020.02.005>
- Huber, J. & Stuckenschmidt, H. (2021). Intraday shelf replenishment decision support for perishable goods. *International Journal of Production Economics*, 231, 107828. <https://doi.org/10.1016/j.ijpe.2020.107828>
- Hüwel, J. D., Berns, F. & Beecks, C. (2021). Automated Kernel Search for Gaussian Processes on Data Streams. *2021 IEEE International Conference on Big Data (Big Data)*, 3584–3588. <https://doi.org/10.1109/BigData52589.2021.9671767>
- Hüwel, J. D., Haselbeck, F., Grimm, D. G. & Beecks, C. (2022). Dynamically Self-adjusting Gaussian Processes for Data Stream Modelling. *KI 2022: Advances in Artificial Intelligence*, 13404, 96–114. https://doi.org/10.1007/978-3-031-15791-2_10
- Hyndman, R. J. & Athanasopoulos, G. (2021). Forecasting: principles and practice (3rd ed.). OTexts.
- Hyndman, R. J. & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. <https://doi.org/10.1016/j.ijforecast.2006.03.001>

- Ibrahim, M. S., Dong, W. & Yang, Q. (2020). Machine learning driven smart electric power systems: Current trends and new perspectives. *Applied Energy*, 272, 115237. <https://doi.org/10.1016/j.apenergy.2020.115237>
- Ingle, C., Bakliwal, D., Jain, J., Singh, P., Kale, P. & Chhajed, V. (2021). Demand Forecasting: Literature Review On Various Methodologies. *12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–7. <https://doi.org/10.1109/ICCCNT51525.2021.9580139>
- Ivanov, D., Tsipoulanidis, A. & Schönberger, J. (2019). Demand Forecasting. *Global supply chain and operations management (2nd ed.)*, 319–333. https://doi.org/10.1007/978-3-319-94313-8_11
- Iwata, T., Nakamura, K., Tokusashi, Y. & Matsutani, H. (2019). Accelerating Online Change-Point Detection Algorithm Using 10 GbE FPGA NIC. *Euro-Par 2018: Parallel Processing Workshops*, 11339, 506–517. https://doi.org/10.1007/978-3-030-10549-5_40
- James, G., Witten, D., Hastie, T. & Tibshirani, R. (2017). An introduction to statistical learning: With applications in R (8th ed.). Springer. <https://doi.org/10.1007/978-1-4614-7138-7>
- Jiao, E. X. & Chen, J. L. (2019). Tourism forecasting: A review of methodological developments over the last decade. *Tourism Economics*, 25(3), 469–492. <https://doi.org/10.1177/1354816618812588>
- Jin, H., Chollet, F., Song, Q. & Hu, X. (2023). AutoKeras: An AutoML Library for Deep Learning. *Journal of Machine Learning Research*, 24(6), 1–6.
- Jin, H., Chen, X., Wang, L., Yang, K. & Wu, L. (2015). Adaptive Soft Sensor Development Based on Online Ensemble Gaussian Process Regression for Nonlinear Time-Varying Batch Processes. *Industrial & Engineering Chemistry Research*, 54(30), 7320–7345. <https://doi.org/10.1021/acs.iecr.5b01495>
- Jospin, L. V., Laga, H., Boussaid, F., Buntine, W. & Bennamoun, M. (2022). Hands-On Bayesian Neural Networks - A Tutorial for Deep Learning Users. *IEEE Computational Intelligence Magazine*, 17(2), 29–48. <https://doi.org/10.1109/mci.2022.3155327>
- Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint*. <https://doi.org/10.48550/arXiv.1412.6980>
- Kingma, D. P. & Welling, M. (2014). Auto-Encoding Variational Bayes. *2nd International Conference on Learning Representations, ICLR 2014, Conference Track Proceedings*.

