UNIVERSITÄT PADERBORN

Faculty of Business Administration and Economics

Working Paper Series

Working Paper No. 2026-16

Forecasting of trend stationary time series in SAP using a data-driven semiparametric ARMA model

Li Chen, Yuanhua Feng

Juni 2026

wiwi

Forecasting of trend stationary time series in SAP using a data-driven semiparametric ARMA model

Li Chen* Yuanhua Feng†

Abstract

Motivated by more and more semi- or nonparametric models applied in time series forecasting and their demonstrated superior performance in many empirical researches, this paper explores the adoption and integration of a semiparametric ARMA model in an enterprise system landscape. We begin by reviewing basic construction of the semiparametric ARMA model, the iterative plug-in algorithm for estimating the trend component of trend stationary times series, forecast techniques and quality measurements, which were well researched and published with the R package smoots. Subsequently, we showcase a novel approach to adopt the semiparametric ARMA model in a forecast application based on SAP Analytics Cloud (SAC), which leverages the platform's strengths in system integrity, state-of-the-art user interface (UI) design as well as seamless connection to a R engine with smoots package embedded. The forecast application addresses key challenges in terms of cost efficiency, user experience, and the requirement for in-house statistical or machine learning expertise while adopting such statistical algorithms in enterprise context. Finally, we empirically evaluate the forecast quality of the integrated semiparametric ARMA model using real-world data, demonstrating promising results overall.

Keywords : Time series forecasting, semiparametric algorithm, forecasting accuracy, *smoots* package, SAP Analytics Cloud, enterprise adoption

*Corresponding author. E-mail: lichen@campus.uni-paderborn.de. University of Paderborn, Faculty of Business Administration and Economics

†University of Paderborn, Faculty of Business Administration and Economics

1 Introduction

Time series forecasting has gained significant attention in both business and scientific research. While traditional parametric models such as ARIMA have established themselves in business applications, the application of nonparametric or semiparametric models, with their fully data-driven nature, is emerging as a compelling trend. Among these nonparametric approaches, local polynomial regression (LPR) has become a prominent technique for trend extraction from time series data. Feng (2004) summarized several important benefits of local polynomial regression including its easy interpretation, implementation and estimation of derivatives. Fan (1993) discovered the nice property of minimax efficiency in local linear regression estimator. As a sort of kernel approach, the problem of boundary correction was further studied and well solved (Fan and Gijbels, 1992; Hastie and Loader, 1993; Ruppert and Wand, 1994). More important is that local polynomial regression is proven to be asymptotically equivalent to certain kernel regression (Müller, 1987; Fan and Gijbels, 1992; Ruppert and Wand, 1994; Feng, 2004), which makes it possible to take use of known asymptotic properties of kernel regressions in local polynomial regression. Based on error criteria weighted mean integrated squared error (MISE) of a kernel estimator, the asymptotic optimal bandwidth can be derived (Gasser and Müller, 1979; Müller, 1988; Feng, 2004; Feng et al., 2020a). This results in a great benefit in selecting the optimal bandwidth for local polynomial regression. Feng et al. (2020a) proposed to use an slightly adjusted iterative plug-in (IPI) approach similar to Bühlmann (1996) to estimate the variance factor c_f, the optimal bandwidth h_A and ultimately the trend and its derivatives, which makes up the core benefits of a data-driven extraction of trend and its derivatives. As a second component of a semiparametric model, Feng et al. (2020a) further suggests to use a well established autoregressive moving average (ARMA) model to capture more information behind the detrended residuals. Ultimately, a R package *smoots* was developed and published on CRAN with a fully data-driven semiparametric ARMA model for fitting and forecasting time series data (Feng et al., 2020b, 2023; Schulz et al., 2021), to encourage further study as well as application of this model.

In the era of data-driven business practices, time series forecasting has become a cornerstone of enterprise decision-making processes both at strategic level and at an operational level. Statistical quantitative forecasting can be well employed to generate baseline forecasts, although practitioners rely heavily on judgemental adjustments (Syntetos et al., 2009; Fildes et al., 2009). Syntetos et al. (2009) summarized statistical forecasting algorithms used in

demand planning context, revealing the fact that algorithms of parametric art dominate till then. Another empirical survey by McCarthy et al. (2006) showed that practitioners are more familiar to straightforward techniques like moving average, exponential smoothing and trend projection. In other words, the models are strongly depend on parametric assumptions on underlying distribution of demand, which is often a challenge in the practice. In the recent review of retail forecasting by Fildes et al. (2022), clearly more semi- or nonparametric techniques were applied in recent years. More importantly, most published results have found performance improvements in forecasting accuracy by using semi- or nonparametric models like neural networks (Aburto and Weber, 2007), fuzzy neural networks (Kuo, 2001), k-nearst-neighbor and decision trees (Žliobaitė et al., 2012). However, most of these semi- or nonparametric methods face the challenge that they cannot automatically provide forecast intervals, which could be necessary to guide practical operations (Fildes et al., 2022). This status quo motivates application of the semi parametric ARMA model proposed by Feng et al. (2020a) and Schulz et al. (2021) in enterprise context, since it is semiparametric and offers an automatic point forecasts capability including confidence interval calculation alongside the *smoots* package.

Enterprise software providers like SAP have integrated several time series forecasting algorithms like ARIMA, exponential smoothing in some of their solutions like Integrated Business Planning (IBP) and SAP Analytics Cloud (SAC) to optimize operations, enhance efficiency and generate tangible value (Laboni Bhowmik and Mukherjee, 2021). Nevertheless, use of diverse algorithms, including semi-parametric methods, could be beneficial in balancing interpretability and complexity as well as in improving the robustness and/or accuracy. However, a recent survey showed that many companies failed in adopting "systematic forecasting methods (SFM)". Among all concerns in the responses, lack of familiarity with statistical or machine learning, high cost of implementation (of software) and lack trust in forecast accuracy are biggest hurdles for companies to adopt SFM (Goodwin et al., 2023). Indeed, adopting a forecasting technique in the practice involves more than robustness and accuracy of the model alone. A cost efficient implementation and integration into current system landscape can be the first criteria for managers facing such decisions. Moreover, good user experience is always a fundamental principle in business context, which requires the solution to be easy to use and easy to interpret. With these considerations, we propose a cost efficient way with an application *smootsForecast* for utilizing further custom forecast algorithms, in our case the algorithm in *smoots* package, based on SAP Analytics Cloud (SAC), which is one of the leading solution in terms of enterprise forecast and planning. The

architecture and design of the application follows TOGAF framework, one the most popular frameworks for enterprise architecture design (The Open Group, 2022). Cost efficiency in implementation and integration into existing system landscape can be well solved by leveraging standard features of SAC. Users are able to write back forecasting results directly to databases for further enterprise process integration. We further propose two different modes for better user experience for beginning user groups with less statistical understanding and those with advanced understanding. With regard to evaluating forecast accuracy, the statistic mean absolute percentage error (MAPE) is calculated in addition to mean absolute scaled error (MASE) and root mean squared scaled error (RMSSE) from *smoots* package, because MAPE is widely used in other SAP solutions. Results of forecast backtesting are visualized in a traffic light logic to enable intuitive interpretation.

This paper is structured as follows. We begin with an introduction, followed by a concise review of relevant findings from prior research, laying the groundwork for the semiparametric ARMA model, which is comprehensively described in Section 3. Forecasting techniques based on the semiparametric model are then presented in Section 4. Subsequently, Section 5 outlines the architecture and design of the *smootsForecast* application, followed by illustrative examples using selected datasets. Finally, Section 7 concludes the paper with a summary of findings, concluding remarks, and a discussion of potential avenues for future research.

2 A review of some results on trend extraction

This section aims to review some fundamental concepts of times series analysis, previous results on nonparametric trend extraction and forecast approaches, upon which the semiparametric ARMA model in Section 3 are built. It begins by reviewing the classical time series decomposition concept, which typically involves separating a time series into trend, seasonal, and residual components. By omitting the seasonal component, the focus shifts to estimating and extracting the trend component from the time series. Subsequently, kernel estimators and their associated asymptotic properties for optimal bandwidth selection are discussed. Finally, local polynomial estimators are summarized with emphasis on their asymptotic equivalence to kernel estimators.

2.1 Decomposition of time series

As time series often have some patterns like trend, cycles and seasonality, decomposition of a time series into different components is one of the central topics in time series analysis and forecasting. While cycles captures fluctuations with not fixed frequency, seasonality describes patterns with a fixed frequency. In some researches, trend and cyclical components are combined and referred to trend component for simplicity, even though it may also include certain cyclical behavior. Furthermore, as seasonality can be seen as a special case of cycles where the period is fixed and the pattern repeats at regular intervals (Chatfield, 2003), this paper ignores seasonality component during modelling. Starting from the 1920s, times series are deconstructed using either additive or multiplicative approach. These two forms of decomposition, the so called classical decomposition method, become foundational techniques for most other methods of time series decomposition (Hyndman and Athanasopoulos, 2018).

An additive model assumes that the observed time series y_t is the sum of its components like $y_t = m_t + \xi_t$ where m_t represents the trend component, and ξ_t the error or residual component. This model is suitable when the seasonal fluctuations and residual variations are constant over time. On the other hand, a multiplicative model assumes that the observed time series y_t is the product of its components with form of $y_t = m_t \times \xi_t$. This model is appropriate when the seasonal variations and residual variations scale with the level of the time series. An interesting fact is that a multiplicative model can be transformed into an additive model by transforming the data first until the variation in the series become stable over time. When a log transformation is conducted to the data before, an additive model on the transformed data will be equivalent to a multiplicative model on original data due to the fact that $y_t = m_t \times \xi_t \Rightarrow \log y_t = \log m_t + \log \xi_t$. Based on the decomposition, estimating of y_t can be split into estimating the trend component m_t and the residual term ξ_t after extracting the trend estimate $\hat{m}_t$ from original time series y_t.

2.2 Kernel regression and some asymptotic properties

After decomposition of time series, estimation of the components becomes the key issue. As the component trend is often not well fitted by parametric approaches, researchers turned to develop nonpaprametric regression, among which kernel regression was proposed and well studied, as summarized by Collomb (1985). Given a time series y_t with observations at time point $t = 1, 2, \ldots, n$, where n is the number of observations, the basic idea of kernel regression

is to estimate y_t without a parametric assumption. Instead, the estimation at any point t_0 is made using a weighted average of nearby observations, where the weights decrease with distance from t_0.

Nadaraya (1964) and Watson (1964) proposed firstly the Nadaraya-Watson estimator (NW-estimator) which is basically a global weighted average based on a selected kernel function and a bandwidth parameter affecting the smoothness. If bandwidth is too small, the estimate may be capture too much noise and tend to be overfitting. If bandwidth is too large, the estimate not capture enough of the underlying pattern. With increasing bandwidth, the variance of the estimate is getting smaller, while the bias is growing. Gasser and Müller (1979) proposed an another kernel-based estimator, the so called GM-estimator, using integrated kernel as weights, which is clearly better than the NW estimator for a regular fixed design described by Feng (2004) (Section 2.4) (Gasser et al., 1991; Herrmann and Gasser, 1994). Following the GM estimator, a kernel estimator of the ν-th derivative can also be estimated.

