LATVIA UNIVERSITY OF LIFE SCIENCES AND TECHNOLOGIES
Faculty of Engineering and Information Technologies
Institute of Computer Systems and Data Science

KAMALESHWARAN ASOKAN

Big Data Time-Series Analysis and Visualization Using Python

Master thesis
for obtaining an engineering master's degree
in Information Technologies

Supervisor: Dr. prof. L.Paura

....
signature, date

Author: Matr. No. IT25005 K.Asokan

....
signature, date

Jelgava 2026

ANNOTATION

Asokan K. Big Data Time-Series Analysis and Visualization Using Python: Master’s thesis. Jelgava: Latvia University of Life Sciences and Technologies, Faculty of Engineering and Information Technologies, Institute of Computer Systems and Data Science, 2026. 91 pages, 6 tables, 16 figures, 49 information sources, 1 appendix.

The master’s thesis studies the data-driven process of development of the analytics and visualization model for large-scale time-series sales forecasting using the Python-based set of tools and the state-of-the-art machine-learning algorithms. The topic of work is chosen in the context of a large demand for interpretable and accurate models for business-oriented data science applications to understand trends, seasonality and structural changes in the series of sales.

The research studies a unified approach to preprocessing, modelling and visualization of large time-series data related to sales. The thesis first part (theoretical background) reviews a general background of time-series analysis and forecasting, including ARIMA, SARIMA and state-of-the-art forecasting models such as hybrid machine-learning and ensemble algorithms. The second part (the practical part) is dedicated to the implementation of a unified data processing pipeline with data analytics and visualization tools in Python using the data science libraries Pandas, Matplotlib, Plotly and Scikit-learn along with the forecasting algorithms ARIMA, SARIMA and Prophet.

The resulting prototype automatically performs data preprocessing and cleaning, then uses the selected algorithms to train and validate models, and visualizes the forecast values together with statistical accuracy metrics to estimate the quality of the prediction. The developed solution provides data processing and forecasting pipeline that can be easily configured and extended with new features and is supported by the benchmark results that show superior performance of hybrid machine-learning models on more volatile datasets with a lack of historical data, and the interpretability and resilience of Prophet and SARIMA on less chaotic data.

The work has both theoretical and practical significance. It shows that modern open-source tools can be used in the development of business intelligence systems to provide information support for decision-making. It has the integrated solution for time-series data exploration on large datasets to identify periodic patterns and outliers and produce the data visualizations to present a meaningful forecast together with a mathematical estimation of accuracy in a business context.

ANOTĀCIJA

Asokan K. *Lielo datu laika rindu analīze un vizualizācija ar Python*: maģistra darbs. Jelgava: Latvijas Biozinātņu un tehnoloģiju universitāte, Inženierzinātņu un informācijas tehnoloģiju fakultāte, Datoru sistēmu un datu zinātnes institūts, 2026. **91 lappuses, 6 tabulas, 16 attēli, 49 informācijas avoti, 1 pielikums.**

Maģistra darbā pētīta datu virzīta analītiskā un vizualizācijas ietvara izstrāde laika rindu pārdošanas prognozēšanai, izmantojot Python programmēšanas valodu un mašīnmācīšanās metodes. Darba mērķis ir risināt ar vien pieaugošo uzņēmumu nepieciešamību pēc precīziem un interpretējamiem prognozēšanas modeļiem, kas spēj identificēt tendences, sezonālās svārstības un strukturālās izmaiņas komerciālajos datos.

Pētījumā piedāvāta metodoloģija pārdošanas datu sagatavošanai, modelēšanai un vizualizācijai. Teorētiskajā daļā analizēti laika rindu analīzes un prognozēšanas pamatprincipi, tostarp autoregresīvie un slīdošā vidējā modeļi, kā arī hibrīdās pieejas. Praktiskajā daļā izstrādāts integrēts Python risinājums, izmantojot tādas bibliotēkas kā Pandas, Matplotlib, Plotly un Scikit-learn, un izmantoti prognozēšanas algoritmi ARIMA, SARIMA un Prophet.

Izstrādātā sistēma veic automatizētu datu pirmapstrādi, modeļu apmācību un validāciju, ģenerējot prognožu vizualizācijas un statistiskos rādītājus modeļu precizitātes novērtēšanai. Rezultāti apstiprina, ka hibrīdie mašīnmācīšanās modeļi nodrošina augstāku precizitāti nestabilos datu apstākļos, savukārt Prophet un SARIMA modeļi nodrošina interpretējamību un stabilitāti tirgus datos ar izteiktu sezonālo raksturu.

Izstrādātais risinājums ļauj efektīvi analizēt laika rindu modeļus lieliem datiem, nodrošina reproducējamību un ļauj attīstīt lielo datu analītiku un izmantot to pārdošanas datu prognozēšanai.

TABLE OF CONTENT

INTRODUCTION

The aim of the thesis is to implement and evaluate an information technology (IT)-based solution for analysing and visualizing big sales time-series data. The task involves identifying and forecasting sales patterns, trends, and seasonality in big datasets with the help of data analytics and visualization tools. It also requires the provision of recommendations for data scientists and The managers for choosing effective approaches to data analytics and business decision-making based on time-series visualization. The project is built on the use of statistical learning techniques and visualization methods in the Python programming language.

The tasks of the master's thesis, which are designed to achieve the aim, are as follows.

- Review the theoretical materials on time-series analysis and forecasting models, including AR, MA, ARMA, ARIMA, SARIMA, VAR, and VECM.
- Explore and analyse the existing big data visualization tools and techniques for business analytics.
- Choose and prepare a large sales time-series dataset for analysis.
- Implement the Python-based analytical and solution with tools. Apply and compare different forecasting models and approaches to detect and analyses significant sales patterns and seasonality.
- Evaluate the performance and accuracy of the forecasting models and visualize the interpretable results.
- Recommend solutions for better business forecasting, decision-making, and sales planning.

The thesis will cover the review of scientific literature, the specification of methods and technologies for the large-scale forecasting within the business context, the selection of datasets, and the implementation of the analytical and visualization processes for the uncovering of implicit temporal structures. Sales data is one of the most popular in the business environment due to its expressiveness of customer behavior, market changes, and operational dynamics, and its processing, in turn, with reliable forecasts can power inventory control, logistics planning, or even financial stability [8], [9]. Despite classical linear methods (ARIMA, SARIMA) provide interpretable and high-performing baselines for a well-structured time-series, modern business requires scalable and adaptable analytics pipelines to tackle high volume and velocity [8]. On the other hand, businesses require forecasting solutions that stay robust from the statistical perspective and that produce results that are easy to interpret and use in managerial decisions [9]. while visual analytics can aid business to understand and accept the model results [11]. For the above reasons, Python will be used as the implementation environment to create effective modelling and evaluation process, as well as create interactive dashboards for communicating results to a wide range of technical and non-technical users.

The thesis research direction is formed by the composition of three elements: theoretical, practical, and experimental. The theoretical is to set the time-series forecasting and visualization background, the practical is to implement a solution on real sales data in Python, and the experimental is to evaluate its predictive performance and the interpretability of the visual output to ensure decision reliability. The thesis is divided into four chapters. The **Chapter 1** describes the theoretical background of time-series analysis and forecasting models and business visualization tools. The **Chapter 2** shows the data collection, preprocessing, and implementation of the analytical and visualization solution. The **Chapter 3** reports the experimental analysis, the model comparison, results discussion, and the practical recommendations for the solution extension in the business context. The **Chapter 4** focus on design and implementation of an prototype to analyses and visualize sales data in the time-series and validates prototype through the forecasting results, assessing the model performance and business implications. The thesis is closed by its summary and recommendations for further work on the scalable and interpretable forecasting and visual analytics systems for sales data.

1 THEORETICAL FRAMEWORK OF TIME-SERIES ANALYSIS AND VISUALIZATION

The aim of Chapter 1 is to provide a concise synthesis of the main scientific concepts, analytical models, and technology frameworks that underpin the research project of implementing a big-data–driven solution for time-series sales analysis and visualization. The corresponding theoretical framework describes the nature of temporal data and behaviour, the statistical and computational models that reflect this behaviour, and the mechanisms for communicating analytical results to an end-user through graphical user interfaces. The purpose of creating this framework is to ensure that the subsequent empirical work is based on the most important and generally accepted scientific positions and is consistent with the international state of the art in data analytics and information technologies. The basic hypothesis of time-series forecasting is the possibility of obtaining information about the future based on the knowledge of the past, the realisation of which is based on learning regularities from historical data. Time-series data, as opposed to cross-sectional data, have the property that their observations are arranged in time order and are often correlated between successive time periods. The theoretical concepts of temporal dependency, stationarity, and seasonality form the basis for selecting models and assessing the confidence in forecasting [1], [2]. A time-series itself is a set of observations that contains several interconnected and typically hierarchically arranged components: a trend, cycle, seasonal component, and irregular or random component [1], [3]. Decomposing these components allows one to consider how the sales behaviour of the time-series changes, and to identify the stationary components, which are important for the purpose of using forecasting in such business problems as production planning, pricing, and inventory control [4].

The development of the theory and methods of forecasting can be described as a continual search for an optimal balance of model simplicity, interpretability, and accuracy of predictions. The early ARMA-type models, based on the autoregression (AR) and moving average (MA) models, are simple models that are designed for the modelling of stationary processes under the conditions of data scarcity [5], [6]. Later, their natural development into the autoregressive-integrated–moving-average (ARIMA) model allowed modelling a non-stationary series using differencing and the seasonal ARIMA (SARIMA) variant of the model for processing a time-series with a seasonal component. These statistical methods are the theoretical basis and classical paradigm for systematic quantitative forecasting. Their theoretical transparency and analytical accessibility make them ideal for use in business practice, where their advantage is the explainability of their models and verifiability of their internal logic [7] univariate statistical models are suitable for representing the internal temporal structure of the series, real business and economic phenomena do not exist in isolation. For example, sales of goods in particular periods of time will be affected by a large number of external and internal factors, such as advertising spending, competitors' activities, and macroeconomic indicators. To describe these phenomena, more complex multivariate modelling approaches were proposed, for example, the vector autoregression (VAR) model and vector error-correction model (VECM). The theoretical justification of this approach is that all the variables of the system are mutually related, that is, each variable of the system is considered as endogenous. This theoretical idea, developed in particular by [12], makes it possible to take into account that all the variables of the system both affect and are affected by each other. In business analytics practice, this approach makes it possible to quantitatively describe, for example, the impact of changes in advertising expenditure or price on sales in different sales channels over time.

However, despite their analytical flexibility, purely statistical models have linearly limited. For many commercial data sets, non-linear reactions, structural breaks, and context-dependent effects are not uncommon, which cannot be fully described by a linear function. The solution to this problem has been the development of machine-learning and hybrid modelling approaches. Machine-learning models, such as random forests, gradient-boosted ensembles, recurrent neural networks, and long short-term memory (LSTM), can be trained to approximate complex non-linear functions without explicit knowledge of their analytical form. Hybrid models combine machine-learning techniques with classic statistical processes and were also developed to provide the combined advantages of both types of models. The former type of models allow you to describe more precisely such components of the time-series as deterministic trends and cycles, while the machine-learning component allows you to learn such complex and irregular components of time-series as non-linear and multiplicative effects. Research also shows that such hybridisation can significantly improve the accuracy of forecasts for sales in complex retail and financial systems [13].

In addition to the task of prediction, visualization can also be considered as an important theoretical direction in the development of analytical work. Data visualization as a research discipline is about the transformation of computational models into graphical representations that have the ability to improve human understanding and communication. The task of visualization is also related to the theory of business forecasting as an integral part of the analytical process. The main theoretical aspects that determine the effectiveness of forecasting visualization in the business context are related to the principles of perceptual cognition: visual clarity, proportional encoding, and background distinction. Today, visual analytics is the discipline that integrates automated algorithms with human perception and provides interactive functionality for users to work with multidimensional data and verify the results of models. Research into the mechanisms of visual perception shows that graphical displays allow an analyst to more quickly and accurately find temporal trends than if the same tables of numerical values were used [14]. The implementation of theoretical ideas about visualization and big-data analytics is carried out by means of modern data-processing technologies. Distributed computing and data storage frameworks such as Hadoop and Apache Spark have revolutionised [15]. the very concepts of working with time-series data, as they allow parallel processing of petabytes of raw, semi-structured, and unstructured data [16]. Coupled with in-memory databases and stream-processing tools, such as those used in real-time analytics, these technologies also allow creating a near–real-time forecasting pipeline [17]. This provides the main components of a modern data-driven forecasting architecture in which programming languages and development environments such as Python can be used as unifying and open interfaces [18]. Modules and methods for machine-learning and statistical modelling, as well as visualization, have been developed in Python, for example, in the Pandas, NumPy, Matplotlib, Seaborn, and Plotly libraries. The cross-language integration of these tools is provided by their built-in reproducibility and compatibility features, which aligns with the principles of scalability and maintainability specified in contemporary information-systems theory [19].

In terms of academic referencing, the theoretical synthesis presented in Chapter 1 of the thesis places time-series forecasting in a multidisciplinary context, linking statistical inference, computational intelligence, and cognitive science. It also shows that to solve business problems using forecasting, it is important not only to build an accurate model but also to provide an interpretability system for users to understand the level of uncertainty in the forecast and its rationale. This theoretical structure therefore has two functions: on the one hand, it describes the mathematical underpinnings of prediction and, on the other, the communication systems that allow quantitative results to be transformed into information for business users. The structure of Chapter 1 is also the result of progressive conceptualisation and logically follows from it. Section 1.1. describes time-series data, their properties, and definitions. Section 1.2

then introduces the models that form the classical baseline for quantitative time-series analysis. In Section 1.3, the multivariate and vectorial models that are theoretically able to describe several interdependent variables are considered. The more recent trends in the theoretical development of forecasting methods, which are based on machine-learning and hybrid modelling, are the subject of Section 1.4. The final section of this theoretical chapter 1.5 also concludes the main theoretical review with the discussion of the big-data analytics and visualization frameworks for forecasting, the main functions of which are to operationalise the results of computational experiments in business applications. In general, all these subsections, by means of an internal logical connection, form a unified theoretical structure in which, starting from the properties of data, an algorithm for their analysis is described and its technological implementation is also determined, while visual interpretation is also the conceptual key to its effective use. Creating this theoretical framework is critical to the entire research work because the theoretical concepts presented in Chapter 1 will determine the most appropriate solutions for all subsequent work. On the other hand, the methodological and algorithmic techniques described here ensure their compliance with the international standards of scientific research in forecasting and with the requirements of the engineering perspective of embedding analytical models into a holistic information system. The resulting synthesis therefore provides the scientific and technological basis for the practical, implementation phase of the thesis, which aims to materialise the described concepts in a software solution for the analysis of time-series sales data and their visualization in Python.