- Kotelnikov, A., Baranchuk, D., Rubachev, I. & Babenko, A. (2022). TabDDPM: Modelling Tabular Data with Diffusion Models. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2209.15421>
- Liu, B., Qi, Y. & Chen, K.-J. (2020). Sequential online prediction in the presence of outliers and change points: An instant temporal structure learning approach. *Neurocomputing*, 413, 240–258. <https://doi.org/10.1016/j.neucom.2020.07.011>
- Liu, H. & Chen, C. (2019). Data processing strategies in wind energy forecasting models and applications: A comprehensive review. *Applied Energy*, 249, 392–408. <https://doi.org/10.1016/j.apenergy.2019.04.188>
- Liu, X. & Ichise, R. (2017). Food Sales Prediction with Meteorological Data - A Case Study of a Japanese Chain Supermarket. *Data Mining and Big Data. Second International Conference, DMBD 2017, 10387*, 93–104. https://doi.org/10.1007/978-3-319-61845-6_10
- Liu, Y., Chen, T. & Chen, J. (2015). Auto-Switch Gaussian Process Regression-Based Probabilistic Soft Sensors for Industrial Multigrade Processes with Transitions. *Industrial & Engineering Chemistry Research*, 54(18), 5037–5047. <https://doi.org/10.1021/ie504185j>
- Liu, Y. & Gao, Z. (2015). Real-time property prediction for an industrial rubber-mixing process with probabilistic ensemble Gaussian process regression models. *Journal of Applied Polymer Science*, 132(6), 41432. <https://doi.org/10.1002/app.41432>
- Lloyd, J., Duvenaud, D., Grosse, R., Tenenbaum, J. & Ghahramani, Z. (2014). Automatic Construction and Natural-Language Description of Nonparametric Regression Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 28(1), 1242–1250. <https://doi.org/10.1609/aaai.v28i1.8904>
- Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J. & Király, F. J. (2019). sktime: A Unified Interface for Machine Learning with Time Series. *Workshop on Systems for ML at NeurIPS 2019*.
- Lu, J., Liu, A., Dong, F., Gu, F., Gama, J. & Zhang, G. (2019). Learning under Concept Drift: A Review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12), 2346–2363. <https://doi.org/10.1109/TKDE.2018.2876857>
- Makridakis, S. & Hibon, M. (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4), 451–476. [https://doi.org/10.1016/S0169-2070\(00\)00057-1](https://doi.org/10.1016/S0169-2070(00)00057-1)

- Makridakis, S., Spiliotis, E. & Assimakopoulos, V. (2018). The M4 Competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4), 802–808. <https://doi.org/10.1016/j.ijforecast.2018.06.001>
- Makridakis, S., Spiliotis, E. & Assimakopoulos, V. (2020). The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. <https://doi.org/10.1016/j.ijforecast.2019.04.014>
- Makridakis, S., Spiliotis, E. & Assimakopoulos, V. (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364. <https://doi.org/10.1016/j.ijforecast.2021.11.013>
- Makridakis, S. & Wheelwright, S. C. (1989). *Forecasting methods for management* (5th ed.). Wiley.
- Makridakis, S., Wheelwright, S. C. & Hyndman, R. J. (1998). *Forecasting: Methods and applications* (3rd ed.). Wiley.
- Martínez, F., Frías, M. P., Pérez, M. D. & Rivera, A. J. (2019). A methodology for applying k-nearest neighbor to time series forecasting. *Artificial Intelligence Review*, 52(3), 2019–2037. <https://doi.org/10.1007/s10462-017-9593-z>
- Mediavilla, M. A., Dietrich, F. & Palm, D. (2022). Review and analysis of artificial intelligence methods for demand forecasting in supply chain management. *Procedia CIRP*, 107, 1126–1131. <https://doi.org/10.1016/j.procir.2022.05.119>
- Meisenbacher, S., Turowski, M., Phipps, K., Rätz, M., Müller, D., Hagenmeyer, V. & Mikut, R. (2022). Review of automated time series forecasting pipelines. *WIREs Data Mining and Knowledge Discovery*, 12(6), 1475. <https://doi.org/10.1002/widm.1475>
- Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239), 76–91.
- Moez, A. (2023). PyCaret: An open source, low-code machine learning library in Python [PyCaret version 3.0].
- Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.
- Murphy, K. P. (2023). *Probabilistic Machine Learning: Advanced Topics*. MIT Press.
- Ni, W., Tan, S. K., Ng, W. J. & Brown, S. D. (2012). Moving-Window GPR for Nonlinear Dynamic System Modeling with Dual Updating and Dual Preprocessing. *Industrial & Engineering Chemistry Research*, 51(18), 6416–6428. <https://doi.org/10.1021/ie201898a>
- Page, E. S. (1954). Continuous Inspection Schemes. *Biometrika*, 41(1-2), 100–115. <https://doi.org/10.1093/biomet/41.1-2.100>

- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., . . . Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 8024–8035.
- Pecini, A. (2023). Synthesizing Time-Series Data with Diffusion Models (Master's thesis). Technical University of Munich, School of Computation, Information and Technology.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Peng, B., Song, H. & Crouch, G. I. (2014). A meta-analysis of international tourism demand forecasting and implications for practice. *Tourism Management*, 45, 181–193. <https://doi.org/10.1016/j.tourman.2014.04.005>
- Perez-Cruz, F., van Vaerenbergh, S., Murillo-Fuentes, J. J., Lazaro-Gredilla, M. & Santamaria, I. (2013). Gaussian Processes for Nonlinear Signal Processing: An Overview of Recent Advances. *IEEE Signal Processing Magazine*, 30(4), 40–50. <https://doi.org/10.1109/MSP.2013.2250352>
- Priyadarshi, R., Panigrahi, A., Routroy, S. & Garg, G. K. (2019). Demand forecasting at retail stage for selected vegetables: a performance analysis. *Journal of Modelling in Management*, 14(4), 1042–1063. <https://doi.org/10.1108/JM2-11-2018-0192>
- Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Hardt, M., Liu, P. J., Liu, X., Marcus, J., Sun, M., Sundberg, P., Yee, H., Zhang, K., Zhang, Y., Flores, G., Duggan, G. E., Irvine, J., Le, Q., Litsch, K., . . . Dean, J. (2018). Scalable and accurate deep learning with electronic health records. *npj Digital Medicine*, 1(1), 18. <https://doi.org/10.1038/s41746-018-0029-1>
- Rasmussen, C. E. & Williams, C. (2006). *Gaussian Processes for Machine Learning* (2nd ed.). MIT Press.
- Rossi, B. (2013). Advances in Forecasting under Instability. *Handbook of Economic Forecasting*, 2, 1203–1324. <https://doi.org/10.1016/B978-0-444-62731-5.00021-X>
- Saatçi, Y., Turner, R. & Rasmussen, C. E. (2010). Gaussian Process Change Point Models. *Proceedings of the 27th International Conference on Machine Learning*, 927–934.

- Salinas, D., Flunkert, V., Gasthaus, J. & Januschowski, T. (2020). DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3), 1181–1191. <https://doi.org/10.1016/j.ijforecast.2019.07.001>
- Sankaran, S. (2014). Demand forecasting of fresh vegetable product by seasonal ARIMA model. *International Journal of Operational Research*, 20(3), 315–330. <https://doi.org/10.1504/IJOR.2014.062453>
- Savitzky, A. & Golay, M. J. E. (1964). Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8), 1627–1639. <https://doi.org/10.1021/ac60214a047>
- Seabold, S. & Perktold, J. (2010). statsmodels: Econometric and statistical modeling with python. *Proceedings of the 9th Python in Science Conference (SciPy 2010)*, 92–96. <https://doi.org/10.25080/Majora-92bf1922-011>
- Seaman, B. & Bowman, J. (2022). Applicability of the M5 to Forecasting at Walmart. *International Journal of Forecasting*, 38(4), 1468–1472. <https://doi.org/10.1016/j.ijforecast.2021.06.002>
- Sharadga, H., Hajimirza, S. & Balog, R. S. (2020). Time series forecasting of solar power generation for large-scale photovoltaic plants. *Renewable Energy*, 150, 797–807. <https://doi.org/10.1016/j.renene.2019.12.131>
- Shukla, M. & Jharkharia, S. (2011). ARIMA models to forecast demand in fresh supply chains. *International Journal of Operational Research*, 11(1), 1–18. <https://doi.org/10.1504/IJOR.2011.040325>
- Shumway, R. H. & Stoffer, D. S. (2000). *Time Series Analysis and Its Applications*. Springer. <https://doi.org/10.1007/978-1-4757-3261-0>
- Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1), 75–85. <https://doi.org/10.1016/j.ijforecast.2019.03.017>
- Snoek, J., Larochelle, H. & Adams, R. P. (2012). Practical Bayesian Optimization of Machine Learning Algorithms. *Advances in Neural Information Processing Systems*, 25, 2951–2959.
- Song, H. & Li, G. (2008). Tourism demand modelling and forecasting - A review of recent research. *Tourism Management*, 29(2), 203–220. <https://doi.org/10.1016/j.tourman.2007.07.016>
- Song, H., Qiu, R. T. R. & Park, J. (2019). A review of research on tourism demand forecasting: Launching the Annals of Tourism Research Curated Collection on