In addition to the bandwidth h, the shape of the kernel function K also affects the performance of the kernel regression estimate. Widely used second order kernels with compact support $supp(K) = [-1, 1]$ are of form $K_\mu(u) = C_\mu(1-u^2)^\mu \mathbf{I}_{[-1,1]}(u)$, where μ is a nonnegative integer, represents the *degree of smoothness of a kernel function* of this type. $\mathbf{I}_A(u)$ is the indicator function of the interval A and C_μ is given by $\int K(u)du = 1$. The most important kernel polynomials for order of kernel k and μ were listed by Müller (1988). To select a global optimal bandwidth, Feng (2004) used the error criteria weighted mean integrated squared error (MISE), which integrates the local error criteria mean squared error (MSE). Due to the fact that MSE includes a variance component and a squared bias component, MISE can be further expanded using the asymptotic variance and bias of GM estimator given by Müller (1988). By minimizing MISE, the asymptotically optimal bandwidth can be obtained.

2.3 Local polynomial regression (LPR) and its asymptotic equivalence to kernel regression

Locally weighted regression is another popular nonparametric technique. Unlike global regression, local regression like local polynomial regression (LPR) fits many polynomials to localized subsets of the data. For a given time series y_t with observations at time point $t = 1, 2, \ldots, n$, where n is the number of observations, a local polynomial regression can be

written as $m(x) \doteq \mathbf{x}^\top \boldsymbol{\beta}(x_0)$, where $\mathbf{x} = (1, x, \ldots, x^p)^\top$, $\beta(x_0) = \big(\beta_0(x_0), \beta_1(x_0), \ldots, \beta_p(x_0)\big)^\top$ is the vector of local regression coefficients depending on x_0. The local polynomial fitting at x_0 can be obtained by minimizing the weighted least squares.

Müller (1987) proposed a local polynomial estimator $\hat{m}^{(\nu)}$ as $\hat{m}^{(v)}(x) = \mathbf{w}^{(\nu)}(x)^\top \mathbf{y} = \sum_{i=1}^n w_i^{(\nu)}(x) y_t$,where $\mathbf{w}^{(\nu)}(x) = \left(w_1^{(\nu)}(x), \ldots, w_n^{(\nu)}(x)\right)^\top$ is a vector of local weight function with compact support $x \in [0, 1]$. Furthermore, Müller (1987), Fan and Gijbels (1992) and Ruppert and Wand (1994) studied and proved that local polynomial estimator can be asymptotically equivalent to certain kernel estimators in case of a regular fixed design. Fan and Gijbels (1992) addressed the boundary issue by introducing boundary correction techniques. Their approach involves modifying the weight function used in the local fitting process to give more influence to points closer to the boundary. Specifically, they proposed the use of one-sided kernels, which adjust the weights asymmetrically near the boundaries, thus providing more robust estimates. Hastie and Loader (1993) expanded on these ideas by proposing an automatic boundary correction method. Their technique adjusts the bandwidth of the kernel function dynamically as it approaches the boundary, ensuring that the local fitting remains stable and unbiased. This automatic adjustment makes the boundary correction process more efficient and easier to implement in practical applications. The work of Ruppert and Wand (1994) also contributed significantly to the development of boundary correction methods. They introduced a locally weighted least squares regression framework that incorporates boundary correction by using polynomial weights that vary with the distance from the boundary. This method provides a smooth transition in the weighting scheme, enhancing the stability and accuracy of the local polynomial estimates. For a known design density f, the kernel estimator of $m^{(\nu)}$ at a local time point x_0 is asymptotically equivalent to the local polynomial estimator (See Theorem 3.2 by Feng (2004)). It's important to note that this asymptotic equivalence holds not only for interior points x_0, but also for boundary points in case a corresponding boundary kernel is used according to Müller (1988).

3 The semiparametric ARMA model and automatic trend estimation

In this section, we provide a detailed description of the semiparametric ARMA model proposed by Feng et al. (2020a). Leveraging the asymptotic equivalence between local polynomial regression and kernel methods, we derive the optimal bandwidth for the nonparametric

component. Subsequently, an iterative plug-in algorithm is employed for bandwidth selection and estimation of the nonparametric component. For modeling the residuals, a parametric ARMA model is adopted.

3.1 Construction of the semiparametric ARMA model

For a given time series y_t with observations at time point $t = 1, 2, \ldots, n$, where n is the number of observations. Due to the fact that an additive decomposition can be transformed by taking logarithm of multiplicative components, an additive component model is assumed for y_t. Under this assumption, y_t can be modeled with a nonparametric component for trend and a parametric model like ARMA for residuals. This could be helpful for analyzing time series, which are not stationary, but have a deterministic trend. The model can be formulated as

$$y_t = m(x_t) + \xi_t \tag{1}$$

where $x_t = t/n$ denotes the rescaled time on the interval $[0, 1]$, m is a smooth deterministic trend function and ξ_t is a zero mean stationary process with autocovariances $\gamma_\xi(l) = cov(\xi_1, \xi_{l+1}) = E(\xi_1 \xi_{l+1})$. Following this definition, it's essential to estimate m or its ν-th derivative $m^{(\nu)}$ and the residual ξ_t. To estimate m or its ν-th derivative $m^{(\nu)}$ with short-term dependent errors, an iterative plug-in algorithm similar to that proposed by Bühlmann (1996) can be taken into use, given assumption that $\gamma_\xi(l)$ converge to zero quickly so that $\sum_{l=0}^{\infty}(l+1)^4|\gamma_\xi(l)| < \infty$. For the residual ξ_t after removing the estimated trend, an autoregressive moving average (ARMA) model with autoregressive order of p and moving average order of q can be assumed as below:

$$\xi_t = \phi_1 \xi_{t-1} + \phi_2 \xi_{t-2} + \cdots + \phi_p \xi_{t-p} + \theta_1 u_{t-1} + \theta_2 u_{t-2} + \cdots + \theta_q u_{t-q} + u_t \tag{2}$$

where u_t fulfills the assumptions of an independently identically distributed ($i.i.d.$) white noise process with zero mean and a constant variance of σ_u^2. Furthermore, θ and ϕ are coefficients of the underlying AR and MA processes respectively. As suggested by Feng et al. (2020a), fitting such a semiparametric ARMA model involves two steps. Firstly, the deterministic trend will be estimated. For this purpose, local polynomial regression (Ruppert and Wand, 1994; Cleveland and Loader, 1996; Fan and Gijbels, 1996) and an slightly adjusted iterative plug-in (IPI) algorithm for selecting optimal bandwidth can be used. As reviewed in Section 2.3, the problem of boundary effect was well solved, this approach combines the flexibility of local polynomial regression and optimization of bandwidth for preventing any over-fitting.

Secondly, the residual time series, obtained by deducting the estimated trend series, will be estimated using the well-established ARMA model.

3.2 The asymptotically optimal bandwidth

A local polynomial estimator of $m^{(\nu)}$ and the ν-th derivative of m need to be investigated, since the asymptotic optimal bandwidth can be written as a function of derivatives of m. Let m be at least $(p+1)$-times differentiable at a point $x_0 \in X$ with $X = \{x_1, x_2, \ldots, x_n\}$, $\nu < p$ and $p - \nu$ is odd, the local polynomial estimator of $m^{(\nu)}$ turns into solving the locally weighted least squares problem

$$Q(x_0) = \sum_{t=1}^{n} \left\{ y_t - \sum_{j=0}^{p} \beta_j (x_t - x_0)^j \right\}^2 W(\frac{x_t - x_0}{h}) \Rightarrow min \tag{3}$$

at point x_0. In Formula 3, p is order of the polynomial, h is the bandwidth and W is a weighting function, i.e. a second order kernel function on $[-1, 1]$. Let $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p,)^T$. Following this idea, $\hat{m}^{(\nu)}(x_0) = \nu! \hat{\beta}_\nu$ is an estimator of $m^{(\nu)}(x_0)$.

To measure quality of the local polynomial estimator, the mean integrated squared error (MISE)

$$MISE(h) = \int_{c_b}^{d_b} [\hat{m}^{(\nu)}(x_t) - m^{(\nu)}(x_t)]^2 dx_t \tag{4}$$

is employed, where c_b, d_b are boundaries cutoffs to correct the boundary effect and $0 < c_b < d_b < 1$ applies. Let $K_{(\nu,k)}(u)$ be the k-th order equivalent kernel function that is realized for the estimation of $m^{(\nu)}$ at an interior point with $k = p + 1$. Under regularity conditions, the asymptotic MISE (AMISE) of the local polynomial estimator can be written as

$$AMISE(h) = h^{2(k-\nu)} \frac{I[m^{(k)}]\beta^2}{[k!]^2} + \frac{2\pi c_f (d_b - c_b) R(K)}{n h^{2\nu+1}} \tag{5}$$

according to Theorom 2 in Beran and Feng (2002a), where c_f is the variance factor and $c_f = f(0)$ with

$$f(\lambda) = \frac{1}{2\pi} \sum_{l=-\infty}^{\infty} \gamma_\xi(l) e^{-il\lambda}, -\pi \leq \lambda \leq \pi \tag{6}$$

being the spectral density of ξ_t, $I[m^{(k)}] = \int_{c_b}^{d_b} [m^{(k)}(x)]^2 dx$, $\beta_{(\nu,k)} = \int_{-1}^{1} u^k K(u) du$ and $R(K) = \int K^2(u) du$. The asymptotically optimal bandwidth can be derived as

$$h_A = C_A n^{-1/(2k+1)} \tag{7}$$

with

$$C_A = \left[\frac{2\nu + 1}{2(k - \nu)} \times \frac{2\pi c_f [k!]^2 (d_b - c_b) R(K)}{I[m^{(k)}] \beta^2_{(\nu,k)}} \right]^{1/(2k+1)} \tag{8}$$

. Provided that $I\left[m^{(k)}\right] > 0$. The use of $h = O(h_A)$ leads to $MISE = O(n^{-2(k-\nu)/(2k+1)})$, which is the optimal rate of convergence for any parametric regression estimator of a k-times continuously differentiable regression function m.

3.3 Estimation of c_f following IPI

In the formula for h_A, only c_f and $I[m^{(k)}]$ remain unknown. Consequently, determining the optimal bandwidth hinges on obtaining suitable estimates for these two quantities. Notably, for a given bandwidth, $I[m^{(k)}]$ is unaffected by error correlation and can be estimated using established methods designed for models with independent errors. Therefore, the primary challenge lies in estimating the variance factor c_f.