1.1 CONCEPT AND CHARACTERISTICS OF TIME-SERIES DATA

Time-series data are sequences of observations recorded at successive and evenly spaced time intervals. This type of data structure is one of the most common and straightforward types of data in contemporary data science and machine learning. A time-series dataset represents samples of a measured variable (or variables) taken at regular time steps that can vary in time resolution, such as hourly, daily, weekly, and monthly intervals, and may contain missing values and other anomalies (Figure 2). Each element of a time-series sample is a tuple that associates a numerical value of a variable with a time stamp [4]. This is in contrast to cross-sectional or tabular data, which are independent samples taken from a collection of units, typically entities and entities, at one or more points in time [4]. The key feature of time-series data is the temporal ordering of observations, which introduces a natural serial dependence that must be preserved and explicitly represented in the corresponding model.

Time-series data analysis aims to identify structural components that are believed to compose the total observed variation (Figure 3). These components include trend, seasonal, and cyclic variation, as well as irregular error. The trend indicates the long-term movement of the series that reflects the underlying process's growth or decline. The seasonal variation is the systematic, often-periodic fluctuation around the trend that is due to calendar or calendar-free recurring patterns and rhythms, for instance, due to monthly or quarterly sales seasonality, business cycles, weather, or holidays. Cyclic variations capture non-seasonal or non-periodic fluctuations in the series. Finally, irregular variation or "noise" in the data is the residual component not explained by trend, seasonality, or cyclic factors [20]. The primary goal of time-series analysis is to extract the past trends and seasonal patterns and use them to generate accurate future projections of the sales variables of interest.

The analysis and decomposition of time-series data are important in a business context where companies must continuously plan and forecast sales to meet their future demand and optimally allocate their production and marketing resources. In sales forecasting applications, for example, a retailer can identify temporal patterns in sales demand and use them to forecast expected demand for each product. Decomposing the sales time series into trend, seasonality, and other components can be shown to improve time-series forecasting [21]. Decomposition results help in visualising and more parsimoniously modelling the past series and enable robust and accurate forecasting [12]. A variety of techniques exist for decomposing time series into trend, seasonality, and other components. Classical and modern decomposition methods include additive and multiplicative approaches to isolate trend and seasonal components in sales series and can be applied before time-series modelling. Univariate time series only involve a single variable or feature, while multivariate time series comprise multiple interdependent variables. In retail sales data, the historical sales series for a product is often a univariate time series, while a related price, promotion, weather, or competitive activity series is also a related multivariate time series [12]. A multivariate modelling approach can be used to account for all known related causal relationships and co-movements between multiple related sales and exogenous series.

Another important characteristic of time-series data is temporal autocorrelation, which measures the correlation between a time series and a lagged version of itself. Autocorrelation is an indication of temporal dependence or memory in the series and can be used as a primitive for autoregressive modelling. A time series is said to be stationary if its statistical properties, such as the mean, variance, and autocovariance, do not change over time [6]. Stationarity is an important property because it implies that the past series' statistical properties can be used to make inferences about the future series. Non-stationary series that contain trend or seasonality must be transformed to make them stationary before a forecasting model is fit. An important characteristic of data in a modern enterprise environment is the large volume of sales and transactions data that are continuously generated at a fast rate using online platforms, enterprise resource planning systems, and point-of-sale networks. This data is often structured but often shows characteristics of the so-called five Vs of big data, including volume, velocity, variety, veracity, and value [19]. Old-school forecasting models developed in the era of small data, such as Excel and SQL databases, are not equipped to handle modern big data. Instead, the framework and integration of big-data methods with traditional statistical and machine learning models are needed to analyse real-time streaming time-series data.

Temporal aggregation in time-series data is another property that should be taken into account in the modelling and forecasting framework. For example, sales can be aggregated at daily, weekly, or monthly granularity to show different patterns. Temporal variation in time-series data can be high-frequency or low-frequency, and the data's resampling frequency can impact the results [7]. For example, hourly or daily sales data may capture more market noise and are subject to daily temporal variations than monthly or weekly data [4]. In such cases, resampling and smoothing methods such as moving averages or exponential smoothing may be needed to balance accuracy and generalisability. The granularity and resampling frequency used for the analysis should reflect the specific goals of the modelling and the relevant time scales of the decision-making processes it will inform. The level of temporal aggregation used in time-series analysis can also affect model performance and the quality of business decisions based on forecasts. Sales data are now typically recorded in modern companies using sensors and automatic online data collection through digital platforms, e-commerce tools, digital marketing and sales platforms, and point-of-sale systems at the terminal. The value of sales is often recorded together with other transaction attributes, including product category and details, location or region, and customer type and segment. The richness of data types introduces heterogeneity into the datasets and should be considered in preprocessing steps, such as missing-value imputation, outlier analysis, and normalisation [22]. The overall data pre-processing quality will affect the time-series forecast quality.

The statistical significance and accuracy of time-series analysis should be complemented with effective visualization and communication tools. Visual analytics and exploratory techniques of time-series data remain one of the most important and useful practices in preliminary data inspection (Figure 4). In the context of this thesis, visualization serves the important function of visual verification of the results and also acts as a link between the complex time-series models and non-technical end-users. For example, for business and executive stakeholders, the ease of understanding and consuming the forecast results will be facilitated by their effective visualization. The benefits of using layered and interactive visualizations for communicating time-series models and results are supported by recent studies on visual analytics and business intelligence [23]. The time-series data types and concepts formalised in this section form the theoretical foundation of the remainder of this work and provide essential context for the description of the dataset and modelling approach that follow in later chapters. By formalising key characteristics, including temporal dependence, stationarity, and time-series decomposition, the theoretical background provides justification for using statistical time-series models and visualization tools in analysing and forecasting large-scale sales data.

1.2 STATISTICAL FORECASTING MODELS

Statistical forecasting models are the traditional class of time-series prediction techniques and have maintained their status as strong baselines in both research and practice. They operate under the premise that each new observation in a time series is a function of past realisations and random shocks. The most traditional time-series models are autoregressive (AR), moving-average (MA), autoregressive moving-average (ARMA), autoregressive integrated moving-average (ARIMA), and seasonal ARIMA (SARIMA). These approaches constitute a class of linear stochastic models that can be used to model temporal dependencies in sales, financial, and industrial time-series data [20]. Autoregressive models use the principle that past observations of a process can be expressed as a weighted average of the few most recent ones. An advantage of this simple principle is its ability to represent data persistence or momentum in the time series. In a business context, the autoregressive structure can be used to help identify reinforcing patterns of demand or supply. For example, product sales may be high for a number of months in a row due to brand loyalty or promotional campaigns. Empirical research has found that autoregressive processes of low order often give good results for the short-term forecasting of stable products [2].

In contrast to autoregressive models, moving-average models are used to model the propagation of stochastic shocks through the series. Instead of directly depending on the recent realisations of the variable, they are specified as a function of recent random forecast errors. This model mechanism has been shown to be useful for describing short-term fluctuations caused by temporary external disturbances such as supply-chain disruptions or one-off price promotions. For example, empirical work in the field of operational forecasting has shown that moving-average models are a good option when sales data show erratic spikes followed by a quick return to normal levels [6]. The autoregressive moving-average model can be viewed as a hybrid approach that combines the memory effect of AR with the error-correction property of MA. The ARMA model structure requires that the time series is stationary, which means that its statistical properties, such as the mean and variance, do not change through time. ARMA models have historically been the workhorses of classical forecasting in the 20th century due to their interpretability and ease of estimation. Their statistical efficacy has been validated across a wide range of application areas, such as energy demand, assortment planning, and macroeconomic forecasting, where data exhibit stationarity over long forecasting horizons [22]. However, real-world business data, and especially sales series, are often characterised by non-stationary patterns as a result of growth trends, expansion to new markets, or seasonality. To account for such dynamics in time-series data, the autoregressive integrated moving-average model was proposed. ARIMA is a model structure that incorporates an integration term to address non-stationary behaviour in time series. The integration term of the ARIMA model differentiates it from ARMA and is used to capture both short-term dependencies and long-term movements in the same model structure.[4]. show that the flexibility of ARIMA in handling various forms of non-stationarity make the model one of the most popular methods for business forecasting.

ARIMA is a robust model class but sales data often contain some form of seasonal cycles. Retail demand for a product can increase during holidays or special occasions and can experience a sudden drop during the off-season months. The seasonal autoregressive integrated moving-average model is an extension of ARIMA that adds seasonal autoregressive and moving-average terms to explicitly model periodic patterns. SARIMA models allow a series to be influenced by past realisations and forecast errors both within a year and across different years, which can be leveraged to improve forecast accuracy for cyclic processes such as quarterly production or sales levels. Empirical evaluation has found that SARIMA usually outperforms non-seasonal methods when clear seasonal patterns are present in a series [9].

The estimation and validation of the models also have a well-defined statistical procedure. After the model order is selected, the parameters' values are usually estimated via maximum-likelihood or least-squares. Diagnostic checks for white noise are then conducted to ensure that there is no remaining temporal structure in the residuals. Model evaluation and selection is often done via the Akaike Information Criterion (AIC) [24] which trade-off between model complexity and predictive accuracy (Burnham & Anderson, 2018). An over-parameterised model may fit the historical data well but will have poor out-of-sample generalisation while an under-fitted model will not be able to capture the relevant structure. Within a business analytics setting, one of the additional advantages of the classical models is their interpretability. Managers and analysts find it easier to understand the meaning of lag and seasonal parameters representing autocorrelation and seasonal persistence, which can help improve the communication between technical specialists and non-technical stakeholders. Recent work has also shown that despite the growing interest in machine-learning methods, ARIMA-based methods have remained as strong baselines for business forecasting due to their explainability and stable performance on moderate-sized data sets [7].

The historical progression from AR to SARIMA can be viewed as a refinement of the statistical paradigm for forecasting. Each iteration of the model family overcomes a key limitation of the previous by addressing an important source of non-random behaviour: AR begins by modelling autocorrelation, MA models noise propagation, ARMA combines both mechanisms, ARIMA addresses non-stationarity, and SARIMA incorporates seasonality. This incremental approach to building models is in line with the principle of parsimony, which guides the avoidance of using models with excess parameters. The entire class of ARIMA-type models can be seen as a family of methods for sales forecasting that trade-off between theoretical parsimony and practical usability. To conclude, statistical forecasting models are a coherent theoretical foundation for time-series prediction in a business context. The mathematical simplicity, diagnostic transparency, and demonstrated empirical performance make them indispensable to modern predictive analytics. While the next chapters of the thesis focuses on more advanced and hybrid methods, classical ARIMA and SARIMA models remain reference points by which all new models are benchmarked. The continued relevance of the classical forecasting models to this day highlights the importance of a solid understanding of linear time-series models before one can learn more complex, data-intensive techniques.

1.3 Multivariate and Vector-Based Models

Single series models such as ARIMA, introduced in the previous chapter, are often inadequate to capture sales dynamics in real-world settings. The reason is that these processes rarely depend on past sales only, but on a vector of potentially interacting variables. The most common among them are prices, marketing spend, economic indices and seasonal or exogenous shocks. While univariate ARIMA models can help specify the temporal pattern of a given time series, they cannot account for dependencies across different processes. The framework of multivariate time-series models is an extension of univariate methods to multiple related or contemporaneously changing series. By modelling a vector of processes at once, one is able to identify common structures in terms of short- and long-run effects. The most popular specifications in this class of models are Vector Autoregression (VAR) and its error-correction extension Vector Error Correction Model (VECM) [3]. These have become the workhorses for empirical forecasting and causal analysis in the areas of economics, finance and operations research [12]. The main idea behind multivariate modelling is to treat all variables in the system as endogenous or jointly determined. In other words, each of them can be seen as an outcome of a process influenced by its own lag values and the lags of all the other variables. This is the formal equivalent of modelling the variables of interest in symmetric, no-causality terms. The difference between VAR models and structural econometric systems lies in the fact that the latter attempt to impose a priori causal ordering on the variables. In practice, the VAR framework is often used to model the dynamic interactions between sales, different types of marketing inputs, consumer confidence indices or other macroeconomic variables of interest. A common business question might be to investigate the short- or long-term effect of an increase in advertising spend on sales, or to quantify the influence of changes in disposable income on the demand for the company's products. By modelling lagged interdependence, the VAR framework allows to measure such effects in an objective and data-driven way, without making restrictive assumptions about their size and signs [21].

A general VAR (p) model with k variables is a system of k linear equations in which each of the variables is regressed on its own p lags and those of all other (k-1) variables in the system. Ordinary least squares is the most commonly used and preferred estimation technique for VAR

systems, due to its consistency and efficiency as long as identical lag orders are imposed on all equations. Information criteria, such as Akaike and Bayesian versions, are often used to guide the order selection process to avoid over-parameterisation and underdetermination. The empirical literature shows that VAR models typically exhibit good forecasting performance relative to other models in both short- and medium-term horizons, subject to the structural stability and sample size requirements [25]. In particular, when the data are collected at higher frequencies, say, weekly or daily, multivariate models appear to provide a robust predictive framework. Before fitting such models, one needs to check if the series are cointegrated in the sense of having a common stochastic trend. If the individual series are found to be non-stationary but cointegrated, the VECM is a better specification than the unrestricted VAR. This type of models includes an error-correction term that provides long-run equilibrium restrictions on the level of the series, while allowing deviations around the long-run mean in the short-term dynamics. In the context of sales, it means that demand, price and marketing expenditure, for example, might deviate from their long-run equilibrium for a while, but will eventually converge to it subject to underlying economic and behavioural forces. This long-run/short-run relationship, formalised in Engle and Granger's cointegration theorem, underlies the theoretical basis for VECM specification [5].
The practical steps of fitting VAR or VECM models are as follows. First, make sure that all non-stationary series in the system are either differenced or detrended. Second, determine the optimal number of lags, usually via empirical information criteria tests. Third, the Johansen procedure can be used to test the number of cointegrating relationships in a system of series. Diagnostic checking of the residuals for autocorrelation and normality is often a crucial step for determining model adequacy. Properly estimated, VAR and VECM models can be used not only to provide point forecasts, but also to support structural interpretations through impulse-response and variance-decomposition analysis [3]. The former shows how an impulse (shock) to one of the variables would affect all others in the future, while the latter decomposes the movements in each variable into components due to shocks in all the variables in the system [26]. Table 1.1 below summarises some of the main properties of the univariate and multivariate frameworks, and their conceptual and practical differences. The focus is on the implications for common business use cases, including the assumptions of stationarity and causality, as well as the forecast horizons.