- tourism demand forecasting. *Annals of Tourism Research*, 75, 338–362. <https://doi.org/10.1016/j.annals.2018.12.001>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56), 1929–1958.
- Takeuchi, J. & Yamanishi, K. (2006). A unifying framework for detecting outliers and change points from time series. *IEEE Transactions on Knowledge and Data Engineering*, 18(4), 482–492. <https://doi.org/10.1109/TKDE.2006.1599387>
- Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1), 267–288. <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>
- Tipping, M. E. (2001). Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1, 211–244.
- Tomašev, N., Harris, N., Baur, S., Mottram, A., Glorot, X., Rae, J. W., Zielinski, M., Askham, H., Saraiva, A., Magliulo, V., Meyer, C., Ravuri, S., Protsyuk, I., Connell, A., Hughes, C. O., Karthikesalingam, A., Cornebise, J., Montgomery, H., Rees, G., . . . Mohamed, S. (2021). Use of deep learning to develop continuous-risk models for adverse event prediction from electronic health records. *Nature Protocols*, 16, 2765–2787. <https://doi.org/10.1038/s41596-021-00513-5>
- Tsoumakas, G. (2019). A survey of machine learning techniques for food sales prediction. *Artificial Intelligence Review*, 52(1), 441–447. <https://doi.org/10.1007/s10462-018-9637-z>
- Turgut, Y. & Erdem, M. (2022). Forecasting of Retail Produce Sales Based on XGBoost Algorithm. *Industrial Engineering in the Internet-of-Things World. Global Joint Conference on Industrial Engineering and Its Application Areas*, 27–43. https://doi.org/10.1007/978-3-030-76724-2_3
- Williams, C. & Rasmussen, C. E. (1995). Gaussian Processes for Regression. *Advances in Neural Information Processing Systems*, 8, 514–520.
- Winters, P. R. (1960). Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science*, 6(3), 324–342. <https://doi.org/10.1287/mnsc.6.3.324>
- Witt, S. F. & Witt, C. A. (1995). Forecasting tourism demand: A review of empirical research. *International Journal of Forecasting*, 11(3), 447–475. [https://doi.org/10.1016/0169-2070\(95\)00591-7](https://doi.org/10.1016/0169-2070(95)00591-7)

- Wu, D. C., Song, H. & Shen, S. (2017). New developments in tourism and hotel demand modeling and forecasting. *International Journal of Contemporary Hospitality Management*, 29(1), 507–529. <https://doi.org/10.1108/IJCHM-05-2015-0249>
- Xu, L., Skoularidou, M., Cuesta-Infante, A. & Veeramachaneni, K. (2019). Modeling Tabular data using Conditional GAN. *Advances in Neural Information Processing Systems*, 32, 7335–7345.
- Yan, X. & Su, X. G. (2009). *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co. Pte. Ltd. <https://doi.org/10.1142/6986>
- Yang, C.-L. & Sutrisno, H. (2018). Short-Term Sales Forecast of Perishable Goods for Franchise Business. *10th International Conference on Knowledge and Smart Technology (KST)*, 101–105. <https://doi.org/10.1109/KST.2018.8426091>
- Yasrebi-de Kom, I. A. R., Dongelmans, D. A., de Keizer, N. F., Jager, K. J., Schut, M. C., Abu-Hanna, A. & Klopotowska, J. E. (2023). Electronic health record-based prediction models for in-hospital adverse drug event diagnosis or prognosis: a systematic review. *Journal of the American Medical Informatics Association*, 30(5), 978–988. <https://doi.org/10.1093/jamia/ocad014>
- Yoon, J., Jarrett, D. & van der Schaar, M. (2019). Time-series Generative Adversarial Networks. *Advances in Neural Information Processing Systems*, 32, 5508–5518.
- Zentralverband Gartenbau e.V. (2022). Jahresbericht 2022. Retrieved June 13, 2023, from <https://www.hortigate.de/publikation/93805/ZVG-Jahresbericht-2022/>
- Zhang, L., Wen, J., Li, Y., Chen, J., Ye, Y., Fu, Y. & Livingood, W. (2021). A review of machine learning in building load prediction. *Applied Energy*, 285, 116452. <https://doi.org/10.1016/j.apenergy.2021.116452>
- Zhang, X., Wu, H. & Yang, J. (2022). HyperTS: A Full-Pipeline Automated Time Series Analysis Toolkit [Version 0.2.x].
- Zhao, Z., Kunar, A., Birke, R. & Chen, L. Y. (2022). CTAB-GAN+: Enhancing Tabular Data Synthesis. *arXiv preprint*. <https://doi.org/10.48550/arXiv.2204.00401>
- Žliobaitė, I., Bakker, J. & Pechenizkiy, M. (2012). Beating the baseline prediction in food sales: How intelligent an intelligent predictor is? *Expert Systems with Applications*, 39(1), 806–815. <https://doi.org/10.1016/j.eswa.2011.07.078>
- Žliobaitė, I., Pechenizkiy, M. & Gama, J. (2016). An Overview of Concept Drift Applications. *Big Data Analysis: New Algorithms for a New Society*, 91–114. https://doi.org/10.1007/978-3-319-26989-4_4

-
- Zou, H. & Hastie, T. (2005). Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>
- Züfle, M. & Kounev, S. (2020). A Framework for Time Series Preprocessing and History-based Forecasting Method Recommendation. *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*, 141–144. <https://doi.org/10.15439/2020F101>