Following Bühlmann (1996), let $r_{t,V} = y_t - \hat{m}_V$ denote the residuals obtained from a pilot estimate using a bandwidth h_V and $\hat{\gamma}(l)$ the sample autocovariances obtained from $r_{t,V}$. Feng et al. (2020a) proposed a lag-window estimator of c_f in form of

$$\hat{c}_{f,M} = \frac{1}{2\pi} \sum_{l=-M}^{M} w_l \hat{\gamma}(l) \tag{9}$$

under Bartlett-window with M being window-width and $w_l = l/(M + 0.5)$ being Bartlett-window weights. According to Feng and Heiler (2009) and Feng et al. (2020a), the asymptotically optimal bandwidth for estimating $\gamma(l)$ can be written as

$$h_{\gamma,A} = CF h_A \tag{10}$$

where CF is the so called bandwidth correction factor defined by

$$CF = \left\{ 2k \left[\frac{2K(0)}{R(K)} - 1 \right] \right\}^{1/(2k+1)} \tag{11}$$

. Since h_A and $R(K)$ are as given in Formula 7, CF only depends on the equivalent kernel K. Values for CF with corresponding kernel functions can be found in Table 1 in Feng and Heiler (2009). Noting that CF values are always bigger than 1, Feng et al. (2020a) suggested to estimate $\hat{\gamma}(l)$ from the residuals of $\hat{m}_\gamma$ obtained using the bandwidth $\hat{h}_\gamma = CF\hat{h}$. This is especially helpful with small size of observations n. Finally, c_f can be estimated following iterative plug-in algorithm as below:

1. Select an initial window-width $M_0 = [n/2]$, where $[\cdot]$ extracts the integer part.

2. Define j as counter of iterations. In the j-th iteration, $\int (f(\lambda)^2)d\lambda$ is to be estimated following IPI by Bühlmann (1996). Put $M'_j = [M_{j-1}/n^{2/21}]$ and estimate $\int f^{(1)}(\lambda)d\lambda$ using the window-width M'_j, where $\int f^{(1)}(\lambda)d\lambda$ is the integral of the first derivative of $f(\lambda)$. Based on the estimate of the first derivative, M_j can be calculated.

3. Repeat the step 2 above until the selected M converges or a maximal times of iteration like 20 is reached. Denote the selected M by $\hat{M}_G$.

4. Local adaptation at $\lambda = 0$: Let $M' = [\hat{M}_G/n^{2/21}]$ and calculate $\int f^{(1)}(\lambda)d\lambda$ using this window-width to get the finally selected window-width $\hat{M}$.

Bühlmann (1996) studied the asymptotic properties of $\hat{M}$ in his **Theorem 1** under Assumptions B1 to B3. While B2 and B3 are more common requirements on lag-window and the spectral density of ξ_t, his assumption B1 is more strict. Under condition B1, the cumulants of ξ_t are absolutely summable up to order 8, implying in particular the existence of the eighth order moments of ξ_t. For the Bartlett-window, the B1 further assumes that $\gamma_\xi(l)$ converge to zero quickly such that $\sigma_{l=0}^{\infty}(l+1)^4|\gamma_\xi(l)| < \infty$. Under these conditions, $\hat{M}$ converges to M in a relative rate of the order $O(n^{-2/7})$, resulting in an estimate of c_f with the rate of convergence of order $O(n^{-1/3})$. Feng et al. (2020a) adopted a smaller inflation factor $n^{-2/21}$ instead of $n^{-4/21}$ by Bühlmann (1996), which improved the rate of bias and, however, may leads to more iterations. Furthermore, Feng et al. (2020a) proposed to start with a large window-width such that the resulting estimate is more stable. Due to the fact that a selected window-width will usually not be affected by M_0 if it converges, M_0 can be selected as any $1 \leq M_0 \leq [n/2]$.

3.4 Estimation of m and $m^{(\nu)}$ following IPI

Following the idea of (Beran and Feng, 2002a,b; Feng et al., 2020a), selecting the bandwidth for local linear and local cubic estimators $\hat{m}$, with $k = 2$ and 4 respectively involves following steps:

1. Select an initial bandwidth h_0.

2. Define j as counter of iterations with $j = 1, \ldots,$ estimate m using CFh_{j-1}. Subsequently, $\hat{c_f}$ can be estimated from the corresponding residuals using data-driven lag-

window estimator described before.

3. Set $h_{d,j} = h_{j-1}^{\alpha}$ with $\alpha = 5/7$ or $5/9$ for $p = 1$, and $\alpha = 9/11$ or $9/13$ for $p = 3$. Estimate $I[m^{(k)}]$ with $h_{d,j}$ and a local polynomial of order p_d respectively. Calculate h_j using Formula 7 and 8.

4. Repeat step 2 and 3 until selected h_j converges or the maximal number of iterations is reached. Finally set the estimate of optimal bandwidth $\hat{h}$ to h_j.

According to the proposal of Feng et al. (2020a), the IPI-algorithm can easily be extended to select bandwidth for estimating $m^{(\nu)}$. Considering an economic use case, $m^{(\nu)}$ could be log-linear growth trends and growth rate of sales. The procedures for $\nu = 1$ and $\nu = 2$ with $p = \nu + 1$ and $k = \nu + 2$ were built into *smoots* package as below:

1. Estimate m of the order p_p, e.g. $p_p = 1$ or $p_p = 3$ to obtain the estimate $\hat{c}_f$.

2. Fix the estimate for $\hat{c}_f$ and run a similar IPI-procedure to select the bandwidth for estimating $m^{(\nu)}$.

3.5 Fitting a parametric ARMA model for residuals

After estimating m, the residual part can be calculated by $r_t = \hat{\xi}_t = y_t - \hat{m}(x_t)$. As the trend component is now extracted, the residual series r_t become a trend stationary process, which can be estimating using popular parametric models like AR or ARMA. Denote θ as the parameter vector of the stationary component model. It's important to mention that, according to Beran and Feng (2002b) and Feng et al. (2020a) although the orders of magnitude of the bias and variance in $\hat{m}$ can lead to approximate innovations in the residual series, their impact on the estimated parameters remains of the order $o_p(n^{-1/2})$ caused by the average effect of the log-likelihood function $\hat{L}(\theta)$, with $L(\cdot)$ being the log-likelihood function defined based on the detrended residual series ξ_t. Let $\tilde{\theta}$ denote the parameter estimator of residual series ξ_t, according to Feng (2004), following equations applies under common regularity conditions.

$$\hat{\theta} - \tilde{\theta} = O_p[\hat{L}^{(1)}(\tilde{\theta})] \tag{12}$$

If a bandwidth of form $h = O(n^{-\alpha})$ with $1/(4k) < \alpha < 1/2$ is applied, $\hat{L}^{(1)}(\tilde{\theta}) = o_p(n^{-1/2})$. In this case, the constraint $1/(4k) < \alpha$ ensures that the additional bias in $\hat{\theta}$ due to the bias of $\hat{m}$ is negligible and the condition $\alpha < 1/2$ makes sure that the bias in $\hat{\theta}$ caused by the

variance of $\hat{m}$ can be ignored. Feng proofed this results in his Theorem 5.2 in Feng (2004). It's important to mention that these conditions are always fulfilled by the asymptotically optimal bandwidth $h_A = O[n^{-1/(2k+1)}]$ or by any consistently selected bandwidth $\hat{h} = h_A[1 + o(1)]$.

4 Forecasting based on semiparametric ARMA model

Given the construction of the proposed semiparametric ARMA model, direct calculation of point forecasts is not feasible due to the absence of available parameters. However, alternative approaches for computing point forecasts and their corresponding confidence intervals can be explored. Specifically, point forecasts and their corresponsding confidence intervals under normal errors and non-normal errors are investigated following Schulz et al. (2021) and Fritz et al. (2018). Furthermore, backtesting techniques are employed to rigorously evaluate the accuracy of the generated forecasts.

4.1 Point forecasts

Given a process y_t modeled as semiparametric ARMA model above, the point forecast for a future time point $t = n + k$ with $k = 1, 2, \ldots$ can be written as $\hat{y}_{n+k}$. Considering the components in semiparametric ARMA model, it's clear that a proper point forecast $\hat{y}_{n+k}$ can be derived based on estimates of its components for trend $\hat{m}(x_t)$ and the residuals $\hat{\xi} = y_t - \hat{m}(x_t)$ by

$$\hat{y}_{n+k} = \hat{m}(x_{n+k}) + \hat{\xi}_{n+k} \tag{13}$$

For the non-parametric trend component, Fritz et al. (2018) proposed an approach using a linear extrapolation. Let D be a dummy variable which equals to either 0 or 1, point forecasts of trend component can then be calculated by

$$\hat{m}(x_{n+k}) = \hat{m}(x_n) + D \times k \times \Delta m \tag{14}$$

where $\Delta m = \hat{m}(x_n) - \hat{m}(x_{n-1})$. If D equals to 1, the trend component will be extended linearly to future k periods. If D takes the value 0, the trend component will stay constantly.

For the parametric ARMA component, there are already well-eatablished linear predictions (see e.g. Section 3.3 in Brockwell and Davis (2016). According to Feng and Zhou (2015), additional errors within best linear predictions resulted from errors in the estimated trend function are asymptotically negligible. Therefore, Fritz et al. (2018) and Schulz et al.

(2021) proposed to obtain a approximately best linear predictions from an ARMA model fitted to $\hat{\xi}$, specifically by

$$\hat{\xi}_{n+k} = \sum_{i=1}^{p} \hat{\phi}_i \hat{\xi}_{n+k-i} + \sum_{j=1}^{q} \hat{\theta}_j \hat{u}_{n+k-j} \tag{15}$$

where $\hat{\xi}_{n+k-i}$ are forecast values for $k-i>0$. However, if $k-i \leq 0$, $\hat{\xi}_{n+k-i}$ should be the actual values of the detrended series. The same applies also for the innovation component $\hat{u}_{n+k-j}$ with $E(u_t)=0$ for $k-i>0$.

4.2 Forecast interval under normal errors

Under the assumption that the errors follow a normal distribution, the interval around the forecast can be derived from by adding and subtracting a margin derived from the forecast error variance.

The error in $\hat{m}(x_{n+k})$ is of order $O(n^{-2/5})$, can be omitted for simplicity as proposed by Fritz et al. (2018). If the ARMA process is stationary, it admits an $MA(\infty)$ representation $\xi_t = \sum_{i=0}^{\infty} c_i u_{t-i}$ with $c_0 = 1$ and $\sum_{i=0}^{\infty} |c_i| < \infty$. Based on this representation, point forecast can be given by $\xi_{n+j} = \hat{\xi}_{n+j} + \sum_{i=0}^{j-1} c_i u_{n+i}$, which leads to the asymptotic mean squared error of a point forecast at future time point $n+k$ being $\tilde{\sigma}_k^2 = \sigma_\mu^2 \sum_{i=0}^{k-1} c_i^2$ as equation (3.3.19) in Brockwell and Davis (2016). Under assumption of normality, i.e. $u_t \sim N(0, \sigma_u^2)$ and omission of error in point forecasts of trend component, the forecast intervals for point forecast $\hat{y}_{n+k}$ at a confidence level α can be obtained by

$$[\hat{y}_{n+k} - \Phi(1-\alpha^*/2)\tilde{\sigma}_k, \hat{y}_{n+k} + \Phi(1-\alpha^*/2)\tilde{\sigma}_k] \tag{16}$$

with $\alpha^* = 1-\alpha$ and $\Phi(i)$ is the $(i \times 100)\%$-quantile of the standard normal distribution.