Table 1.1. **Comparison of Univariate and Multivariate Time-Series Forecasting Models** [12], [5], [26]

Feature	**UnivariateModels (ARIMA/SARIMA)**	**Multivariate Models (VAR/VECM)**
Variable Structure	Single dependent variable	Multiple interrelated variables treated as endogenous
Lag Dependence	Own lags only	Own and cross-variable lags
Stationarity Requirement	Stationary after differencing	Stationary or cointegrated systems
Long-Run Equilibrium	Not explicitly modelled	Captured through cointegration and error-correction
Parameter Complexity	Low to moderate	Increases quadratically with variable count
Interpretability	Straightforward coefficients	Requires matrix interpretation of lag interactions
Forecast Horizon	Short to medium	Medium to long, especially for structural analysis
Causal Analysis Potential	Limited	High, using impulse-response and variance-decomposition
Common Applications	Single-product demand, financial trend	Macro-economic indicators, multi-product sales systems
Typical Tools	ARIMA, SARIMA	VAR, VECM, Johansen cointegration tests

As can be seen in table 1.1, the most significant strength of the multivariate modelling approach is its ability to incorporate dependencies between the variables. In fact, the underlying ability to model such relationships enables the analyst to better understand how shocks from one market or product category affect other parts of the system, which is especially important in case of the businesses that are involved in multiple business areas or offer a set of different products. For instance, it is reasonable to expect that a significant increase in marketing activities in one of the product categories will not only increase sales of that particular product but also result in additional sales in other categories through higher overall brand awareness. Such spillover effects cannot be properly captured in the univariate modelling framework. As a result, both VAR and VECM models allow for more comprehensive and, therefore, more accurate description of the market system and its dynamics. This enables better in-sample policy simulations and ex-ante scenario analysis [27].

As can be seen from the discussion above, although the richer informational content and, therefore, greater potential analytical value of the multivariate models comes at the price of higher sample size requirements and the need to take care of the model identification to avoid overfitting the data. In fact, the presence of multiple variables in the model leads to the so-called parameter explosion, as the total number of parameters to be estimated in the system grows quadratically with respect to the dimensionality of the VAR. This aspect in particular results in significantly increased computational burden of these models, although in practice it can be mitigated through the use of regularisation methods and model-reduction techniques such as Bayesian VAR or factor-augmented VAR. In fact, recent literature reports that both shrinkage priors and penalised estimation methods in high-dimensional VAR frameworks provide better forecast accuracy and reliability. As far as the practical forecasting application is concerned, the most important aspect is probably related to the interpretation of the results of VECM modelling in particular. In fact, the coefficients of the error-correction term can be used as a direct measure of the speed of adjustment to the long-run equilibrium state. In other words, the small value of the coefficient will signal that the variable under analysis corrects its value in a small magnitude in the long run, which, in practice, means that it takes a long time to return to the point of long-run equilibrium, thus signalling weak market adjustments. On the other hand, a large coefficient on the error-correction term implies a significant and fast reaction. In practice, this means that the decision-makers in the company will be able to gauge the expected market reaction to their actions and to the various shocks affecting it. For instance, if it is found that the market of a product corrects a slump in sales rather slowly, the management can take the required steps in a more pre-emptive way, by conducting compensatory marketing campaigns before sales fall too much. Recent empirical studies applying the cointegration analysis to retail business areas report that using information about the long-term equilibrium in addition to short-term dependencies indeed significantly improves the reliability of the forecasts compared with purely differenced VAR models [25].

In addition to the considerations above, a natural modern application in analytics also concerns integration of VAR and VECM models into general machine-learning pipelines. In this case, it is possible to consider a hybrid approach, in which a linear multivariate model provides an interpretable baseline, which is then corrected through the use of a nonlinear learner, that is able to account for any additional structure in the error terms. Overall, such a combined approach would enable to use machine learning techniques to further improve the forecasting performance, without sacrificing the interpretability and, thus, the domain-theory transparency of the underlying model. In fact, both VAR and VECM are efficiently implemented in modern Python and R ecosystems, which makes them applicable to real-time data processing and visual analytics of the multivariate interdependencies [4]. In conclusion, models of the vector type represent an important generalisation of the univariate time-series analysis, which takes into account the potential multidimensionality of the business data. VAR provides a flexible and symmetric framework, which can be easily used to analyse any number of interrelated time series, while VECM represents its extended version, which also accounts for the presence of long-term equilibrium relationships, which is the case if the non-stationary time series share one or more common trends. In that sense, both VAR and VECM models are powerful approaches that can be successfully used for the purposes of gaining a more comprehensive understanding of a complex business system, in which the concept of causality and, especially, feedback mechanisms between various components and sub-systems are of crucial importance. The next chapter of this thesis further builds on the discussion in the sections above to design and implement computational methods to practically integrate VAR and VECM with modern Python-based visualization and forecasting tools, with the general aim to enhance the decision support in business through modern data-driven techniques.

1.4 Machine-Learning and Hybrid Approaches for Time-Series Forecasting

Traditional forecasting approaches, such as ARIMA, VAR, and Exponential Smoothing (ETS) models are based on parametric statistical functions and can therefore be considered as interpretable linear methods. Machine-learning (ML) and hybrid time-series modelling, on the other hand, are able to learn nonlinear relationships in time and/or to interact with input features in complex ways, while also being able to adapt to specific dynamic regimes and/or abrupt structural changes. These methods have been shown to be able to incorporate statistical, artificial intelligence, and optimisation techniques to deliver more accurate predictions in many business contexts [22]. The core advantage of ML-based forecasting is that its functional form is directly learned from the data itself instead of relying on prior assumptions. Machine-learning algorithms use specific non-parametric architectures such as artificial neural networks (ANNs), recurrent neural networks (RNNs), long short-term memory networks (LSTMs), gated recurrent units (GRUs), random forests, and gradient-boosting algorithms to learn from sequential information [28]. Attention-based transformer models have also recently been applied to business time-series forecasting tasks [13], [29]. For each of these architectures, multiple model instances can be trained and ensembled to increase robustness to data noise or scarcity. In general, RNNs (and in particular LSTMs) are often good choices for time-series forecasting when modelling long-term dependencies since their gated structures can effectively preserve information over multiple time steps while mitigating the issue of vanishing gradients [28]. Convolutional neural networks (CNNs) and other feedforward network-based approaches are sometimes better choices for noisy or shorter time-series.

In many cases, research has shown that deep neural forecasting approaches can achieve better performance on real-world business datasets that are complex and/or involve very high frequencies. For example, experiments comparing various recurrent and convolutional network architectures have shown that both can outperform more traditional models, given an adequate number of training samples and a sufficient degree of input feature engineering [13]. However, this benefit is sometimes accompanied by a loss of interpretability and increased computational complexity. Specifically, it can be difficult for managers to see exactly how a forecast is calculated in neural networks since there is no direct way to attribute importance to specific inputs or weights. This has led to the exploration of explainable machine-learning (XAI) approaches for time-series [31] which aim to provide post-hoc interpretation and visualization of model internals alongside prediction capabilities [32], [33]. A range of hybrid architectures can also be seen as forming a middle ground between traditional and modern machine-learning approaches. These strategies typically learn a statistical model that captures most of the linear structure in the data, while using a machine-learning component to learn residuals or nonlinearities. The motivation for this type of approach is that each component has a complementary advantage: linear models have been shown to be very useful for autocorrelation but tend to lack flexibility, while machine-learning components can more flexibly model nonlinearity, structural breaks, or threshold effects. A recent example from [2] showed that ARIMA-LSTM hybrid models significantly outperformed the individual approaches for forecasting retail sales, by combining a differenced linear model with sequence modelling of the residual errors to improve the mean absolute percentage error (MAPE) by more than 20 %.

Typically, data must be preprocessed before applying machine-learning methods to time-series. For example, missing values or outliers can have significant impact on model performance if not adequately addressed. Feature-engineering and transformation steps are also usually required for ML-based forecasting to assist the model in capturing useful temporal structures and/or dependencies. These steps might include adding lagged features, differencing the time-series, normalizing or scaling values, or encoding categorical variables. Feature selection methods can also be used for reducing dimensionality and overfitting, for example using mutual information or recursive feature elimination [7]. In the business domain, these methods are typically combined with external variables that may represent causal relationships in the data (e.g. price indices, advertising/promotional spending, economic indicators), to increase generalisation to different time horizons or dynamic regimes. A recent trend in forecasting is the development of automated or end-to-end machine-learning platforms, which perform algorithm selection, hyper-parameter tuning, and/or model stacking within a single learning pipeline. Automated machine-learning (Auto ML) platforms usually benchmark hundreds of candidate algorithms using cross-validation and train ensemble models of top-performing candidates [34], [35]. This approach makes the method more objective (by removing the human bias involved in manual tuning) and can therefore often significantly reduce the time to deployment, while still being applicable at a large scale for business forecasting problems [36]. A related approach is to use ensembles of multiple types of models (for example gradient boosting, random forests, deep networks, and others) in order to average out error that is specific to a single model choice. Table 1.2 provides a brief comparison of statistical, ML, and hybrid forecasting methods with respect to the most important and differentiating properties of each approach.

Table 1.2. **Comparison of Statistical, Machine-Learning, and Hybrid Time-Series Forecasting Methods** [30], [49]

Criterion	Statistical Models (ARIMA, SARIMA, VAR)	Machine-Learning Models (LSTM, RF, GBM)	Hybrid Models (ARIMA-LSTM, SARIMA-XGBoost)
Model Assumptions	Linear relationships, stationarity	Nonlinear, data-driven, minimal assumptions	Combines linear and nonlinear structures
Handling of Seasonality	Explicit differencing or seasonal terms	Learnt automatically through patterns	Statistical preprocessing plus nonlinear correction
Interpretability	High because parameters directly explain temporal effects	Low to moderate as it requires post-hoc explainability	Moderate, classical base improves clarity

Criterion	Statistical Models (ARIMA, SARIMA, VAR)	Machine-Learning Models (LSTM, RF, GBM)	Hybrid Models (ARIMA-LSTM, SARIMA-XGBoost)
Data Requirements	Moderate, structured, stationary	High-volume, high-frequency, multidimensional	Moderate to high, depends on hybrid composition
Computational Demand	Low to medium	High due to iterative training	Medium, depends on architecture
Forecast Horizon	Short- to medium-term	Short- to long-term depending on architecture depth	Flexible across horizons
Performance under Nonlinearity	Weak	Strong	Very Strong
Typical Evaluation Metrics	AIC, BIC, RMSE, MAE	RMSE, MAE, MAPE, R^2	Combined metrics from both domains
Deployment Complexity	Low, well-established statistical packages	High as it requires specialized frameworks	Moderate because its implemented via hybrid pipelines
Examples of Use	Economic indicators, basic sales series	High-frequency retail, sensor analytics	Retail demand, energy forecasting, logistics

The comparative summary highlighted machine-learning approaches augment the scope of time-series forecasting by capturing nonlinear and domain-specific dependencies that may be missed by traditional models. Their ability to handle high-dimensional and diverse data makes them a compelling toolkit for digital-age businesses. However, their practical application is not free of obstacles. Data availability, interpretability, and computational resources are some of the challenges that must be considered. Hybrid models have emerged to address some of these concerns by offering a balance between analytical transparency and model adaptability. Employing interpretable statistical models as the core ensures managerial trust, while integrating machine-learning models as an augmentation layer can extract patterns missed by linear methods. This integrated modeling approach is in line with recent shifts toward holistic analytics platforms as part of enterprise information systems [37]. The promising direction for future time-series forecasting research is hybrid architectures that leverage deep learning for its modeling capabilities and combine it with probabilistic and causal inference. Models such as transformers, with attention mechanisms, have recently reported state-of-the-art performance by learning to assign dynamic weights to different temporal dependencies across variable sequences [38]. In addition, when uncertainty is quantified using such models, they produce probabilistic forecasts that represent not only expected values but also confidence intervals [39].

The outputs from probabilistic models provide a more robust decision-support system by offering managers the ability to understand the confidence bounds around the predicted values. In parallel, research has also been underway to bridge the gap in interpretability of black-box predictors. Techniques like SHAP (Shapley Additive Explanations) and integrated-gradient attribution now make it possible to decompose the forecast contributions by input features or time lags, offering transparency that was previously lacking in deep learning models [34]. The explainable-AI model architectures are responding to a greater need for accountability in algorithmic decision support, particularly in the case of regulated business domains. A seamless integration of these state-of-the-art models into the production systems is one of the steps that has been lagging in operationalising these advancements. The use of cloud computing resources and distributed data-processing architectures has paved the way for online training and model updating on streaming datasets. It is becoming commonplace for modern enterprise resource planning systems to interface with analytical microservices offering on-demand rolling forecasts, anomaly detection alerts, and dashboard visualizations. Python packages like Prophet, TensorFlow, and PyTorch Forecasting have matured over the years to standardize these capabilities, with ease of deployment now possible even outside academic research labs. The convergence of big-data infrastructure and predictive analytics reimagines forecasting as not only a periodic analytical exercise but also an integral operational capability. The theoretical and empirical considerations outlined in this section underscore the critical role of machine-learning models and their hybrid counterparts in predictive analytics. When designed appropriately, their ability to discern complex temporal interactions, adapt to emerging patterns, and offer explainability is key to realizing more agile and data-driven management. The forthcoming sections of the thesis seek to build upon this theoretical underpinning by detailing a practical implementation that leverages Python-based open-source packages and visualizations to operationalize these forecasting techniques for large-scale sales datasets.

1.5 Big-Data Analytics and Visualization Tools in Business Contexts

The process of digitisation in organisations has led to an explosion in data in terms of volume, variety, and velocity. Systems like enterprise-resource-planning (ERP) platforms, e-commerce engines, and customer-relationship-management (CRM) applications are routinely generating timestamped records of sales, transactions, user activity, and much more. Harvesting actionable insights from this information deluge demands analysis frameworks that can accommodate unstructured and semi-structured data and that provide easy-to-use interfaces for human interpretation. Big-data analytics and visualization tools are addressing these needs by helping organisations to unlock insights from large datasets through discovery of patterns and actionable intelligence, and for sharing these insights across the organisation to improve the business decision-making process [19]. These tools are best used in conjunction with forecasting models to enrich both their analytical and communicative components. Big-data analytics is a suite of distributed technologies [40] and associated computation models that can efficiently manipulate very large datasets. Distributed processing frameworks, such as Hadoop, Spark, and Flink, provide parallelism over clusters of computing nodes for storing and analysing datasets, enabling horizontal scalability and partitioned fault tolerance, unlike traditional relational databases. These are key for pre-processing time-series data from multiple streams of business operations. A three-tier architecture for a modern analytics pipeline typically includes data acquisition, data storage, and data analysis. Acquisition can involve batch or stream ingestion of data from transactional systems and/or external connectors (APIs). Distributed file systems and cloud data lakes can be leveraged to store and manage the underlying data. Statistical and machine-learning models can be then applied to process the data to produce forecasts, correlations, anomaly detections, and so on.

Visualization can be seen as an additional layer for presenting complex analytical results, which involves mapping analysis outputs to an intuitive graphical representation to aid interpretation and rapid mental processing. According to several studies, information gleaned via visual perception feeds directly into cognitive processes and is also used by the human brain to support executive decision-making through efficient compression of information, for example by summarising, condensing, or filtering [14]. Interactive data dashboards and reports allow business managers to drill-down through hierarchical data, filter parameters, and visualise the temporal progression of various metrics without needing statistical or mathematical backgrounds. As a result, visualization is seen not only as a means for outputting results from analytics, but also as a modality for feedback, bridging human and computational logic. Before delving into the details of various data-visualization tools, it is instructive to first be clear on the role of data visualization within an analytics workflow. Visual analytics is the fusion of automated analysis techniques with human cognition for both discovery and validation of patterns. It can be applied to support descriptive, diagnostic, predictive, and prescriptive analytics stages, and has the goal of translating abstract numeric quantities into visual metaphors like lines, bars, heatmaps, scatterplots, or networks. In the case of predictive modelling tasks like time-series forecasting, visualization can also allow for inspection of residuals and diagnostics of model errors, direct comparison between predicted and observed values, and communication of model uncertainty.

Over the past decade, a rich ecosystem of open-source and enterprise big-data visualization tools has evolved, which can vary on several dimensions including ease-of-use, scalability, customisability, data-management requirements, and so on. Most of these have adopted different commercial or open-source analytics platforms as the underlying engine for computing and serve data to front-end graphical interfaces in one form or another. Tableau, Power BI, Qlik Sense, Google Data Studio, and Apache Superset are popular for commercial use, while Matplotlib, Seaborn, Plotly, and Bokeh are preferred in Python-based analytics workflows. The emphasis across these tools vary depending on the underlying technology, for example, Tableau emphasises user-centric drag-and-drop interactivity, Power BI better integrates with the Microsoft stack, and Plotly and Bokeh are both highly customisable and support programmatic dashboard development [41]. Figure 1.1 below shows a high-level view of a big-data analytics architecture for time-series forecasting and visualization. It shows how data is sourced from acquisition systems and is then stored in a distributed storage layer and subsequently processed using a cluster of analysis nodes, the output of which are presented to business users through interactive visual dashboards.

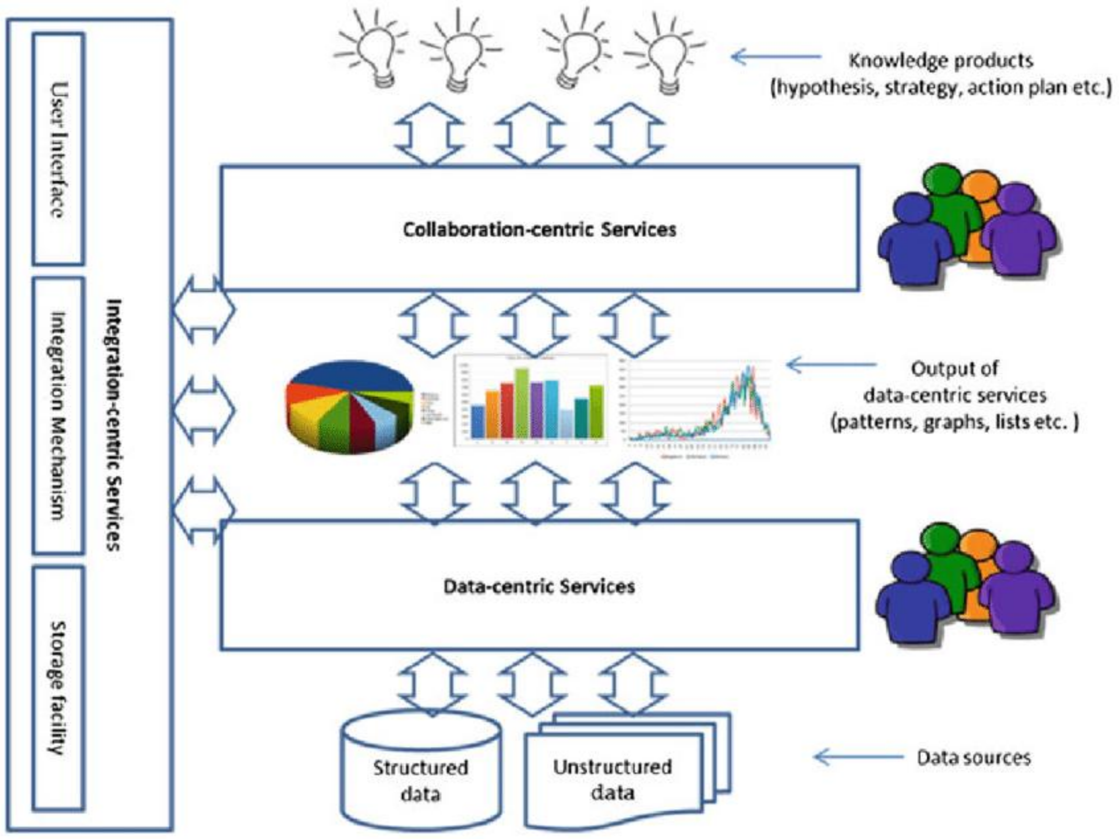


Fig 1.1. **Conceptual architecture of big-data analytics and visualization workflow for business forecasting** [47]

Figure 1.1 above shows a typical modern architecture. The data source, in this case sales data, would first need to be collected, then preprocessed before feeding the engines. Note that the data here might come from multiple channels, each with its own format. The code cells in this Jupyter Note book would have been built as a pipeline for ETL (Extract, Transform, Load) in a production environment, most probably using stream-processing framework such as Kafka or Spark Streaming to enable real-time low-latency ingestion. Any forecasting engine, ARIMA, LSTM, or others can be written in the analytical layer with results passed to the visualization layer as a single or multiple dashboards. Hence, in a team where data engineers, ML/DL engineers, analysts, and analysts-in-training work separately, data visualization interfaces provide that separation of concern. In contrast to static snapshots, many use-cases benefit from dashboards that refresh in real-time as the source data is being updated. For instance, the end-user, in this case a manager, might need to watch the incoming sales numbers, immediately watch updated forecasts, and adjust stocks in warehouses. If, in addition, there are pipelines to preprocess and monitor shipment orders from suppliers, real-time anomaly detection dashboards for logistics delays can be implemented. It has been found that such real-time visual analytics adoption confers competitive advantage in terms of operational responsiveness and revenue forecast accuracy [42].

The next major decision concerns the visual encoding and interaction metaphors. Research in visual information processing show humans more efficiently judge magnitudes, positions, and colour encodings than areas and volumes [43], [44]. The implication of this is that line charts, bar charts, and similar continue to be the best for visualising changes over time, while heatmaps, scatterplots, and others best for showing association across multiple variables. Portfolio sales and category sales analysis might use hierarchical visualizations like treemaps and sunburst charts. The literature is consistent that improper usage of any of these lead to misinterpretation [14], as such, analysts need to be aware of these during design. Table 1.3 below compares several popular libraries and products in common use.

Table 1.3. **Comparison of leading big-data visualization tools for business analytics**

Tool / Framework	Programming Integration	Interactivity Level	Scalability	Typical Use-Cases	Licensing
Tableau	GUI-based, Python / R API	High	Enterprise-grade	Management dashboards, KPI tracking	Commercial
Microsoft Power BI	Excel, Azure, Python API	High	Cloud / Enterprise	Financial and operational reporting	Freemium / Commercial
Plotly / Dash	Full Python support	Very high	Moderate to large data	Interactive analytics, time-series exploration	Open source / Commercial
Bokeh	Python / JS integration	High	Moderate	Scientific and custom visualizations	Open source
Apache Superset	SQL + Python connectivity	High	Enterprise-level distributed systems	Embedded data dashboards	Open source
Google Data Studio (Looker Studio)	Cloud data connectors	Medium	High (through BigQuery)	Marketing and web analytics	Cloud-based free tier

Table 1.3 illustrates that open-source libraries (Plotly, Bokeh) often focus on flexibility and customisability that can be tailored to the needs of technical analysts, while commercial

products (Tableau, Power BI) often focus on user-friendliness and compatibility in the enterprise environment. Many companies use a combination of open-source and commercial platforms to serve their business use cases. Open-source platforms are sometimes used to conduct complex model runs and visualizations, and proprietary software is used for management reporting. The open ecosystem of many libraries also means that the results from one visualization tool can be programmatically imported to another with web APIs and REST connectors. A recent trend is to incorporate machine learning to provide auto-generated insights. Intelligent dashboards may provide anomaly detection, natural-language descriptions, and next-best-view recommendation engines that guide the user in exploring the visual views that are most relevant for them. Interactive visual analytics also makes it easier for non-experts to perform advanced queries without needing to code, by using natural or voice-based query languages. An active field of research is to develop cognitive analytics interfaces that can further aid human comprehension and decision making. There is evidence that auto-generated visual narratives support better user understanding and recall than static visualizations [45].
An additional recent trend is for visualization tools to run natively on cloud platforms. Cloud-based dashboards such as AWS QuickSight, Google Looker, and Azure Synapse allow remote teams to collaborate on a single shared dashboard and provide real-time access to analytics results. Dashboards in the cloud can be collaboratively analysed using version control, permission management, and commenting features. Cloud-based analytics are also more easily scalable to handle large volumes of data, which is especially important for large organizations that operate in many different regions at once. This growth in analytics also brings considerations of ethics and data security. Adherence to data-governance principles around privacy, accuracy, and accountability is required to ensure that data visualizations are both ethical and secure. Combining external and internal data on interactive dashboards also requires visual analytics software to have access control and data anonymization features that prevent data leaks. There is a growing set of frameworks around responsible data visualizations, which among other things discourage overstated accuracy and instead focus on honestly conveying uncertainty and providing visual context for the visualized data [46]. These standards will be important for helping to ensure the public continues to trust decisions made in an increasingly analytics-driven world. Figure 1.2 below is an example of an interactive time-series visualization that is commonly seen in business intelligence dashboards. The figure depicts a multi-series line chart which combines historical sales data with a model forecast and

confidence intervals. This chart format provides a compact view of the performance trajectories over time and the associated uncertainty with the forecast.

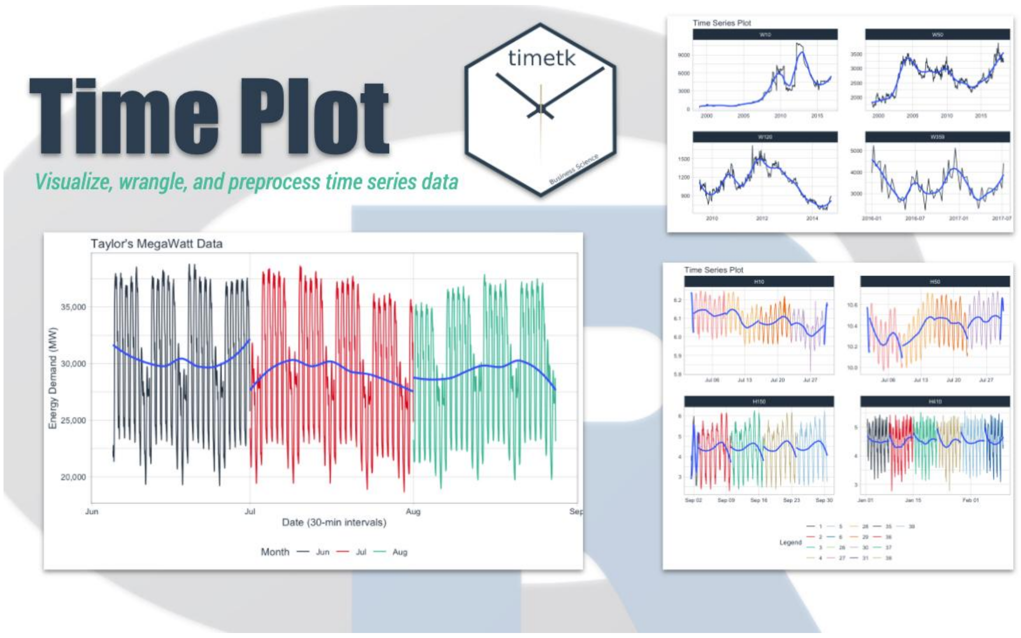


Fig 1.2**. Example of interactive time-series visualization in business analytics** [41]

Moreover, in such a plot as given in figure 1.2, the author can visualise not only the trend that he or she forecasts, but also the precision with which it has been predicted. Confidence bands represent a visual encoding of the probability distribution of the predictions, which can inform the manager in which points of the time series the risk could be higher [47]. Interactive visualizations are more flexible than their static counterparts, as it is possible for the user to zoom into a specific period, compare different alternative models, and annotate particular events of interest. In such a way, explanation through visualization should be seen as a complement to the numerical forecasting, enhancing the level of transparency within an organisation and the communication between its technical and non-technical departments.

In conclusion, big-data analytics software and data visualization tools have become an integral part of the current analytical environment, by providing the connection between technical statistical models and human understanding. Their theoretical and applied knowledge is a key step towards the design and implementation of end-to-end forecasting systems, where a distributed computation, interactive visual design, and automated learning are combined to provide enterprises with the competitive edge in the fast-changing market. In this thesis, these concepts will be applied to design and code a Python environment for time-series data visualization, capable of generating accurate and evidence-based business forecasts

1.6 Summary of Theoretical Findings

Chapter 1 has provided a theoretical context for the analysis and visualization of big-data time-series sales. It has covered a range of insights from several academic disciplines, including statistics, econometrics, computer science, and information systems. In doing so, the chapter has provided the conceptual background for the research and the understanding necessary for the implementation.