4.3 Forecast interval under unknown errors

As the error term does not necessarily follows a normal distribution, other approaches, especially the so called forward bootstrap, were introduced and developed. Pascual et al. (2004) firstly proposed a forward bootstrap approach in context of ARIMA processes. Later on, Pan and Politis (2016) extended the idea for autoregressive (AR) processes of order p and Lu and Wang (2020) further developed the algorithm in context of ARMA processes with order p and q to obtain forecasting interval for the future time point of $n+1$, where the order

p and q are assumed to be known. Schulz et al. (2021) expanded the algorithm of Lu and Wang (2020) from $n+1$ future time points to $n+k$ future time points with $k = 1, 2, \ldots, H$. Moreover, the ARMA models are fitted only with maximum likelihood estimation (MLE) for simplicity. For an observed series ξ_t with totally n observations, the forecasting intervals can be calculated with following iterative procedures with $s = 1, 2, \ldots, M$ being the number of iterations and M being the maximal number of iterations following Schulz et al. (2021):

1. Estimate the coefficients ϕ_i and θ_j of the underlying ARMA process and calculate the point forecasts $\hat{\xi}_{n+1}, \hat{\xi}_{n+2}, \ldots, \hat{\xi}_{n+k}$. Thereafter, calculate the residuals $\hat{u}_t$ and define $\hat{F}_u$ as the empirical distribution of the residual series.

2. Start an iteration counter $s = 1$. Draw a sample set $u^*_{t,s}$ with n observations from $\hat{F}_u$ to form a new series $\xi^*_{1,s}, \xi^*_{2,s}, \ldots, \xi^*_{n,s}$ using sampled residual $u^*_{t,s}$ and estimated coefficients $\hat{\phi}_i$ and $\hat{\theta}_j$ obtained in step 1.

3. Estimate the coefficients $\hat{\phi}^*_{i,s}$ and $\hat{\theta}^*_{j,s}$ of the new simulated series in step 2 and calculate a new residual series $\hat{u}^{**}_{t,s}$ from the original process ξ_t based on these coefficients.

4. Obtain the point forecasts $\hat{\xi}^*_{n+1,s}, \hat{\xi}^*_{n+2,s}, \ldots, \hat{\xi}^*_{n+k,s}$ using Formula 8 based on ξ_t, $\hat{u}^{**}_{t,s}$, $\hat{\phi}^*_{i,s}$ and $\hat{\theta}^*_{j,s}$.

5. Calculate the future bootstrap observations $\xi^*_{n+k,s}$ based on $\xi_t, \hat{u}_t, \hat{\phi}_i, \hat{\theta}_j$ and H bootstrapped future innovations $u^*_{t,s}$.

6. Calculate the forecasting error by $\xi^*_{n+k,s} - \hat{\xi}^*_{n+k,s}$ for each $k = 1, 2, \ldots, H$.

7. Increase s by 1 and repeat step 2 to 6 M times and obtain the empirical distribution of $\xi^*_{n+k,s} - \hat{\xi}^*_{n+k,s}$ for each k with the respective $(\alpha^*/2)100\%$-quantiles, denoted as $q_k(\alpha^*/2)$.

The forecasting intervals can then be described as

$$[\hat{\xi}_{n+k} + q_k(\alpha^*/2), \hat{\xi}_{n+k} - q_k(\alpha^*/2)] \tag{17}$$

Under omission of the error in $\hat{m}(x_{n+k})$, the final forecasting intervals for the semiparametric ARMA model can be given by

$$[\hat{y}_{n+k} + q_k(\alpha^*/2), \hat{y}_{n+k} - q_k(\alpha^*/2)] \tag{18}$$

4.4 Evaluation of forecasting accuracy

To ensure the accuracy and reliability of time series forecasts, it's essential to conduct backtesting to the forecasts. Tashman (2000) summarized previous researches and suggested to use out-of-sample test comparing to in-sample test. Since it's practically not convenient to wait for results of forecast periods, practitioners generally split the sample time series, say y_t with $t = 1, 2, 3, \ldots, n$ into two sub-datasets. The first sub-dataset, denoted as *traindataset*, with n_{tr} observations will be used to fit the model and generate point forecasts $\hat{y}_{n_{tr}+1}, \hat{y}_{n_{tr}+2}, \ldots, \hat{y}_{n_{tr}+n_{te}}$, while the second sub-dataset, denoted as *testdataset*, with $n_{te} = n - n_{tr}$ will be used to backtest the quality of the point forecasts. A crucial consideration is the optimal partitioning of the sample dataset into training and test datasets. While tests can be conducted either once using a fixed training set size n_{tr} or repeatedly using a rolling window of size n_{tr}, fixed-size training sets offer an efficient and practical approach for assessing out-of-sample forecast accuracy, despite potentially mixing near-term and far-term forecast errors (Tashman, 2000). Since forecast errors are calculated as the difference between generated point forecasts and observations in the test set, the number of observations in the test set n_{te} must be at least equal to the number of point forecasts. To evaluate forecast performance, a confidence level α is typically specified. Forecasts are deemed acceptable if at least $n_{te} * \alpha$ observations in *testdataset* fall within the corresponding confidence intervals. For commonly used confidence levels of 0.95 or 0.99 with $n_{te} = 5$, a minimum of 4.75 or 4.95 point forecasts, respectively (rounded up to 5), must fall within the confidence intervals to be considered acceptable from backtesting perspective.

There are already many researches on the choice of forecast-error statistics. Tashman (2000) proposed to prefer **scale-independent and percentage** error measures for averaging forecast-errors. Hyndman and Koehler (2006) proposed the mean absolute scaled error (MASE) as a scale-independent forecast-error statistics, which is adopted by Schulz et al. (2021) in *smoots* package as one indicator to measure quality of forecasts. Mean absolute scaled error (MASE) is a scale-independent metric that compares model performance to a naive benchmark, as define in formula below:

$$MASE = \frac{\frac{1}{n_{te}} \sum_{k=1}^{n_{te}} |y_{n_{tr}+k} - \hat{y}_{n_{tr}+k}|}{\frac{1}{n_{tr}-1} \sum_{t=2}^{n_{tr}} |y_t - y_{t-1}|} \tag{19}$$

Obvisouly, MASE is the ratio of mean absolute error (MAE) of the point forecasts and mean absolute error of a benchmarking naive forecast applied to *traindataset*, meaning that values less than 1 indicate better performance than a naive forecast. Furthermore, the closer MASE

is approaching to zero, the better is the performance. Due to scale-independence, MASE based on different forecasting methods can be compared directly. For the same time series, MASE is likely to grow with n_{te}, since the point forecasts are generated based on one step forward approach and errors of forecasts are theoretically getting larger for more distant future time point (Schnaars, 1986).

The mean absolute percentage error (MAPE) is another widely used measures of forecast accuracy, due to its advantages of scale-independency and interpretability (Fildes et al., 2022), which can be defined by

$$MAPE = \frac{100}{n_{te}} \sum_{k=1}^{n_{te}} \left| \frac{y_{n_{tr}+k} - \hat{y}_{n_{tr}+k}}{y_{n_{tr}+k}} \right| \tag{20}$$

MAPE takes the arithmetic average of percentage errors for all point forecasts, can be simply interpreted as percentage of information not covered by the forecasts. A MAPE closer to zero should be preferred. Although previous researches point out several limitations of MAPE like sensitivity to outliers and meaningless for zero values, it is useful as an intuitive indicator in the practice and widely adopted in enterprise softwares like SAC (SAP SE, 2024a).

5 A forecast application *smootsForecast* in SAP

As one of the world leading software providers, SAP Group offers software solutions for managing business processes. Products of SAP include classic Enterprise Resource Planning (ERP) solutions, cloud platforms, database solutions, data warehousing and data analytic software (SAP SE, 2024b). As data driven decision making is gaining more and more importance, getting reliable forecasts is becoming a huge competitiveness for enterprises, among which time series forecasting plays an important role due to the fact that many data sources like sales and costs have characteristics of time series. To meet this rising demand, SAP provides a built-in Predictive Analytics Library (PAL) in its database product SAP HANA, which makes some traditional time series models including ARIMA and exponential smoothing models easily consumable (SAP SE, 2024a). Nevertheless, the integration of further models e.g. the semiparametric ARMA model for time series analysis and forecast can be beneficial in some use cases, especially considering the full data-driven property. In Section 5.1, the R *smoots* package implementing the semiparametric ARMA model and some relevant functions are reviewed. Thereafter in Section 5.2, the architecture of a forecast application *smootsForecast* using algorithms in *smoots* is described detailedly according to a TOGAF

framework.

5.1 The *smoots* package with semiparametric ARMA model

The in Section 3 and 4 described semiparametric ARMA model and forecasting techniques were implemented as the R package *smoots*, standing for "smoothing time series", which provides direct applicable functions for estimating nonparametric trend and its derivatives in equidistant time series (TS) with short-memory stationary errors. The estimation is conducted via local polynomial regression using an automatically selected bandwidth obtained by a built-in iterative plug-in (IPI) algorithm or a bandwidth fixed by the user. The version 1.0.0 of *smoots* package was firstly published by Feng et al. in 2019. The package was extended with further functions to the current version of version 1.1.4 (Feng et al., 2023; Schulz et al., 2021).

For estimating nonparametric trend with automatically identified optimal bandwidth, the *msmooth* or *tsmooth* function can be used. Both functions include a series of input arguments for e.g. order of polynomial p, kernel weighting functions mu, method for estimating variance factor c_f, percentage of boundary omission cb for automatic bandwidth selection etc, which makes it possible to cover estimation of very specific times series. For generating point forecasts, the function $modelCast$ can be used with specific parameters like forecast method $method$, number of point forecast h, confidence level α and so on. To evaluate the quality of forecasts, the package provides the function $rollCast$, which outputs, most importantly, the number of observations in $testdataset$ lying outside the given confidence intervals as well as the forecast-error statistic MASE (Feng et al., 2023; Schulz et al., 2021).

5.2 Architecture and design of a forecast application $smootsForecast$

Since SAP has a very strong focus on enterprise context, we propose to apply the TOGAF framework for designing architecture of the forecast application. According to TOGAF framework, an application need be specified in four interrelated areas, socalled architecture domains. Business architecture defines the business strategy, governance, organization, and business processes. Data architecture describes the structure of an organization's logical and physical data assets and the associated data management resources. Application architecture provides a blueprint for the individual application to be deployed and functioned within the core business processes. Lastly, technology architecture describes the hardware, software,

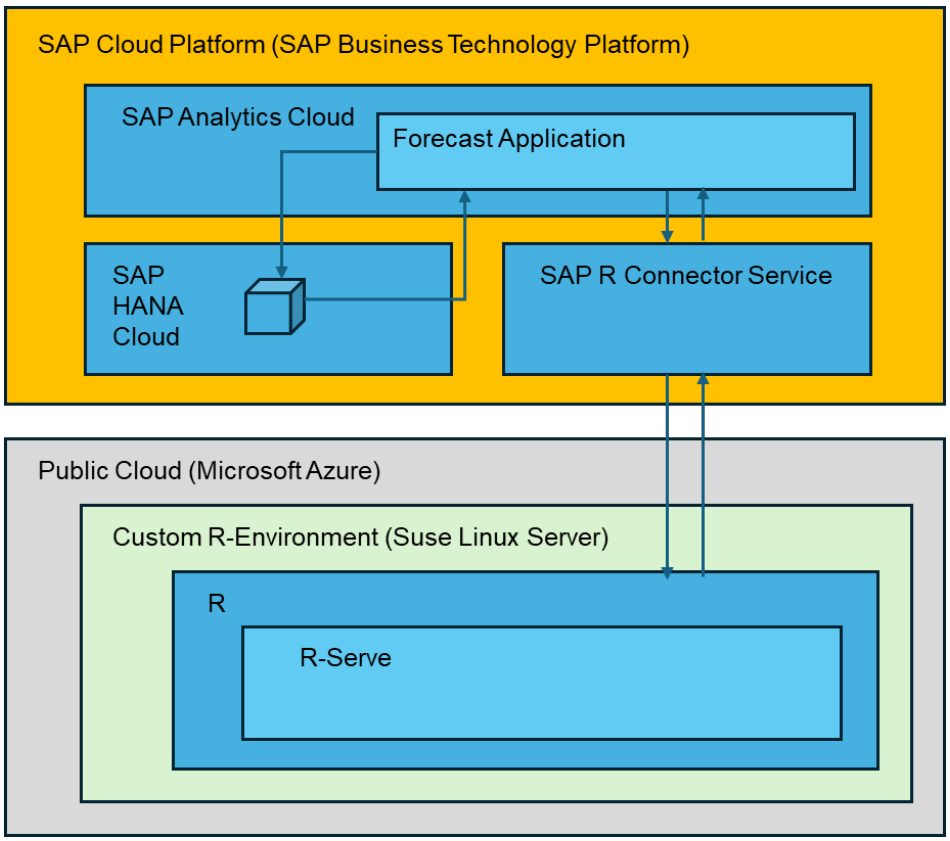

Figure 1: Architecture of forecast application

and network infrastructure needed to support the deployment and function of the application (The Open Group, 2022).