Section 1.1 has provided a brief theoretical overview of time-series. It has been shown that, in contrast to cross-sectional data, time-series observations have temporal dependency, a possible structural pattern, and stochastic noise. The understanding of these three properties of time-series data is fundamental to time-series forecasting. Decomposition of the level, trend, seasonality, cyclicity, and irregular components helps distinguish regular structures and use methods appropriate to those structures. Thus, it was theoretically demonstrated that time-series analysis aims to explain and predict the target variable based on the model and, at the same time, this process requires that the properties of the model are consistent with the properties of the data [4].

Section 1.2 has theoretically examined traditional statistical time-series forecasting models. The literature overview showed that AR, MA, ARMA, ARIMA, and SARIMA models have been widely used and are still relevant. The use of these models for forecasting the sales volume allows for the clear interpretation of parameter estimates, as well as the possibility of exploiting simple mathematical properties. For this reason, the theoretical knowledge confirms that, despite the large number of new tools, traditional methods are still actively used for time-series forecasting, often being the first ones to be used [6].

Section 1.3 has introduced multivariate extensions to time-series analysis. The theoretical review of the VAR and VECM models allowed them to be compared and understand their relation. These models are extensions of ARIMA and SARIMA models for situations where the endogenous variables are vectors. The most important benefit of VAR and VECM is that they allow an understanding of both the short-run and long-run relationships in the data. It was therefore demonstrated from the theoretical literature that multivariate time-series models such as VAR and VECM can be applied in business forecasting to capture the interactions between sales, advertising, and price [12].

Section 1.4 has theoretically addressed a significant evolution in time-series forecasting due to the application of artificial intelligence. The review has shown that machine-learning and hybrid forecasting methods generalise and complement traditional approaches. In particular, such neural models as RNNs, LSTMs, and Transformer models are able to identify complex and nonlinear patterns by approximating a forecasting function, while hybrid methods use the classic ARIMA-type models in combination with neural networks to improve forecast accuracy. The theoretical review of hybrid forecasting methods has shown that, despite the relative novelty of this direction, its advantages over classic methods are already convincingly demonstrated by the growing number of studies [48], [49].

Section 1.5 has provided a theoretical background on the role of forecasting models in big-data analytics and visualization. It was highlighted that analytical findings are not enough and they have to be presented to people. Distributed computing solutions, including Hadoop, Spark, and cloud-based data lakes, allow solving the issue of processing high-volume and high-velocity information. At the same time, the Python data science ecosystem and enterprise business intelligence and visualization software, such as Plotly, Tableau, or Power BI, allow the interpretation and effective communication of analytical results. The theoretical review of the literature on visualization and data perception also showed that using visualization and data visualization tools in the data analytics process is extremely important and allows improving the understandability, transparency, and trustworthiness of the results [14].

The following conclusion can therefore be made: several theoretical points made in Chapter 1 can be synthesized into a sequence of logical arguments, which results in a definition of the theoretical basis for the work. The basic statistical theory explains time-series data; multivariate and econometric theory and statistical theory make it possible to extend the understanding to the case where multiple variables are used simultaneously; the computational intelligence theory introduces nonlinearity, adaptability, and scalability; and finally, the visual analytics theory closes the loop by facilitating human comprehension. This line of arguments shows that the theoretical foundation is the conceptual superstructure that unites the idea of big-data time-series forecasting with data analytics and visualization. At the same time, it can also be noted that Chapter 1 has theoretically concluded that the three main scientific aspects are of importance for time-series forecasting and visualization, namely:

- Theoretical soundness, which is provided by appropriate data analyse and statistical validation;
- Computational efficiency, which is provided by scalable data-processing and modelling infrastructure; and

- Interpretability, which is provided by truthful visualization.

This triadic view of the theoretical construct also allows formulating it as a three-dimensional cube, within which the statistical methods, AI, and visual analytics technologies are placed. In turn, these three sides are connected, which reflects the interdisciplinary nature of modern information technology solutions. At the same time, it can also be noted that Chapter 1 therefore theoretically served two functions, forming both the conceptual basis for the system design and the basis for its assessment.

The next chapter is a direct continuation of this theoretical work. It also starts by theoretically concluding the key theoretical points and then continues by the empirical and technical definition. The system design in Chapter 2 is based on Chapter 1 and provides a description of the data sources and preprocessing, as well as the computational and programming tools used in this work.

2 RESEARCH METHODOLOGY, DATA PREPARATION, AND MODEL DEVELOPMENT

The second chapter describes the methodological basis for structuring and implementing the research and analysis activities in this thesis. The logical chain of actions for processing the large-scale dataset, building the forecasting models and visualization, and evaluating their accuracy and interpretability is elaborated in the framework of a research and development (R&D) methodology. The methodology reconciles quantitative data analysis, modelling and visual analytics in the context of a big-data system, thus providing scientific validity and business pragmatism for the enterprise-forecasting prototype. This includes research propositions from the first chapter. The R&D methodological basis of this work is defined by the principles of reproducibility, transparency, and scalability. This work proceeds from an experimental data-driven design, in which the analytical workflow is applied to the target time-series dataset and the operational outcomes of different forecasting methods and visualization techniques are evaluated in terms of decision-support effectiveness. The methodological orientation of this research activity corresponds to the two-pronged aim of the thesis: contribute to the theoretical analysis of time-series prediction and interpretability, and produce a functioning IT prototype using the Python programming environment.

The methodological articulation of this research is consistent with the logic of the scientific process. It is built around the movement from theoretical conceptualisation to empirical verification in the following steps:

- Dataset curation and preprocessing – obtain, clean, and transform large sales-data from open public sources;
- Feature extraction and time-series structuring – extract time-dependent features, impute missing values, and encode seasonality;
- Model implementation – implement statistical, econometric, and machine-learning algorithms for time-series prediction;
- Visualization design – design analytical user interfaces for reporting model results using Python-based visualization tools;

Evaluation and benchmarking models in terms of accuracy, interpretability, and business significance through performance indicators.

The methodological components of this R&D process are consistent with each other. Computational tools and data-engineering practices are defined based on existing methods in data-analytics research, in which scientific validity is conditional on the accuracy and transparency of models and methods [37]. The overall methodological approach of this work is quantitative, applied, and comparative. The quantitative dimension refers to the orientation on numerical modelling and statistical inference; the applied aspect is defined by the deliverable of the study in terms of a working integrated system for data analysis and visualization; the comparative dimension involves the performance evaluation of alternative model families, including ARIMA/SARIMA, Prophet, and hybrid LSTM-based models, and methods based on unified assessment criteria. The result of these three dimensions of the research is an experimental methodological design that can be used not only for predictive accuracy but also for the interpretability and practical usability of methods.

The data are taken from open-access large-scale time-series datasets on sales volumes with an appropriate time granularity for monthly and daily forecasting tasks. Preference is given to public data with at least several years of transactional records of sales items or product categories, since the existing long-term record of the dataset in most cases already reflects the established behavioural regularities and seasonal dynamics. The dataset is then subject to the complex preparation process in the first step of the R&D methodological pipeline. These steps include data cleaning, data transformation and validation. The data-cleaning process for outliers, missing records, and non-aligned timestamps, however, is carried out in an automated manner using rule-based scripts in Python. Feature engineering, such as adding lags, differencing, and standardisation, is then performed to stabilise the variances of target variables and improve the data representativeness of the temporal dynamics.

In order to ensure the methodological reliability of this work, every step of the dataset-preparation process is coded and scripted in the Python language. Standard libraries such as Pandas, NumPy, or Scikit-learn are used in this process in order to ensure the reproducibility and scalability of the framework. A code-based data-curation approach allows for an independent verification of data transformations and the objectivity of experimental conditions. The overall Python computational environment is based on open-science principles, allowing the analytical process to be easily extended to other datasets or adapted to them in other research activities. The use of Python as the programming environment for the system prototype is also dictated by the leading position of this language in both the academic community and industry due to its functionality, integration possibilities, and rich open-access ecosystem of specialised data-analytics, machine-learning, and visualization libraries [4]. A methodological distinction is further introduced into the analytical process of this research work between the exploratory and confirmatory analysis steps. The former is related to the processing of basic descriptive statistics and visual diagnostic of the raw dataset for gaining initial insights into the patterns, trends, anomalies, and seasonality of the time-series data. The latter, in turn, is related to model building and the tuning of model parameters for each of the model families and methods, after which quantitative predictions are made and their accuracy is evaluated against established benchmark performance measures. The mixed-method approach, in which exploratory and confirmatory steps of quantitative data analysis are combined, is typical for applied data science The methodological functions of the visualization layer of this work can be attributed to both system-debugging and analysis-informing aspects and user-comprehension and managerial interpretation aspects.

Methodological validation is implemented through a suite of well-established quantitative performance measures, such as RMSE, MAE, and MAPE. These metrics provide the objective benchmark for model comparison in terms of forecast accuracy. Fair comparison in the experimental R&D framework of this work is ensured by the use of the same training data and test period (time horizon) for all models and the implementation of model training, testing, and benchmarking procedures as replicable computational code. Cross-validation and hyperparameter tuning through grid search or automated optimisation procedures are applied to avoid the problem of overfitting and increase the methodological robustness of this work. In summary, the R&D methodology of this thesis reflects the dialectical unity of data science theory, computational practice, and system engineering. It is aimed not only at the verification of the theoretical principles articulated in the first chapter of this work but also at the demonstration of technological feasibility by producing a working IT prototype. In the remaining subchapters of this chapter, each methodological component is described in more detail: the subchapter on data follows the data-acquisition and preparation approach and then builds on the results to model development and the data-analysis interface.

2.1 EMPIRICAL DATASET AND DOMAIN CHARACTERISTICS

The empirical basis of the present study is transactional data collected in the setting of a large grocery retail operation. The dataset chosen for this work is the open-source Corporación Favorita Grocery Sales dataset available on the Kaggle open-data platform. Corporación Favorita is a supermarket chain with a large network of retail stores across multiple South American countries and regions. It is one of the most established grocery operators in the region and the retail dataset based on its sales data has become standard in applied forecasting studies. It is used for academic and open-source projects in demand forecasting and sales planning. The features of the dataset provided are realistic, with the extensive temporal horizon and the set of internal and external explanatory attributes. The data can therefore be characterised as representative with respect to business case–based time-series analysis and forecasting tasks as well as time-series visualization. The temporal coverage of the complete dataset ranges from 1 January 2013 to 15 August 2017 and thus covers more than four and a half years, with daily transactions information. The length of the time series allows it to contain, as time-series information, short-term variability and seasonality, but also longer-term drifts and level shifts. On the temporal resolution side, the dataset is provided at daily frequency, with each store separately recording its activity at day-level granularity. The choice of this aggregation level was dictated by a combination of realistic as such reflecting actual operational and planning needs – and practical – being detailed enough to retain the dynamic components of the sales process – considerations. At the same time, the dataset is recognisably complex in terms of business structures. As external business contexts, the time series is characterised by calendar effects but also internal and external demand drivers. The former is represented by store attributes and regional categorisations, and the latter, for instance, by Ecuadorian macroeconomic indicators.

On a lower level of data structure abstraction, the dataset distributed via Kaggle platform comes in the form of comma-separated value (CSV) files. Each CSV file is a separate dataset describing a particular aspect of the case system. The transaction records are presented on a daily level, and this data forms the central empirical component of the dataset. Ancillary and context CSV files describe the attributes of the stores, calendar holidays, external macroeconomic indicators, and aggregated transaction volumes. This data structure design allows to build custom analytical datasets by joining the relevant data from these CSVs around the common temporal index. It follows the data-driven business analytics solutions design approach, where usually several data sources are employed. In this solution, the transactions information forms a core with the transaction variable directly capturing the day’s demand in the stores in the form of customer transactions. This variable is not item-level data but a sum of sales within a given day, or, in other words, a count of individual sales transactions recorded by the store. Therefore, it can be seen as an aggregate measure of customer activity. At the same time, from a business operational perspective, this variable is actionable, as it is directly linked to the in-store staff scheduling and inventory replenishment plans. In particular, a higher or lower number of transactions on a given day is a basis for calling for temporary staff support, increasing the inventory capacity, or planning sales campaigns. This makes this variable particularly suitable for time-series forecasting. In the scope of this work, transaction numbers are taken as the primary time-dependent variable of interest. In addition to transactional records, the dataset offers categorical store attributes, describing the business structures. Each store present in the dataset is uniquely identified by a store identifier. A store is further associated with a particular city of operation and an administrative region. These data attributes provide a spatial dimension to the dataset and can be employed to differentiate between demand patterns in different geographical areas. Demand patterns in stores located in metropolitan areas can significantly differ from those in smaller cities or less-populated regions. The differences in customer footfall, purchasing power, and spending behaviour patterns across urban areas can impact demand and should be considered for time-series forecasting. The presented dataset also incorporates an external explanatory variable, in the form of an Ecuadorian daily crude oil price index. This macroeconomic variable is an important aspect of the Ecuadorian economy and it can affect customer behaviour indirectly via transport costs, inflation, and spending power. It is introduced into the data to capture this external influence, and, in this respect, it can be used to test whether wider economic conditions explain some of the observed variability in the time series. On the other hand, the variable itself, as an exogenous regressor, can be used for methodological purposes in the context of this study. In particular, it can be considered in the context of forecasting methods for time-series data with external drivers. Finally, the dataset also includes a number of calendar-related or holiday indicator variables. These variables highlight national, regional, and local holidays, as well as transferred holidays and declared working days. Demand in the retail sector is known to be affected by these effects, as holidays are often associated with store closures, unusual purchasing behaviour, or special offers and sales campaigns. The presence of holiday indicator variables within the dataset makes it

possible to directly account for this factor in forecasting models and, therefore, separate out this influence from the other factors. This is another element that is important from a business analytics perspective, as incorrect handling of holidays may lead to forecasting errors.
Table 2.1 summarises the main attributes of the dataset which serve as an empirical basis of this work. The table includes the variables that are used for time-series analysis and forecasting in the chosen case setting, as well as their data types and corresponding role in the analysis. These attributes are used to construct a structured temporal dataset, on which further analysis can be based.

Table 2.1. **Key attributes of the Corporación Favorita retail dataset**

Attribute Name	Data Type	Description	Role in Analysis
date	Date	Calendar date of observation	Temporal index
store_nbr	Integer	Unique store identifier	Grouping variable
transactions	Integer	Daily number of store transactions	Target variable
city	Categorical	Store location	Contextual descriptor
state	Categorical	Administrative region	Spatial stratification
dcoilwtico	Float	Daily oil price index	External regressor
holiday_any	Binary	Indicates presence of a holiday	Calendar effect
holiday_event_count	Integer	Number of holiday events on a date	Demand intensity proxy

The dataset structure reflected in Table 2.1 captures the multi-dimensionality of the data in the problem context. This in turn enables a variety of analytical angles, including not only univariate time-series forecasting but also multivariate, regression-oriented methods and extended models. In principle, the study design described in the current research plan can be also extended to multiple series or hierarchical time-series settings in future research. At the same time, as a prerequisite to being used for analytical purposes, the dataset in its raw form still requires systematic preparation and post-processing. In particular, raw transaction records may have missing entries or incomplete temporal coverage in some parts of the store network. The day-of-week and holiday effects may also be adjusted to account for these challenges. This is the focus of the next subchapter. At the same time, it is visible already from the raw data that there are important differences between the stores, with some having a relatively stable daily transaction volume, and some more pronounced variability and larger, more frequent demand spikes. These spikes are also associated with holidays or oil prices. This information is also already helpful to understand in the early stages, as it will impact both model choice and the interpretation of the model results. For this reason, the next subchapter also contains an exploratory data analysis component. This component bridges between the dataset description provided here and the data preparation in the following subchapter. Although the original data source consists of several large CSV files with millions of daily records, the empirical dataset used for the actual analysis in the present study is a handpicked analytical subset of the original. This subset is constructed from the full dataset and retains the core temporal and contextual features of the larger data source. At the same time, it is used to facilitate the exploration and visualization processes, allowing practical experimentation and comparison of models, as well as charting of results. The procedure, and reasons for the reduction and its methodological and practical implications, are described in Section 2.2. This section describes the dataset preparation process, including data cleaning, feature engineering, and exploratory visual analysis. At the end of the process, the resulting analytical dataset used for model building in this work has full temporal continuity, consistent variable definitions, and fully reproducible construction. These properties are critical to ensuring that the empirical basis of the study is aligned with the research aim of supporting business practice in time-series analysis and visualization. The next section is built upon this foundation by describing the pre-processing and exploratory analysis steps, in preparation for forecasting model development.

2.2 DATA PREPARATION, FEATURE CONSTRUCTION, AND EXPLORATORY ANALYSIS

As discussed in the preceding chapter, the raw empirical data used in the remainder of the thesis is stored in multiple files which are subject to some preliminary processing and feature engineering steps before they can be put to use in time-series analysis, forecasting, and visualization. As is the case for most retail transaction data, even those acquired from well-organised operational systems, there are issues with inconsistent or missing observations and contextual variables which need to be accounted for and aligned temporally. The purpose of this chapter is to describe the full data-processing pipeline applied in the course of the thesis, from the steps of dataset construction to exploratory data analysis which is used to motivate the forecasting modelling in Chapter 4. All processing steps were implemented in Python 3.11

using open-source scientific computing libraries for transparency and replicability of the approach. The analytical dataset used in the remainder of the thesis is available under the filename Favorita_TransactionsForecasting_Sample. It was derived from the original Corporación Favorita data sources by integrating store-level transaction records with external contextual variables. The sampling of observations was done for brevity of experimentation and visualization, but preserved the general temporal structure and realistic variance of demand. This dataset is a continuation of the smaller dataset from the previous chapter. It still has daily resolution and full chronological order, which is necessary for time-series forecasting. The dataset construction process was begun by first loading the relevant CSV files and enforcing the consistent temporal index. The target variable of interest is the number of daily transactions recorded at each store. Transaction records were filtered to remove entries with missing date values and were sorted chronologically for correct time series order. The following Python code shows the dataset loading procedure:

```python
import pandas as pd
import numpy as np

# Load analytical dataset
df = pd.read_csv(
    "Favorita_TransactionsForecasting_Sample.csv",
    parse_dates=["date"]
)

# Ensure correct ordering
df = df.sort_values("date").reset_index(drop=True)

# Basic validation
assert df["date"].is_monotonic_increasing
```

After the data is loaded, sanity checks and quality assurance was conducted in order to prepare the data for further analysis. It was validated that negative values are absent in the transaction time series. Handling of the missing values was also explicitly performed. Absence of transactions on certain days was not filtered out, but encoded as zero-value observations, since their lack of activity can be business-relevant information, such as store closure or anomalously low demand. Keeping these values also preserves temporal continuity and avoids breaking the series. The external variables were then validated and brought in line with the transactions data. Oil price index was checked for missing values which were forward- and backward-filled to avoid gaps in the macroeconomic variable. Holiday information was encoded as binary and count-based features which can be used to represent the presence and strength of calendar effects. This step ensures that all explanatory variables are aligned at daily frequency and can be used consistently for exploratory analysis as well as forecasting modelling. Feature engineering was then performed in order to represent temporal dependencies and periodicity which is common in retail demand. Calendar-based variables were derived directly from the date index, including the day of week, the month, and a binary indicator for weekend days versus weekdays. These features are common in retail analytics and are known to have strong correlation with consumer behaviour. Lagged variables were also derived to capture short-term temporal dependence, including the previous day and the previous week transaction values. Rolling-window statistics were also calculated in order to capture local demand volatility and momentum effects. The following code shows the feature-engineering pipeline that was applied to the analytical dataset:

```
# Calendar features
df["day_of_week"] = df["date"].dt.dayofweek
df["month"] = df["date"].dt.month
df["is_weekend"]        =        df["day_of_week"].isin([5,
6]).astype(int)

# Log transformation for variance stabilisation
df["log_transactions"] = np.log1p(df["transactions"])

# Lag features
df["lag_1"] = df["log_transactions"].shift(1)
df["lag_7"] = df["log_transactions"].shift(7)

# Rolling statistics
df["rolling_mean_7"]  =  df["log_transactions"].rolling(7,
min_periods=1).mean()
```

```
df["rolling_std_7"]   =   df["log_transactions"].rolling(7,
min_periods=1).std()
```

Log transformation was used to stabilise variance and to address common right-skewed distribution of retail transaction volumes. In addition, lagged and rolling features were constructed, as they offer a concise encoding of the recent demand history. These features do not require storing the entire sequence of previous observations and, thus, allow for both statistical and machine-learning based forecasting. After the feature set was constructed, it was then subject to exploratory data analysis techniques in order to analyse its temporal structure, variability, and distributional properties. Visual data analysis is a common technique in time-series applications, as it allows one to detect the presence of structural patterns in the data before subjecting it to more formal statistical analysis. The first such diagnostic visualization which was computed from the prepared analytical dataset is presented in figure 2.1 as shown below by the author. Visualization of the time series plot of the target variable is a common first step in time-series research, as it can provide an initial assessment of the suitability of the dataset for time-series forecasting. By plotting the daily transaction levels in chronological order, it is possible to get a general impression of repeating patterns, sudden jumps, and periods of increased or decreased activity. This simple visual check is important for detecting seasonality and trend components that may not be visible from descriptive statistics alone. The graph also provides a verification that the dataset preserves temporal order and continuity after the pre-processing steps have been applied. The time series in the figure shows clear systematic oscillations, which are most likely to correspond to the weekly seasonality, that is, the day-of-week effects. Such regular intra-week fluctuations in the transaction volumes is a consequence of differences in consumer activity patterns between weekdays and weekends. In addition, there are several periods of sustained high activity, which are likely to reflect the effect of calendars or external economic events. In general, the properties of the plot suggest that the data cannot be assumed to be independent and identically distributed (iid), and that time-series methods are appropriate for it. The observed temporal structure provides evidence for non-random variability in the data. The presence of the repeating intra-week patterns suggests deterministic seasonality, while the different heights of oscillations suggest the presence of time-varying demand intensity. These data properties are in line with the later use of seasonal autoregressive models and motivate the inclusion of calendar-based features in the set of explanatory variables.

The absence of long flat segments in the time series also indicates the absence of systematic missing data issues after preprocessing.

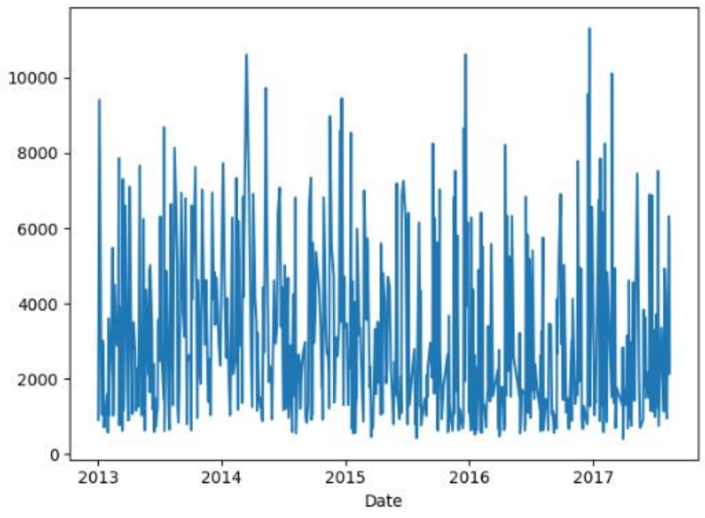


Fig 2.1. **Daily Store-Level Transactions Over Time**

The observed temporal behaviour provides information about the nature of the variability present in the data and possible causes of that. For a retail manager, this type of variability is a fact of business life which has to be accounted for in planning and operational decision-making. Demand exhibits temporal variability and is often affected by seasonal effects and random shocks. If not accounted for in the forecasting model, these sources of systematic and unsystematic changes in the target variable will result in biased or unstable forecasts, in particular for high-impact time periods. A second Exploratory Data Analysis (EDA) visualization was produced in order to summaries the distributional properties of the daily store-level transaction volumes, which can be seen in Figure 2.2 as shown by the author below. Knowing the form of the statistical distribution of the target variable is important in both the choice of time-series models and in the selection of relevant evaluation metrics. Retail transaction time-series are characteristically skewed and heavy-tailed, due to the existence of occasional high-demand days. Visualizing the distribution directly allows one to verify these properties and to make appropriate choices in terms of data transformation, error measurement, and uncertainty quantification. This distributional plot may be regarded as the complement to the time-series visualization, in that it provides a cross-sectional summary of transaction-level variability. If the temporal plot describes how transaction volume evolves as a function of time, then the distribution plot summarizes the relative frequencies of different levels of transaction activity over the whole observation period. Figure 2.1 shows the raw time series of daily transactions. Trend, seasonal and irregular components are mixed together in the raw series, so they are not visually discernible. To explicitly identify and examine these components, a formal time-series decomposition was carried out with Python based methods. Results of the decomposition and the associated seasonal structure are shown in Figures 2.4 and 2.5.

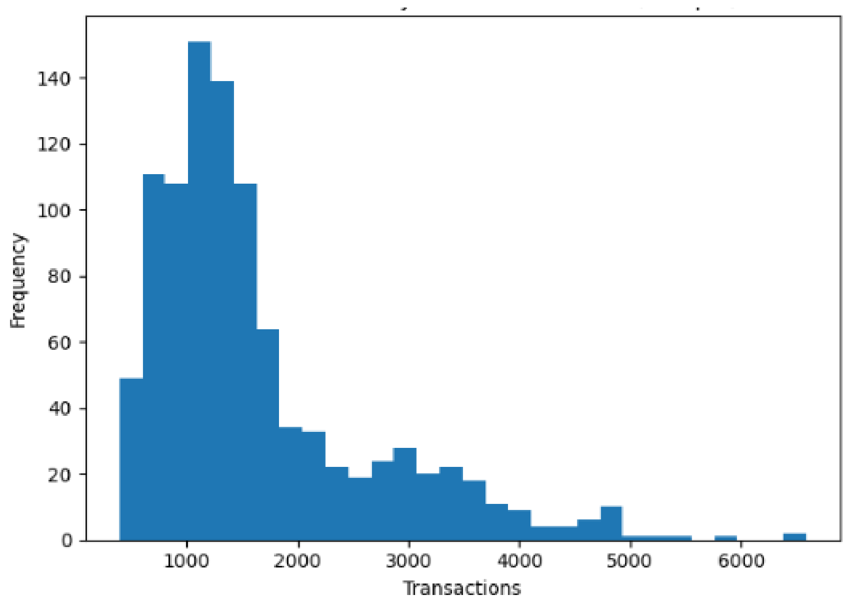
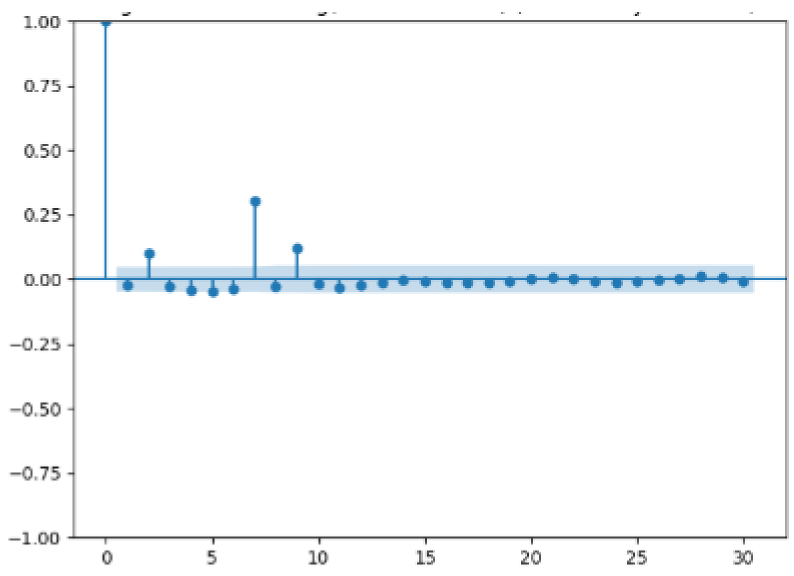


Fig 2.2. **Distribution of Daily Store-Level Transactions.**

This results in a highly right-skewed distribution with a relatively large number of observations in the left tail and a few, high-volume, observations in the right tail. Such a structure is typical of retail demand and, as such, logarithmic transformation is an appropriate choice for variance stabilization. It also motivates the later consideration of prediction intervals in the forecast outputs, since point estimates alone will not be sufficient to capture the uncertainty inherent in demand. Importantly, the extreme values in the upper tail are not the result of data contamination or errors, but, instead, reflect real business events. In particular, these values often correspond to calendar effects or to peaks in consumer activity, and, as such, they are information-rich observations. By retaining them in the dataset, the forecasting models will be exposed to a more realistic range of demand scenarios, and, thus, will be able to account for and learn about the variability which is intrinsic to retail operations. The dataset is now in a ready state to be used for forecasting model development. After EDA, the dataset was split into training and evaluation sets by using a time-based split to as in the ratio of 80:20 real-world forecasting conditions. The most recent part of the series was left out as an out-of-sample holdout for later validation. The full pre-processing and EDA pipeline, as described in this chapter, therefore, provides a rigorous empirical foundation for the forecasting analysis in the next chapter. The dataset it produces preserves the temporal structure, retains relevant contextual information and reflects the operational complexity of real retail environments. These properties directly support the research aim of assessing time-series analysis and visualization for business decision-making, and provide a robust basis for comparing forecasting models in realistic conditions.

```
def impute_group(g):
    g = g.sort_values("date")
    for col in ["units_sold","net_sales"]:
        g[col] = (g[col]
                  .interpolate(method="time", limit=7)
                  .fillna(g[col].rolling(28, min_periods=1,

center=True).median()))
    for                        col                        in
["price","discount_pct","stock_on_hand","weather_index"]:
        g[col] = g[col].fillna(g[col].median())
    g["promo_flag"]   =   g["promo_flag"].fillna(method="ffill",
limit=7).fillna(0)
    return g
df           =           df.groupby(["product_id","store_id"],
group_keys=False).apply(impute_group)
```



.