From a business perspective, a forecasting application within the SAP ecosystem should exhibit seamless interoperability with other enterprise processes. In the context of integrated financial planning, the application can be leveraged to predict future sales based on historical sales data, thereby serving as a crucial input for revenue planning. From a data perspective, the application must be capable of storing time series data and forecasting results, necessitating a robust database connection. To facilitate the computationally intensive forecasting tasks, the *smoots* package in R will be utilized as the forecasting engine, which requires secure data communication between the R environment and the SAP system. Several SAP products, such as SAP HANA database and SAP Analytics Cloud (SAC), offer standardized mechanisms for connecting to R environments. SAC even offers a standard widget $RVisualization$ to expand its visualization capabilities. Furthermore, SAC provides a user-friendly low-code development environment for constructing the application's user interface (UI). Considering these capabilities, we propose an overall technology architecture as illustrated in Figure 1 for the forecast application $smootsForecast$.

5.2.1 Integration of custom R-Environment with the forecast application

For the forecast application *smootsForecast*, we hosted a R environment with a R-Serve on a Suse linux virtual machine based on Microsoft Azure platform. R of version 4.4 is installed on the virtual machine. The R packages *Rserve* and *smoots* are installed. With architecture described above, execution of R-script can be triggered from the forecast application in SAC. To be specific, a *RVisualization* widget is embedded into the forecast application, which collects the input parameters, data sources and a custom R-script (see Appendix A). Via calling functions from *smoots* package in R-script, the forecast application is able to train nonparametric models and generate point forecasts. The R-script, along with the input parameters and data frames, are sent from SAC to the SAP R Connector Service. This service is responsible for forwarding the request to the appropriate R environment. The SAP R Connector Service communicates with the R-Serve using the R-Serve TCP/IP protocol. This protocol allows for secure and efficient transmission of data and execution commands. The connection to R-Serve is encrypted using a SSL-certificate. The R-Serve in the custom R environment receives the request and runs the R-script. Once the computations are complete, the results are sent back through the R-Serve to the SAP R Connector Service. The SAP R Connector Service then relays the results back to the forecast application in SAC.

5.2.2 Storage of time series data and forecast results

To store time series data, a data model *TSModel* of type planning is created in SAC. Time series data can be imported and stored in this data model, under which a star schema with master data tables and fact tables is created in SAP HANA Cloud. The data model is designed to contain actual values, forecast values, confidence intervals as measures. In addition to the time dimension, a dimension *TSAccount* is created to store multiple time series in the data model. Data sources based on *TSModel* are connected to the forecast application *smootsForecast* and embedded into R-script together with other input parameters. After completion of execution on R-Serve, the results are returned to *smootsForecast*, where they can be evaluated, visualized and written back to data model for usage in further business processes. Since time series data can be stored within the same database as business application data, optimal cost efficiency in terms of data storage, ETL processes, and data security can be achieved.

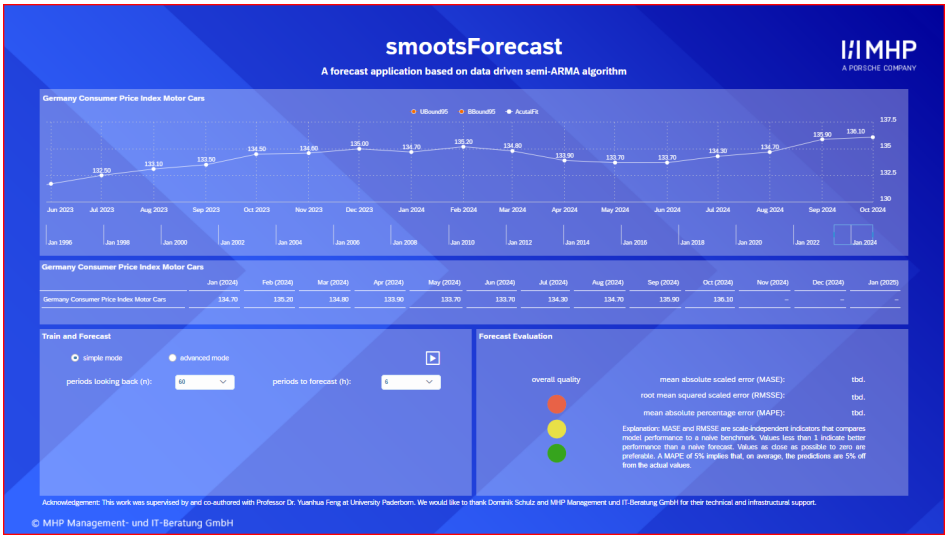

Figure 2: User interface(UI) of *smootsForecast* simple mode

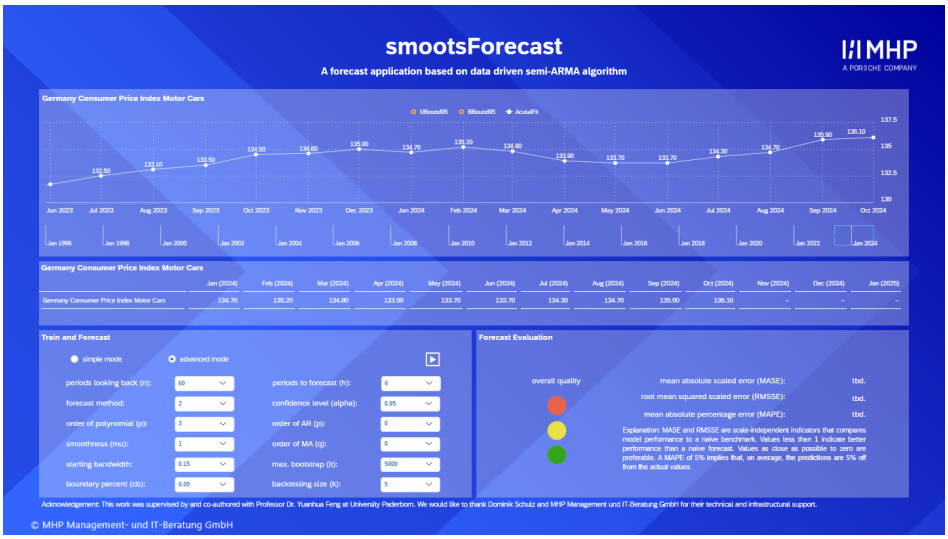

Figure 3: User interface (UI) of *smootsForecast* advanced mode

5.2.3 User Interface (UI) and function design of *smootsForecast*

The forecast application *smootsForecast* is implemented within SAC, benefiting on the one hand from the low code technical convenience, on the other hand from the high integration with other components on SAP Business Technology Platform (BTP). As illustrated by Figure 2 and Figure 3, the UI design is overall characterized by a professional color scheme and arranged into four primary sections, each serving a distinct purpose. The uppermost section of the interface visualize a time-series and optionally with point forecasts and confidence intervals as a line chart. Directly below the line chart, a table displays actual and forecasted data points together, showcasing a popular rolling forecast use case in enterprise context.

The "Train and Forecast" panel allows users to configure and customize the forecasting model, offering two operational modes:

- Simple Mode (Figure 2) provides a streamlined setting for simple use cases and users who have only limited understanding on data science and the semiparametric ARMA algorithm behind the application. In this mode, only number of historical periods and desired forecast horizon need to be specified by the user. All other parameters are either fixed or left to be default in background. Note that the parameter k for backtesting periods are fixed to 6 in this case, which results in highest possible periods for forecast horizon of 6 in simple mode.

- Advanced Mode (Figure 3) is designed for expert users, which enables fine-tuning of model parameters. In this mode, expert users are able to input ten more parameters to semiparametric ARMA algorithm for more flexibility and customization.

An overview of all input parameters are listed in Table 1. Moreover, a play button is provided to initiate the forecasting computation.

The last panel on bottom right provide users with quantitative and qualitative information about statistical quality of the generated forecasts. Quantitatively, MASE, RMSSE and MAPE of selected model are calculated and displayed. An explanation to both statistics is added as note for the users to better interpret these indicators. For even better user experience, an overall quality is represented visually using a traffic light system, providing an intuitive indicator of forecast reliability. The traffic light system is designed to be green, if the out-of-sample test described in Section 4.4 accepts the model, meaning no observations in *testdataset* lying outside the confidence interval of point forecasts based on *traindataset*. The traffic light becomes yellow, if the absolute sum of breaches of point forecasts is smaller than maximal $n_{te} \times (1 - \alpha)$. In this case, users should evaluate the results and fine-tune the model by adjusting the input parameters. If the absolute sum of breaches of point forecasts is bigger than $n_{te} \times (1 - \alpha)$, the model should be rejected. A code Snippet for implementing this traffic light logic in *JavaScript* in SAC can be found as Appendix B. In case that a model can be accepted (traffic light in green or yellow), users can write back point forecasts and confidence interval for each time point in forecast horizon back to $TSModel$ for further integration into other business processes. The execution of write back can be done by clicking on the icon of data model in the panel.

Input parameter in *smootsForecast*	Influence on semiparametric ARMA model	Available in mode
periods looking back (n)	defines number of data points used to for training	simple mode
periods to forecast (h)	customize a number of point forecasts to be generated	simple mode
forecast methode ($method$)	assumes a normal distribution or not	advanced mode
order of polynomial (p)	customize wether to use linear or cubic pilynomial	advanced mode
smoothness (mu)	customize the selected kernel function in local polynomial regression	advanced mode
starting bandwidth ($startB$)	customize a starting bandwidth for the plug-in algorithm for selecting optimal bandwidth	advanced mode
confidence level (α)	customize a confidence level 0.95 or 0.99 for back testing point forecast	advanced mode
order of AR (p)	customize an AR order for the underlying ARMA model	advanced mode
order of MA (q)	customize an MA order for the underlying ARMA model	advanced mode
max. bootstrap (it)	customize a maximal number of iterations for boostraping point forecast	advanced mode
backtesting size (k)	customize a size of out-of-sample test	advanced mode

Table 1: Input parameters in simple and advanced mode

5.2.4 Notes on application development

Overall, the *smootsForecast* is implemented technically as an analytical application in SAC. Different standard widgets of SAC like a time series chart, a table, text labels and input controls are employed to visualize data and application. The time series data is stored via a planning model in SAC and writing back point forecasts involves running a data action to book data in the planning data model. A hidden *RVisualization* widget is used as a R engine executing R-script on a customized R server. Interactions like trigger for starting training or write back are realized via *JavaScript* provided by SAC.

Due to the fact that *smootsForecast* is based on the standard product SAC instead of a fully individual development, its implementation does profit from standard features or components of SAC on the one hand. On the other hand however, this leads to technical limitations to standard technical possibilities within SAC. It is important to note that at least a planning standard license of SAC is needed at least, since data actions are used in the architecture. Moreover, this application is only made to generate point forecasts for a single time series data each run, because SAP doesn't provide an API in *JavaScript* for reading a data frame, a vector or a json object with multiple single values from an R environment. If a panel data with multiple time series, for example a time series for each product, is involved, this application can not generate point forecasts for all products at once. In the practice, this might be a big limitation, since enterprises often have panel data.

6 Application to practical examples

This section demonstrates the application of the semiparametric model within *smootsForecast* application using selected datasets. Initially, point forecasts alongside associated confidence intervals are visualized and interpreted within *smootsForecast* for both simple and advanced mode. Subsequently, a comprehensive evaluation of forecast accuracy is undertaken through a comprehensive analysis of model performance across various parameter combinations.

6.1 Data selection

As for data, we select the monthly harmonized index of consumer prices for motor cars in Germany (GCPIC) from January 1996 to October 2024 including totally 346 data points. According to Eurostat (2025), the harmonized index of consumer prices category for motor

cars is a price index of durable goods that includes both new and second-hand motor cars, passenger vans, station wagons, estate cars, and the like with either two-wheel drive or four-wheel drive. The GCPIC dataset is not seasonally adjusted. The selected dataset is imported into $TSModel$ in SAC. In R-script the logarithm of the data points are calculated and further used for model fitting and forecasting.

6.2 Application in *smootsForecast*

After loading data, data points until October 2024 are displayed in *smootsForecast* as shown in Figure 2 and Figure 3. The indicator for overall quality and MASE, RMSSE and MAPE are set to be defined initially. Due to the fact that users are able to select looking back periods n for training, not all 346 data points are used for training. For a short memory use case, we select $n = 60$ and $h = 6$ in a simple mode in *smootsForecast* and start training and forecast by clicking on the play icon. As shown in Figure 4, after execution an overall quality of green traffic light is obtained, meaning that 0 of 6 data points in *traindataset* are lying outside of 95%-confidence interval during backtesting. This indicates a very good quality of the model. Furthermore, both MASE and RMSSE are smaller than 1, which indicates a better performance comparing to a naive forecast. The statistic MAPE incidates that point forecasts are 9.27% off from the actual value on average. While this error may not represent the absolute minimum, industry guidelines generally consider forecasts with average errors below 20% to be acceptable. Consequently, the overall forecast accuracy can be deemed satisfactory. After taking over the results by clicking on the cube icon in forecast evaluation panel, the point forecasts along with their forecast intervals at 95% confidence level (red lines on time series chart in Figure 4) are written into database table of $TSModel$ and displayed on the time series chart and table in *smootsForecast*. As can be seen, German consumer price index for motor cars is predicted to slightly decrease in November and December 2024 before it goes up in next 6 months. Looking at the statistics MASE and RMSSE obtained in simple mode with $n = 60$ and $h = 6$, one may argue that the results are just a bit better than a naive forecast, since $MASE = 0.87881435$ and $RMSSE = 0.74949270$ are still quite high. In this case, it could make sense to adjust some more parameters to optimize the model and forecasts, where the advanced mode comes into play. For a further training and forecast in advanced mode, we selected $n = 120$, $h = 3$ and $k = 3$ and obtained results as displayed in Figure 5. First of all, the traffic light yellow suggests that one point forecasts are at the edge of confidence interval during backtesting. Nevertheless, the MASE, RMSSE and MAPE

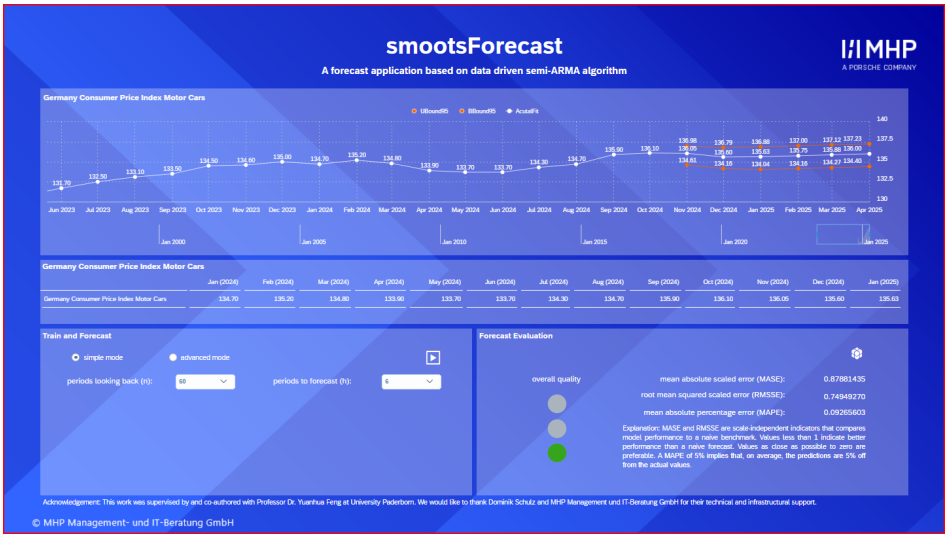

Figure 4: simple mode with $n = 60, h = 6$

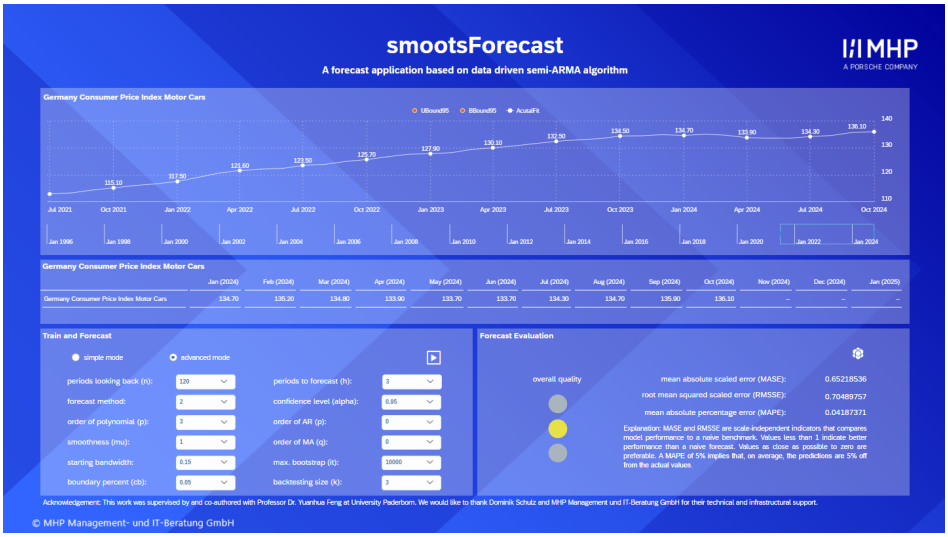

Figure 5: advanced mode with $n = 120, h = 3, k = 3$

all show an improvement in model quality in comparison with those with simple mode with $n = 60$ and $h = 6$. With regard to point forecasts, the results slightly go down in first two periods, which align almost with the results with the simple mode. For the third following period, the prediction goes however further down, while with a shorter training horizon the third point forecast turns slightly to an upper direction.

	$k=1$			$k=2$			$k=3$		
n	sum of breaches	MASE	MAPE	sum of breachess	MASE	MAPE	sum of breaches	MASE	MAPE
36	0	0.35	0.04	0.01	1.91	0.21	0.02	2.17	0.24
72	0	0.21	0.02	0.00	1.41	0.13	0.00	0.96	0.09
108	0	0.05	0.00	0.00	1.55	0.11	0.00	0.94	0.07
144	0	0.37	0.22	0.00	0.97	0.06	0	0.80	0.05
180	0	0.77	0.04	0.00	1.22	0.06	0	1.08	0.05
216	0	1.69	0.08	0.00	1.83	0.09	0	1.44	0.07
252	0	1.82	0.08	0.00	1.95	0.08	error 1	error 1	error 1
288	0	1.94	0.08	0	2.01	0.08	0	1.64	0.07
324	0	2.03	0.08	0	2.04	0.08	0	1.70	0.07

Table 2: Performance metrics for different n and k

	$k=4$			$k=5$			$k=6$		
n	sum of breaches	MASE	MAPE	sum of breachess	MASE	MAPE	sum of breaches	MASE	MAPE
36	0.06	3.25	0.36	error 1	error 1	error 1	0.03	1.37	0.16
72	0.00	0.93	0.08	0	0.56	0.05	0	1.10	0.10
108	0	0.64	0.04	0	0.96	0.07	-0.00	1.92	0.14
144	0	0.80	0.05	-0.00	1.32	0.08	-0.00	2.45	0.14
180	0	1.06	0.05	-0.00	1.64	0.08	-0.01	2.83	0.14
216	0	1.21	0.06	error 2	error 2	error 2	-0.00	1.76	0.08
252	0	1.28	0.06	0	1.52	0.07	error 1	error 1	error 1
288	0	1.33	0.06	-0.00	1.59	0.07	-0.00	2.21	0.09
324	0	1.34	0.05	-0.00	1.73	0.07	-0.00	2.30	0.09

Table 3: Performance metrics for different n and k

6.3 Comprehensive evaluation of forecast accuracy with different parameters

Since forecast accuracy is of crucial importance, further analyses were conducted with selected dataset using different combinations of n and k, where k is the out-of-sample size for backtesting as well as the highest forecast horizon $max(h)$. Table 2 and Table 3 show results for k from 1 to 6 and n being number of the latest 36, 72, 108, 144, 180, 216, 252, 288 and 324 months. Sum of breaches represents the total number of out-of-sample data points lying out side of the confidence interval of point forecasts. Since MASE and RMSSE are per definition highly corelated. MASE and MAPE measures rounded to two decimal places are taken into

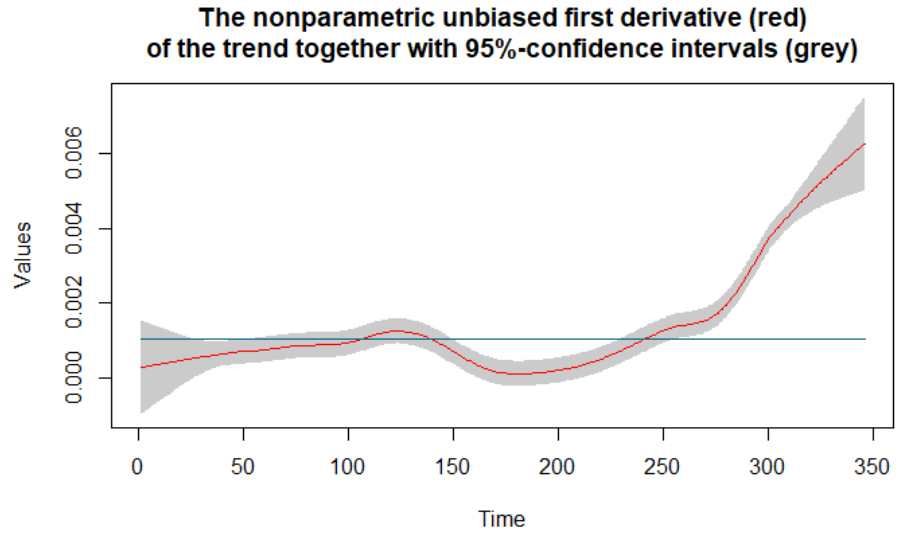

Figure 6: first derivative of trend

consideration. We marked the sum of breaches green if it equals 0. Some cases with sum of breaches very close to zero are marked yellow. All MASE values which are smaller than 1 are marked as green, indicating a better performance than a naive forecast. As for MAPE, we marked values smaller than 10% as green for excellent, between 10% and 20% as yellow for acceptable.