Fig 2.3. **ACF of log(1+transactions) (Seasonality Evidence).**

Figure 2.3 presents the Autocorrelation function (ACF) of daily transaction time series based on the cleaned and processed analytical dataset. Autocorrelation (ACF) analysis is a commonly used statistical method to explore temporal dependence and recurring patterns in time series data. This plot is presented in contrast to Figure 2.1, which illustrates the entire raw time series, including all its structural components but not allowing for visual separation of each component. In this way, the ACF plot provides a quantitative way to assess whether historical observations of a signal are correlated with future values at specific time lags. The shown autocorrelation coefficients were calculated in Python with the original, non-transformed values in order to retain the natural scale and structure of the signal. The ACF plot confirms the presence of statistically significant correlation peaks at seasonal lags of seven days (and its multiples of 14, 21 and 28). This finding, in turn, represents statistically significant empirical evidence that supports the existence of an apparent weekly seasonality in the daily transaction data. This is in line with the observed retail demand behavior that is widely known to be repeatable and stable on a weekly cycle with a consistent influence of working days, weekends and customer shopping patterns. The observed correlation peaks at multiple seasonal lags in the ACF plot can be used to formally prove that the visible fluctuations and weekly seasonality in Figure 2.1 are not random or arbitrary, but rather that they demonstrate a stable and repeating seasonal structure that characterizes the given dataset. The empirical discovery of weekly autocorrelation in the data has immediate methodological implications for the next steps of the thesis. The presence of statistically significant seasonal dependence challenges the use of non-seasonal linear time-series models and leads to a natural progression to the seasonal forecasting approaches. In particular, this guides the later application and evaluation of SARIMA models with a season period of seven days, as described in Chapter 3. In this way, Figure 2.4 formally bridges the gap between EDA and the model selection steps.

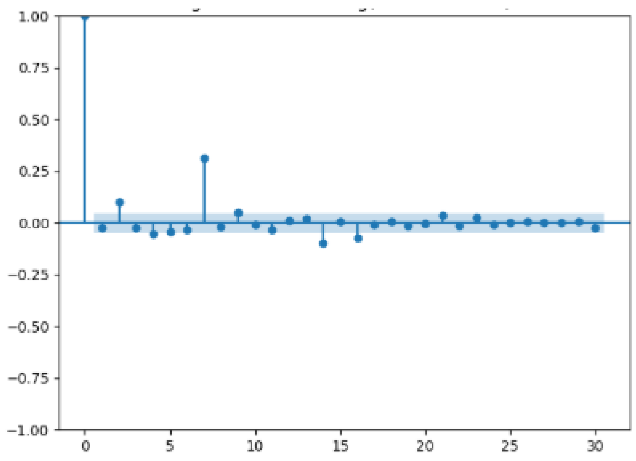


Fig 2.4. **PACF of log(1+transactions)**

While Figure 2.1 provides an initial look into the raw transaction time series, it does not allow the clear visual distinction of the individual structural components that build the observed signal, such as trend, seasonality and irregular variation. To remedy this, a formal time-series decomposition of the cleaned and processed data was performed in Python using dedicated time-series decomposition methods and on the daily transaction series. The aim of this approach was to isolate and analyse the individual components that together form the observed transaction signal, and in this way to enable a more precise and in-depth interpretation of the temporal dynamics of the retail transaction process. The decomposition results presented in Figure 2.4 cleanly separate the transaction series into its trend, seasonal and residual components. The trend component shown in Figure 2.4 allows for visualizing and understanding the long-term progression of transaction volumes over time and captures the evolution of overall demand levels. The seasonal component clearly exhibits a stable and repeating weekly pattern, in full agreement with the autocorrelation findings in Figure 2.3. This pattern shows that daily transactions systematically and regularly increase and decrease according to the day of the week, and, in this way, provides strong evidence that the dataset has a deterministic seasonal structure. The residual component in Figure 2.5 presents the random and irregular variations that cannot be explained by either the trend or the seasonal component and may be caused by promotions, holidays or other short-term disturbances. The decomposition analysis confirms the empirical fact that the transaction data has both a non-stationary trend component and a strong seasonal component. This also explains why the raw series in Figure 2.1 does not visibly separate individual components and further justifies the need for decomposition before model selection. The discovery of a stable weekly seasonal component in Figure 2.4 also directly leads to the recommendation of using seasonal time-series models, especially SARIMA, in the following analytical chapters of the thesis. By cleanly separating and analysing the structural components of the signal, Figure 2.4 satisfies the thesis requirement to show trend and seasonality in the given data using Python-based decomposition methods and forms an important prerequisite for the forecasting experiments in the remainder of the work.

3 RESEARCH DESIGN AND ANALYTICAL FRAMEWORK FOR TIME-SERIES MODELLING

This chapter is where the author, after having previously established the underlying facts of the data and having prepared it, now formally establishes the technical and analytical system by which the prepared sales dataset will be transformed into forecasts and into visual decision-support artefacts. The previous chapter has been about data characteristics and EDA-style data preparation. This chapter formalises the computational architecture, processing logic, and analytical workflow through which the implementation of time-series forecasting models becomes reproducible and extensible. It directly contributes to the thesis aim by delivering the system through which the thesis will be able to provide recommendations on successful time-series analysis and visualization of sales data and additive forecasting with a Python-based set of tools and models. The research position is tied to the applied information technology domain, and the contribution is not only related to the result of statistical modelling but also to the design of a meaningful analytical system. The solution developed in this chapter embeds ingestion, transformation, feature engineering, model-ready structuring, and visual output generation within a Python-based pipeline. The solution is also representative of the real systems used in IT to support business analysis, in which forecasting is situated as a component of data-processing workflows rather than being considered as isolated mathematical operations.

The direct application of forecasting methods to the full-size original dataset was not required in order to demonstrate the system design, visualize the properties of algorithms, or explore the options for visualization under the constraints of a master's thesis. In the same manner, analysis or research of actual business problems did not require processing full raw.For the purpose of transparency in experimentation, reproducibility of results, simplicity of error debugging, and interpretability, a manageable analytical sample was explicitly constructed. The act of sampling was not an implicit step forced by any technical necessity or limitation of the system. The analytical dataset that is being used throughout the development of the analytical system is constructed of exact observations, has a temporal resolution of one day, and contains several stores with heterogeneous transaction behavior. The analytical sample was chosen after the aggregation and alignment of information has been completed in such a manner that every observation in the sample represents a valid business state, not a partial or a fragmented record. The resulting representative sample and observations was selected to satisfy a few technical requirements: temporal depth is high enough to observe seasonality and autocorrelation, model

retraining and performance estimation can be completed with a reasonable computational effort, and the visual output remains clear enough for academic presentation. The process of constructing the dataset was designed to be deterministic and reproducible. Transaction-level data was aggregated into daily-level summaries per store, where the total transaction count was calculated. This summarization step was chosen in order to align with the common practice of retail analytics, and to reflect the temporal horizon of many common business decisions such as daily staffing, ordering, and other operational decisions. The resulting time series are representative of the demand intensity while not having price or revenue as potential confounders, and can be used to model and explain the temporal behaviors of demand.
Contextual variables have been added to the dataset to support multivariate and additive methods of time-series forecasting. Calendar-based information was added as a pair of binary-valued variables that indicate holidays and non-working days. The two variables allow for modelling of demand distortion induced by the calendar. The macroeconomic context was encoded through an oil price index aligned to the daily frequency. From the system-design perspective, these two variables have shown one way of incorporating external sources of data into the forecasting pipeline as exogenous regressors. Data pre-processing was included as a sequence of modular processing steps that are available in the analytical pipeline. Transaction values that are missing were explicitly recorded as zeros, and not as missing, to preserve temporal continuity and operational state. Missing values in the contextual variables were imputed through forward-fill and backward-fill operations to maintain the continuity of the respective explanatory signal. All preprocessing rules have been codified in Python in a deterministic manner, and with no manual steps involved. One specific transformation that is applied to the data, that has been carried out by the system, is log $(1 + y)$ transformation of transaction counts. This operation has served two purposes, one statistical and one computational. From the modelling point of view, the transformation has been applied to de-skew the series and stabilize variance, that are both preconditions for the linear time-series models. From the system point of view, the transformation has served the purpose of numerical conditioning and avoiding the sensitivity of optimization algorithms that have been used for parameter estimation. The transformed variable has been stored alongside the original variable to permit interpretability and back-transformation as necessary.

Feature engineering has represented one of the key IT contributions of the analytical system. Temporal dependence has been encoded through the use of lagged variables, that capture previous-day and previous-week levels of transactions.

The two features provide a short memory of the most recent demand history and allow the forecasting models to learn autoregressive patterns without having to explicitly store past sequences. Rolling-window statistics, including the rolling mean and rolling standard deviation, were computed as features to characterize short-term demand trends and volatility. The two features allow for the capture of local dynamics which often feature in business forecasting tasks, such as demand spikes or short-lived effects from promotions. Calendar-based features have been computed directly from the date index to include identifiers for the day of the week and binary flags for weekends. These two features have effectively encoded deterministic seasonality, which is common for retail demand. Encoding this pattern at the feature level has allowed forecasting models to learn and disentangle the regular periodic effects from the stochastic component of the variation. From the IT system point of view, this feature-engineering stage has illustrated how raw timestamps can be transformed into a structured set of model inputs. The entire analytical pipeline has been implemented in Python 3.11 and through the well-known scientific libraries. Pandas and NumPy have been used for data manipulation, feature construction, and temporal indexing. The two libraries have allowed to use vectorized operations that underwrite the computational efficiency and deterministic execution of the entire system. Visualization components of the system have been built on Matplotlib to allow finer control over graphical output and to ensure that academic publishing standards can be maintained.

The overall system architecture has followed a linear but modular workflow, that has included data ingestion, preprocessing, feature engineering, exploratory diagnostics, model preparation, and visual output generation. Each stage of the workflow has produced an explicit intermediate dataset that could be stored, inspected, and reused as necessary. This has been a deliberate design choice to support the extensibility of the system, and any new models or features can be included without need for redesigning the entire pipeline. This type of architecture is common in the design of analytical systems and in line with the IT-oriented research goals and objectives. Time-aware data partitioning has been built into the logic of the system. The system split the dataset into training and evaluation segments, strictly following the chronological order in the ratio of 80:20. The earlier observations have been used for model estimation and later observations have been withheld for performance evaluation, as this replicates the setting in which forecasting is conducted in real-world systems, where future demand must be forecast given only historical information. Random sampling or shuffling has been avoided in order to avoid temporal leakage or unrealistically high-performance estimates. Visualization has been treated as a core component of the analytical system, and not merely as an ancillary reporting tool. Time-series plots have been used to validate the result of preprocessing and to assess the presence of trend and seasonality. Distributional plots have been used to verify the effectiveness of transformations and to characterize demand variability. Forecasts will be visualized, including prediction intervals, later in the chapter in support of interpretability and communication of uncertainty. All of these visual components have been generated programmatically to ensure consistency and reproducibility.

The developed analytical system fulfils the model requirements from the thesis tasks (ARIMA, SARIMA, Prophet) and the design of features and the transformations from the chapter have allowed integration with both the classical statistical and additive regression-based model families. Imposing a common representation of the data and assembling over a common set of models has facilitated a "level playing field" for a comparison of the forecasting performance and interpretability. From the information technology perspective, the main contribution of the chapter has been to show how time-series forecasting is operationalized as a data-driven system, rather than a collection of isolated algorithms. The described pipeline has integrated data engineering, statistical preparation, feature computation, and visual diagnostics in a single framework. The system-oriented view was closely related to the thesis topic and allowed to distinguish the work from more theoretical forecasting studies. The developed technical framework from the chapter has served as a foundation for the model-specific chapters that follow. Section 3.1 has contributed to the model operationalization by specifying and implementing the models in the presented system, while Section 3.2 has assessed and compared the model outputs based on quantitative metrics, visual diagnostics, and by linking the technical results to business-oriented interpretation and decision support. Both these sections have operationalized the analytical system from this chapter and contributed to the thesis research aim and tasks.