Based on the color map on both tables, it's very clear that the sum of breaches criteria for most scenarios are in green or yellow, thus acceptable. Some scenarios run into error, which is ignored in this paper. Looking at the statistic MAPE, the performance are in most cases excellent (in green, smaller than 10%) and a few cases acceptable with $k = 6$. As for the MASE, the model shows good performance for $k = 1, 3, 4, 5$ with a training dataset including the last 72, 108, 144 observations, while the performance with regard to MASE is generally bad for $k = 6$ and for large number of observations from 180 upwards. With the increase of k, the performance of forecasts tends to get worse, which can be well understood intuitively, since uncertainty grows with the forecast horizon. Moreover, for a small number of observations for training like $n = 36$, scenarios with k bigger than 1 are showing bad performance or even error, which may suggest that enough observations for training the model is a key to obtain forecasts of good quality.

Furthermore, it's very interesting to observe that the MASE indicator is generally bad for large number of observations, which leads us to investigate the nonparametric unbiased first derivative of the trend. Since *smootsForecast* does not include the graphical test tool for

linearity and stationarity, the first derivative is obtained with 95% confidence level in local R with same dataset separately as shown in Figure 6. Obviously, a very strong change in first derivative of the trend (the changing rate) is observed at about 180 months before the last observation. Due to the strong change in trend and its derivative, the forecast quality measure MASE might be influenced negatively, if data points with totally different trend direction from more than 180 months ago are used for backtesting the forecast accuracy for latest observations.

7 Concluding Remarks

In this paper, the semiparametric ARMA model proposed by Feng et al. (2020a) along with its theoretical background were reviewed. Forecasting techniques based on the semiparametric ARMA model under normal or unknown errors following idea of Schulz et al. (2021) were discussed. Given the fact that statistical forecasts can be well adopted as baseline forecasts in enterprise planning context and nonparametric or semiparametric models are proven to be better performed as traditional parametric models (Fildes et al., 2022; Aburto and Weber, 2007; Kuo, 2001; Žliobaitė et al., 2012), this paper proposes to adopt the semiparametric ARMA model for enterprise use case. However, most enterprises actually face challenges in cost efficiency, missing statistical or machine learning understanding and user experience during adoption of custom algorithms (Goodwin et al., 2023). Therefore, we showcased an approach to integrate the semiparametric ARMA algorithm in *smoots* package into one of the most popular enterprise software SAC with a forecast application $smootsForecast$. The application utilizes some standardized features of SAC in combination with *smoots* package, which realized optimal cost efficiency and user experience. By utilizing intuitive visualization like the traffic light system for overall quality evaluation and extended statistic MAPE, the interpretation of application results is improved. To address the problem of potentially missing knowledge in statistics and machine learning, we propose to offer a simple mode and an advanced mode based on knowledge requirements. Finally, the application was tested with a practical example for forecasting German consumer price index of motor cars. Overall, the forecasting quality was positively confirmed. Some remarks were made about how to optimal configure the application for better performance.

Nevertheless, some technical limitations inherited from SAC features do exist in the forecast application, especially for handling panel data, which is often the case in enterprise

context. Moreover, the semiparametric model adopts several assumptions like stationarity and short term error, which can not be checked with the forecast application currently. This may lead to inappropriate results as shown in Section 6.3 during application. From perspective of algorithm diversity, it might be valuable to extend the forecast application with other custom algorithms with seasonality and long memory characteristic.

Acknowledgments

This work was supervised by and co-authored with Professor Dr. Yuanhua Feng at University Paderborn. I am very grateful for his valuable guidance and support throughout this research. I also would like to thank Dominik Schulz at University Paderborn and MHP Management- und IT-Beratung GmbH for their technical and infrastructural support. Furthermore, I thank my family and friends in Paderborn for their mental and physical support during my research.
This work was supervised by and co-authored with Professor Dr. Yuanhua Feng at University Paderborn. I am very grateful for his valuable guidance and support throughout this research. I also would like to thank Dominik Schulz at University Paderborn and MHP Management- und IT-Beratung GmbH for their technical and infrastructural support. Furthermore, I thank my family and friends in Paderborn for their mental and physical support during my research.

References

Luis Aburto and Richard Weber. Improved supply chain management based on hybrid demand forecasts. *Applied Soft Computing*, 7(1):136–144, 2007.

J. Beran and Y. Feng. Local polynomial fitting with long-memory, short-memory and antipersistent errors. *Annals of the Institute of Statistical Mathematics*, 54:291–311, 2002a.

J. Beran and Y. Feng. Semifar models - a semiparametric approach to modelling trends, long-range dependence and nonstationarity. *Computational Statistics & Data Analysis*, 40:393–419, 2002b.

P. J. Brockwell and R. A. Davis. *Introduction to Time Series and Forecasting*. Springer International Publishing AG, Basel, Switzerland, 3rd edition, 2016.

P. Bühlmann. Locally adaptive lag-window spectral estimation. *Journal of Time Series Analysis*, 17(3):247–270, 1996.

Chris Chatfield. *The Analysis of Time Series - An Introduction*. Chapman and Hall/CRC, 2003.

W. S. Cleveland and C. Loader. Smoothing by local regression: Principles and methods. In *Statistical Theory and Computational Aspects of Smoothing*, pages 10–49. Physica-Verlag HD, 1996.

Gérard Collomb. Nonparametric regression: An up-to-date bibliography. *Statistics: A Journal of Theoretical and Applied Statistics*, 16(2):309–324, 1985.

Eurostat. Harmonized index of consumer prices: Motor cars for germany (including former gdr from 1991) [cp0711dem086nest]. https://fred.stlouisfed.org/series/CP0711DEM086NEST, 2025. retrieved from FRED.

J. Fan. Local linear regression smoothers and their minimax efficiencies. *The Annals of Statistics*, pages 196–216, 1993.

J. Fan and I. Gijbels. Variable bandwidth and local linear regression smoothers. *The Annals of Statistics*, pages 2008–2036, 1992.

J. Fan and I. Gijbels. *Local Polynomial Modelling and Its Applications: Monographs on Statistics and Applied Probability 66*. Chapman and Hall, London, England, 1st edition, 1996.

Y. Feng. *Non- and Semiparametric Regression With Fractional Time Series Errors: Theory and Applications to Financial Data*. PhD thesis, University of Konstanz, Germany, 2004.

Y. Feng and S. Heiler. A simple bootstrap bandwidth selector for local polynomial fitting. *Journal of Statistical Computation and Simulation*, 79(12):1425–1439, 2009.

Y. Feng and C. Zhou. Forecasting financial market activity using a semiparametric fractionally integrated log-acd. *International Journal of Forecasting*, 31(2):349–363, 2015.

Y. Feng, T. Gries, and M. Fritz. Data-driven local polynomial for the trend and its derivatives in economic time series. *Journal of Nonparametric Statistics*, 32(2):510–533, 2020a.

Y. Feng, T. Gries, S. Letmathe, and D. Schulz. The smoots package in r for semiparametric modeling of trend stationary time series. Preprint, Paderborn University, 2020b.

Y. Feng, S. Letmathe, D. Schulz, T. Gries, and Marlon Fritz. smoots: Nonparametric estimation of the trend and its derivatives in ts, 2023. URL https://CRAN.R-project.org/package=smoots. R package version 1.1.4.

Robert Fildes, Paul Goodwin, Michael Lawrence, and Konstantinos Nikolopoulos. Effective forecasting and judgmental adjustments: an empirical evaluation and strategies for improvement in supply-chain planning. *International Journal of Forecasting*, 25(1):3–23, 2009.

Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022.

M. Fritz, S. Forstinger, Y. Feng, and T. Gries. Forecasting economic growth processes for developing economies. Preprint, Paderborn University, 2018.

T. Gasser and H.G. Müller. Kernel estimation of regression functions. In T. Gasser and M. Rosenblatt, editors, *Smoothing Techniques for Curve Estimation*, pages 23–68, Heidelberg, 1979. Springer-Verlag.

T. Gasser, C. Jennen-Steinmetz, and J. Engel. Comment on c.k. chu and j.s. marron (1991). *Statistical Science*, 6:419–421, 1991.

P. Goodwin, J. Hoover, S. Makridakis, F. Petropoulos, and L. Tashman. Business forecasting methods: Impressive advances, lagging implementation. *PLoS One*, 18:e0295693, 12 2023.

T. Hastie and C. Loader. Local regression: Automatic kernel carpentry. *Statistical Science*, 8(2):120–129, 1993.

E. Herrmann and T. Gasser. Iterative plug-in algorithm for bandwidth selection in kernel regression estimation. Discussion Paper, 1994.

R. J. Hyndman and A. B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679–688, 2006.

R.J. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice*. OTexts.com, Melbourne, Australia, 2nd edition, 2018.

RJ Kuo. A sales forecasting system based on fuzzy neural network with initial weights generated by genetic algorithm. *European Journal of Operational Research*, 129(3):496–517, 2001.

Avijit Dhar Laboni Bhowmik and Ranajay Mukherjee. *Machine Learning with SAP - Models and Applications*. SAP PRESS, 2021.

X. Lu and L. Wang. Bootstrap prediction interval for arma models with unknown orders. *REVSTAT-Statistical Journal*, 18(3):375–396, 2020.

Teresa M. McCarthy, Donna F. Davis, Susan L. Golicic, and John T. Mentzer. The evolution of sales forecasting management: a 20-year longitudinal study of forecasting practices. *Journal of Forecasting*, 25(5):303–324, 2006.

H.-G. Müller. Weighted local regression and kernel methods for nonparametric curve fitting. *Journal of the American Statistical Association*, 82(397):231–238, 1987. doi: https://doi.org/10.2307/2289159.

Hans-Georg Müller. *Nonparametric Regression Analysis of Longitudinal Data*. Springer, New York, 1988. ISBN 978-1-4612-3926-0.

E. A. Nadaraya. On estimating regression. *Theory of Probab. Appl.*, 9:141–142, 1964.

L. Pan and D. N. Politis. Bootstrap prediction intervals for linear, nonlinear and nonparametric autoregressions. *Journal of Statistical Planning and Inference*, 177:1–27, 2016.

L. Pascual, J. Romo, and E. Ruiz. Bootstrap predictive inference for arima processes. *Journal of Time Series Analysis*, 25(4):449–465, 2004.

D. Ruppert and M. P. Wand. Multivariate locally weighted least squares regression. *The Annals of Statistics*, pages 1346–1370, 1994.

SAP SE. Sap hana database predictive analysis library (pal), 2024a.

SAP SE. What is sap? https://www.sap.com/germany/about/what-is-sap.html, 2024b.

Steven P. Schnaars. A comparison of extrapolation models on yearly sales forecasts. *International Journal of Forecasting*, 2:71–85, 1986. URL https://api.semanticscholar.org/CorpusID:154164456.

Dominik Schulz, Yuanhua Feng, Thomas Gries, Marlon Fritz, and Sebastian Letmathe. Diagnosing the trend and bootstrapping the forecasting intervals using a semiparametric arma. Technical report, Paderborn University, 2021.

A. A. Syntetos, J. E. Boylan, and S. M. Disney. Forecasting for inventory planning: a 50-year review. *Journal of the Operational Research Society*, 60(sup1):S149–S160, 2009.

Leonard J. Tashman. Out-of-sample tests of forecasting accuracy: an analysis and review. *International Journal of Forecasting*, 16(4):437–450, 2000. ISSN 0169-2070. The M3-Competition.

The Open Group. *The TOGAF® Standard, 10th Edition - Introduction and Core Concepts*. van Haren Publishing, 2022. URL https://pubs.opengroup.org/togaf-standard/index.html.

G. S. Watson. Smooth regression analysis. *Sankhyā A*, 26:359–372, 1964.

Indrė Žliobaitė, Jorn Bakker, and Mykola Pechenizkiy. Beating the baseline prediction in food sales: How intelligent an intelligent predictor is? *Expert Systems with Applications*, 39(1):806–815, 2012.

A Appendix A: Custom R-script in forecast application

```
library(smoots)

if(startR == 0){
## do nothing on application initialization

}else if (startR==1) {
## run simple mode training

## prepare ts data
time <- TSModel$Date
GHCT <- TSModel$`Germany Consumer Price Index Motor Cars`
log_GHCT <- log(GHCT)
log_GHCT_tr <- tail(log_GHCT,para_n)

## train and generate point forecasts
est_GHCT <- msmooth(log_GHCT_tr)
pre_GHCT <- modelCast(est_GHCT, h = para_h ,method = "boot",
pb=FALSE, plot = FALSE)

## backtesting results
set.seed(1)
backtest_GHCT <- rollCast(log_GHCT_tr, alpha = 0.95, K=6,
method ="boot",  pb=FALSE, plot = FALSE)

## output results
output_overall <- sum(backtest_GHCT$breach.val)
output_mase <- backtest_GHCT$MASE
output_rmsse <- backtest_GHCT$RMSSE

# calculate mape
x<-backtest_GHCT$fcast.roll[1,]
y<-backtest_GHCT$y.out
output_mape <- mean(abs((y - x) / y)) * 100

## set output forecast values
## mean function to transform to number
output_fc1 <- mean(exp(pre_GHCT[1,][1]))
output_fc2 <- mean(exp(pre_GHCT[1,][2]))
output_fc3 <- mean(exp(pre_GHCT[1,][3]))
output_fc4 <- mean(exp(pre_GHCT[1,][4]))
output_fc5 <- mean(exp(pre_GHCT[1,][5]))
output_fc6 <- mean(exp(pre_GHCT[1,][6]))

output_bb1 <- mean(exp(pre_GHCT[2,][1]))
output_bb2 <- mean(exp(pre_GHCT[2,][2]))
output_bb3 <- mean(exp(pre_GHCT[2,][3]))
```

```
output_bb4 <- mean(exp(pre_GHCT[2,][4]))
output_bb5 <- mean(exp(pre_GHCT[2,][5]))
output_bb6 <- mean(exp(pre_GHCT[2,][6]))

output_ub1 <- mean(exp(pre_GHCT[3,][1]))
output_ub2 <- mean(exp(pre_GHCT[3,][2]))
output_ub3 <- mean(exp(pre_GHCT[3,][3]))
output_ub4 <- mean(exp(pre_GHCT[3,][4]))
output_ub5 <- mean(exp(pre_GHCT[3,][5]))
output_ub6 <- mean(exp(pre_GHCT[3,][6]))

## reset startR
startR <- 0

}else if(startR==2){
## run advanced mode training

## prepare ts data
time <- TSModel$Date
GHCT <- TSModel$`Germany Consumer Price Index Motor Cars`
log_GHCT <- log(GHCT)
log_GHCT_tr <- tail(log_GHCT,para_n)

## convert parameters
if(para_method == 1) {
str_method <- "norm"
}else{
str_method <- "boot"
}

## train and generate point forecasts
est_GHCT <- tsmooth(log_GHCT_tr,p=para_poly, mu=mu, bStart=para_startb, cb=cb)

if(p+q ==0){
pre_GHCT <- modelCast(est_GHCT, h = para_h ,
method = str_method, alpha = alpha, it=it, pb=FALSE, plot = FALSE)
}else{
pre_GHCT <- modelCast(est_GHCT, h = para_h,
method = str_method, p=p, q=q, alpha = alpha, it=it, pb=FALSE, plot = FALSE)
}

## backtesting results
set.seed(1)
if(p+q ==0){
backtest_GHCT <- rollCast(log_GHCT_tr, K=k,
method =str_method, alpha = alpha, it=it, pb=FALSE, plot = FALSE)
}else{
backtest_GHCT <- rollCast(log_GHCT_tr, K=k,
method =str_method, p=p, q=q,  alpha = alpha, it=it, pb=FALSE, plot = FALSE)
}
```

```
## set output quality values
output_overall <- sum(backtest_GHCT$breach.val)
output_mase <- backtest_GHCT$MASE
output_rmsse <- backtest_GHCT$RMSSE

# calculate mape
x<-backtest_GHCT$fcast.roll[1,]
y<-backtest_GHCT$y.out
output_mape <- mean(abs((y - x) / y)) * 100


## set output forecast values
## mean function to transform to number

output_fc1 <- mean(exp(pre_GHCT[1,][1]))
output_fc2 <- mean(exp(pre_GHCT[1,][2]))
output_fc3 <- mean(exp(pre_GHCT[1,][3]))
output_fc4 <- mean(exp(pre_GHCT[1,][4]))
output_fc5 <- mean(exp(pre_GHCT[1,][5]))
output_fc6 <- mean(exp(pre_GHCT[1,][6]))

output_bb1 <- mean(exp(pre_GHCT[2,][1]))
output_bb2 <- mean(exp(pre_GHCT[2,][2]))
output_bb3 <- mean(exp(pre_GHCT[2,][3]))
output_bb4 <- mean(exp(pre_GHCT[2,][4]))
output_bb5 <- mean(exp(pre_GHCT[2,][5]))
output_bb6 <- mean(exp(pre_GHCT[2,][6]))

output_ub1 <- mean(exp(pre_GHCT[3,][1]))
output_ub2 <- mean(exp(pre_GHCT[3,][2]))
output_ub3 <- mean(exp(pre_GHCT[3,][3]))
output_ub4 <- mean(exp(pre_GHCT[3,][4]))
output_ub5 <- mean(exp(pre_GHCT[3,][5]))
output_ub6 <- mean(exp(pre_GHCT[3,][6]))

## reset startR
startR <- 0
}
```

B Appendix B: Code sneppit for traffic light logic in JavaScript

```
var Rstatus=R_Engine.getStatus();
if(Rstatus===RVisualizationStatus.Running){
        Timer_1.start(2);
}else if(Rstatus===RVisualizationStatus.Error){
        Timer_1.stop();
        Application.hideBusyIndicator();
        Application.showMessage(ApplicationMessageType.Error,
        "train and forecast run into error,
        please check your parameters and try again.");

        statusFC = false;
        TXT_TAKEOVER.setVisible(false);

}else if(Rstatus===RVisualizationStatus.Success){
        Timer_1.stop();

        statusFC = true;

        fc1=R_Engine.getEnvironmentValues().getNumber("output_fc1");
        fc2=R_Engine.getEnvironmentValues().getNumber("output_fc2");
        fc3=R_Engine.getEnvironmentValues().getNumber("output_fc3");
        fc4=R_Engine.getEnvironmentValues().getNumber("output_fc4");
        fc5=R_Engine.getEnvironmentValues().getNumber("output_fc5");
        fc6=R_Engine.getEnvironmentValues().getNumber("output_fc6");

        bb1=R_Engine.getEnvironmentValues().getNumber("output_bb1");
        bb2=R_Engine.getEnvironmentValues().getNumber("output_bb2");
        bb3=R_Engine.getEnvironmentValues().getNumber("output_bb3");
        bb4=R_Engine.getEnvironmentValues().getNumber("output_bb4");
        bb5=R_Engine.getEnvironmentValues().getNumber("output_bb5");
        bb6=R_Engine.getEnvironmentValues().getNumber("output_bb6");

        ub1=R_Engine.getEnvironmentValues().getNumber("output_ub1");
        ub2=R_Engine.getEnvironmentValues().getNumber("output_ub2");
        ub3=R_Engine.getEnvironmentValues().getNumber("output_ub3");
        ub4=R_Engine.getEnvironmentValues().getNumber("output_ub4");
        ub5=R_Engine.getEnvironmentValues().getNumber("output_ub5");
        ub6=R_Engine.getEnvironmentValues().getNumber("output_ub6");

        overall=R_Engine.getEnvironmentValues().getNumber("output_overall");
        mase=R_Engine.getEnvironmentValues().getNumber("output_mase");
        rmsse=R_Engine.getEnvironmentValues().getNumber("output_rmsse");
        mape=R_Engine.getEnvironmentValues().getNumber("output_mape");

        // display results
        if(overall===0){
```

```
                SH_GREEN.setVisible(true);
                SH_YELLOW.setVisible(false);
                SH_RED.setVisible(false);
                TXT_TAKEOVER.setVisible(true);

        }else if(overall===1){
                SH_GREEN.setVisible(false);
                SH_YELLOW.setVisible(true);
                SH_RED.setVisible(false);
                TXT_TAKEOVER.setVisible(true);

        }else{
                SH_GREEN.setVisible(false);
                SH_YELLOW.setVisible(false);
                SH_RED.setVisible(true);
                TXT_TAKEOVER.setVisible(false);
        }

        var txt_mase=mase.toString().substr(0,10);
        var txt_rmsse=rmsse.toString().substr(0,10);
        var txt_mape=mape.toString().substr(0,10);
        TXT_MASE.applyText(txt_mase);
        TXT_RMSSE.applyText(txt_rmsse);
        TXT_MAPE.applyText(txt_mape);

        Application.hideBusyIndicator();
}
```