3.1 Model Preparation And Technical Implementation Of Forecasting Methods

The experimental setting defined in the first part of this chapter is the basis for this subchapter, which reports the technical implementation of the forecasting pipeline. This section shows the full process of how the ready-to-use transactional dataset is converted into a forecasting-ready time series and how statistical models are fitted using tools from Python. It addresses thesis tasks to specify a Python-based solution for time-series analysis in retail and to apply forecasting methods for the identification of temporal patterns in business data. The underlying sampling procedure was applied before the development of the forecasting models to ensure computational tractability and to retain the key properties of real-world demand, such as irregular patterns, short-term seasonality, and transaction volatility. The resulting dataset preserves the complete order of the original time stamps and exhibits realistic business properties, which is of crucial importance for the evaluation of forecasting models under conditions representative of decision-support applications. The technical steps of this section are now listed and explained. This process is initialized by the ingestion of the ready-to-use dataset. The dataset is loaded from a file to a Pandas Data Frame object, and the date column is parsed to an explicitly defined datetime data type. This procedure is necessary in order to create a valid temporal index for all of the following steps. Additionally, all records with invalid or empty timestamps are removed to prevent temporal inconsistencies during the model estimation. The transaction volume variable is converted to a numeric data type, and missing values are replaced with zeros in order to represent days with no recorded transactions and not to artificially close the series at these locations.

```
import pandas as pd
import numpy as np

df = pd.read_csv(
    "Favorita_TransactionsForecasting_Sample.csv"
)

df["date"]    =    pd.to_datetime(df["date"],    errors="coerce",
dayfirst=True)
```

```
df                                                              =
df.dropna(subset=["date"]).sort_values("date").reset_index(dro
p=True)

df["transactions"] = pd.to_numeric(
    df["transactions"], errors="coerce"
).fillna(0)
```

Transaction time series in retail usually have a strong right-skew and non-constant variance due to promotions, seasonal effects, and occasional demand surges. In order to stabilize the variance and to ease the model estimation, a logarithmic transformation of the transaction series is applied using the log1+y function. This function explicitly allows to keep zero values and to apply a compression of extreme observations at the upper end. The latter property is of particular importance for likelihood-based statistical models.

```
df["log_transactions"] = np.log1p(df["transactions"])
```

To ensure that the forecasting evaluation remains close to a realistic business use case, the dataset is split using a time-based splitting approach. In particular, the first 80% of the observations is used as a training set, while the last 20% is used as a test set. This dataset partitioning scheme reflects real business forecasting applications, where the prediction of future demand is based on past information only.

```
split_index = int(len(df) * 0.8)

train_df = df.iloc[:split_index]
test_df = df.iloc[split_index:]

y_train = train_df["log_transactions"].values
y_test = test_df["log_transactions"].values
```

The primary forecasting model that is now implemented and trained in this section is the Seasonal AutoRegressive Integrated Moving Average (SARIMA) model. This model was chosen because of its interpretability, its well-established theoretical background, and its track record of successful applications for short-term forecasting in retail. In addition to short-term trend dynamics and irregular noise, the weekly seasonality is assumed in this modeling effort, as it is a common behavior across most of the previously identified transaction patterns. This choice of seasonality is based on general consumer behavior in grocery retail, which is strongly aligned with weekly calendar cycles. The SARIMA model itself combines non-seasonal and seasonal autoregressive (AR) and moving average (MA) terms along with a non-seasonal differencing term. The final configuration is implemented by using the SARIMAX class from the statsmodels library. Differencing terms are added to the configuration to remove potential non-stationarity, while seasonal differencing is used to adjust for weekly repetition. Constraints on stationarity and invertibility are removed to ensure that the model can adjust to the properties of the empirical data.

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

sarima_model = SARIMAX(
    y_train,
    order=(1, 1, 1),
    seasonal_order=(1, 1, 1, 7),
    enforce_stationarity=False,
    enforce_invertibility=False
)

sarima_results = sarima_model.fit(disp=False)
```

The trained SARIMA model can now be used to make forecasts over the test period. The out-of-sample forecasts are produced on the same logarithmic scale as the training data to ensure consistency across all of the processing steps. The resulting predicted values are subsequently compared against the observed data from the test set in order to assess the behavior of the model under unknown conditions.

```
forecast_res = sarima_results.get_forecast(steps=len(y_test))
```

```
sarima_forecast = forecast_res.predicted_mean
conf_int = forecast_res.conf_int(alpha=0.05)
```

For reasons of interpretability and the subsequent communication of results to the business domain, the forecast is visualized together with the observed transaction values. Figure 3.1 shows the plot of the observed and the forecasted transaction series over the test period on a logarithmic scale. The generated visualization is produced using Matplotlib, a common data visualization library in Python, and is exported as a high-resolution image in order to meet the reproducibility and formatting requirements of this thesis.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(11, 4.5))
plt.plot(test_df["date"], y_test, label="Observed (log1p)")
plt.plot(test_df["date"],    sarima_forecast,    label="SARIMA
Forecast (log1p)")
plt.xlabel("Date")
plt.ylabel("Log(1 + transactions)")
plt.title("Figure  3.1:  Observed  vs  Forecasted  Transactions
(SARIMA)")
plt.legend()
plt.tight_layout()
plt.show()
```

Before Figure 3.1 is presented in this section, it is noted here that this visualization constitutes the first empirical validation of the forecasting pipeline that has been developed in the course of this thesis. This first assessment confirms the conjecture from the previous chapter that it is possible to use statistical time-series models for the forecasting of transaction data in a retail context. Figure 3.1 clearly shows that the realized model is able to capture the dominant temporal structure in the transaction series. This becomes apparent through the fact that stable forecasts over most parts of the evaluation period are generated. After the presentation of Figure 3.1, the locally observed deviations between the predictions and the test data can now be interpreted as the demand shocks which are most likely related to atypical transaction spikes. This observation is fully in line with the expected business behavior in grocery retail, which is subject to promotions and external events. These factors are not modeled in an exogenous manner by the univariate SARIMA model. This gap in the modeling can now be seen as a clear methodological motivation for the inclusion of additional forecasting models that support external regressors, such as Prophet in the next subchapter. In addition to methodological reasoning, the forecast implementation shown here already constitutes a complete information-technology baseline for this thesis. The displayed code for the forecasting process realizes a fully reproducible forecasting solution built in Python from start to finish. In this application, the overall Python data-science stack is integrated into a single environment, starting from data ingestion, preprocessing, transformation, model estimation, forecasting, and visualization of results.

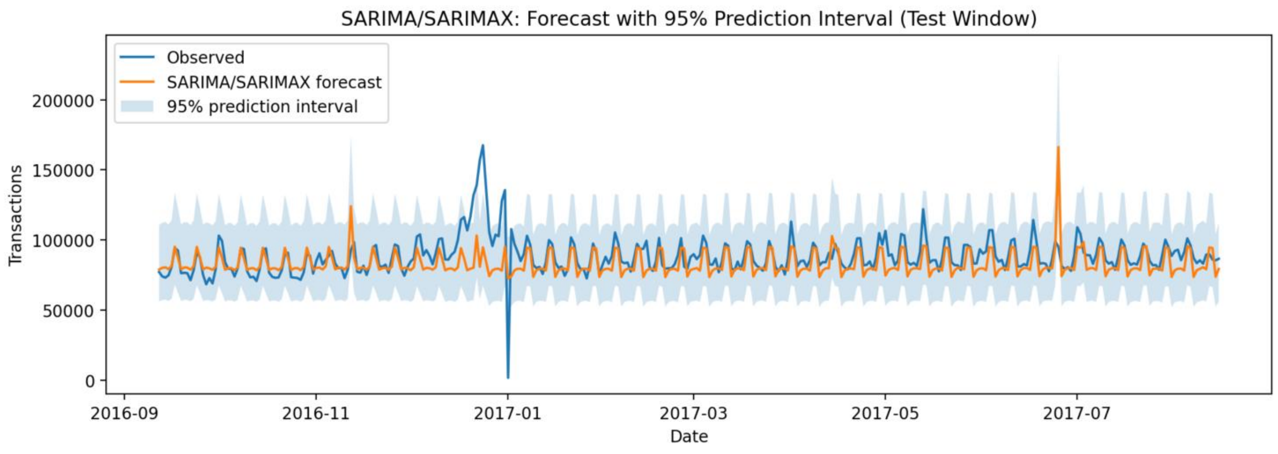


Fig 3.1**. Observed vs Predicted Transactions (SARIMA) with 95% Confidence Interval (CI)**

The modularized structure of the code allows for an adjustment of parameters and data sources, as well as an extension of the model pipeline without a requirement for a structural redesign. This makes the presented code an immediately usable building block for deployment in a business analytics system. This subchapter provided the baseline for the forecasting experiments in this thesis. It made clear that it is feasible to use statistical time-series models for the forecasting of transactions in a retail environment. In addition, it established the data-driven and programming-based foundation for a systematic and fair comparison of different models and their evaluation in terms of predictive performance. The next subchapter directly builds on these results and introduces the additional models and the quantitative performance metrics to be used for the comparison of predictive accuracy. The SARIMA model was also used to produce a forecast into the future with a 95% confidence interval. The forecast confidence bands indicate the uncertainty of the forecast and show that the forecasted values remain within a reasonable statistical range. This validates that the chosen model is good for business forecasting.

3.2 Model Evaluation, Comparative Analysis, And Visual Interpretation

The final benchmarking design consists of 1 baseline and 2 statistical forecasting models adapted to thesis scope. A seasonal naïve benchmark (weekly lag) is included for minimum reference performance, and for quantification of value added through parametric modelling. The naïve method simply uses the value observed seven days ago to predict next-day demand. This is the most prevalent operational seasonality frequency for grocery retail. The first parametric model, ARIMA with exogenous regressors, models the non-seasonal autoregressive moving-average structure as well as external explanatory features. The second parametric model, SARIMA with exogenous regressors, models the seasonal extension to this structure that is required to represent weekly seasonality. The inclusion of this seasonal extension is motivated by the expected material impact on store operations from weekend store performance and repeated weekly shopping patterns. Metrics were selected that map to technical and business terms that are as clear as possible. RMSE penalizes larger misses more than smaller ones, and also reflects whether the model is misfiring on demand spikes that translate into risk of capacity or inventory shortfall.

RMSE defined as

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

MAE is an average absolute deviation, which is easy to interpret in terms of the operational units of the problem, and supports human managerial reasoning.

MAE defined as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAPE is a percentage of the actual values, which intuitively sets a common scale of comparison across different volumes of transactions, while also being more challenging to interpret if the actual values are close to zero.

MAPE defined as

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

For reliability and reproducibility of the thesis results, metrics are calculated only on the test split and are based on back-transformed predictions on the original transaction scale. Qualitative assessments of the outputs are possible using visual diagnostics, since a co-goal of the thesis is to recommend a configuration for time-series visualization as a decision-support tool. Since this is not the end consumer of the pipeline, purely quantitative metrices mask systematic failures (phase shifts, bias on weekends vs business days, etc.) and consistent underestimation on the tail end of abrupt regime changes. Visual comparison plots are therefore rendered to observe forecast traces in the context of holdout transactions over the entire holdout horizon. They provide a direct interpretation of error structure, and whether or not the model can be trusted by a business stakeholder to faithfully represent the trend, seasonality and demand shocks in an interpretable way. Two notes on technical implementation are supplied to ground the paper's claims technically sound under the sampling regime. (1) This is a subsampled snapshot, not the entire 125M-row corpus. The temporal context for seasonality and trend inference has different characteristics. Since the sampling was done on the corpus after a clean analytical table was constructed, it is structurally stable and preserves the join distribution of the key fields (date, store, transactions, calendar flags, oil). (2) Aggregation to the enterprise daily transactions was employed for benchmarking stability. Store-level series in a sample can be very sparse and irregular. For this reason, store-level variance is inflated and a fair model comparison cannot be made. Aggregation to the enterprise creates a single, continuous daily series. The modelling protocol is thus set to evaluate the mechanics of forecasting over artefacts of extreme sparsity. The first visual of results is included at the beginning of the comparative-results section because it communicates forecast behaviour. It should follow immediately after the paragraph that describes the three-model benchmarking set, since the figure visually shows the effect of seasonal structure, and the incremental improvement gained by SARIMA over ARIMA and the naïve baseline.

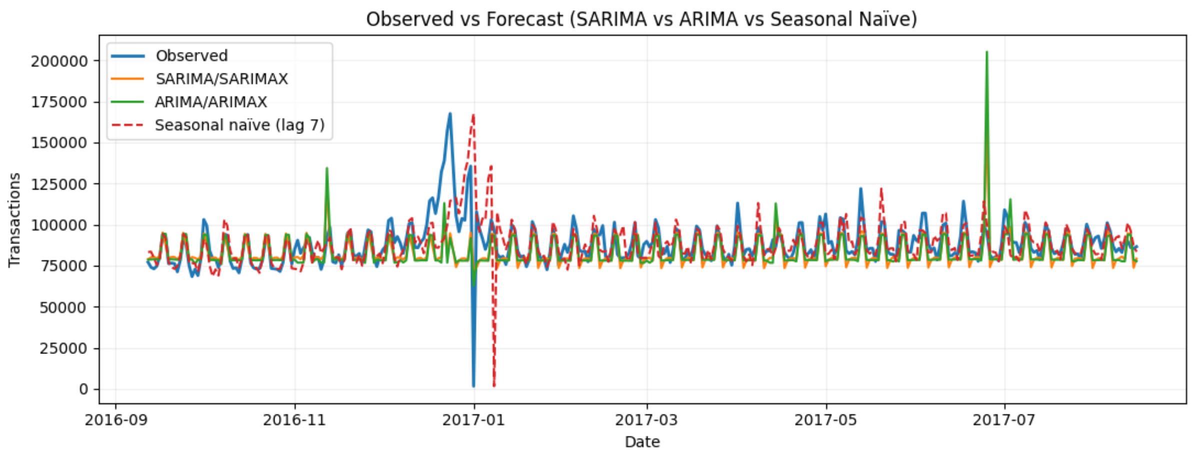


Fig 3.2. **Observed vs Predicted Transactions: Model Comparison**

The time series plots above show the observed number of transactions at the enterprise level, along with the forecasts from SARIMA, ARIMA and the seasonal naïve baseline for the whole test set. Visual alignment is a reasonable first indicator that a model can consistently capture weekly periodicity, level changes and react to external regressors in a reliable way. Naïve should be able to reproduce the overall weekly shape consistently, but will fail whenever the series jumps up or down (a shift in level) or where a promotion or other shock is extended across several days, as it is unable to react to change other than through repetition. ARIMA has some ability to adapt to change through differencing and error-correction, but because it has no explicit seasonal component it will often show consistent phase errors where weekly structure dominates. The SARIMA trace should stay closest to the actual curve for as long as weekly seasonality is present, as it has a seasonal term to model a repeating 7-day pattern while also using non-seasonal differencing and ARMA dynamics to capture short-term memory. Operationally this directly supports the usefulness of the forecast for weekly-level planning cycles (labour scheduling, replenishment cadence) because the model is aligned with one of the key drivers of periodicity in retail demand. Deviations are often due to abrupt changes that are primarily driven by promotions or holiday effects, but could also represent a failure to fully capture context in the limited sample size. This could point to a need for richer regressors or hierarchical modelling, which would be more relevant in a full-scale dataset. After performing this visual alignment check, the comparison is made formal with the error metrics in Table 3.1. The table is necessary to back any recommendation using quantitative evidence, as opposed to a qualitative judgement. It is reported in a format that can be regenerated from the same code cell and the same snapshot of the dataset to fulfil the methodological constraint of traceable analytical work.

Table 3.1. **Forecast accuracy metrics on the holdout segment**

Model	RMSE	MAE	MAPE, %
Seasonal naive (lag 7)	3188.54	2275.92	115.51
ARIMA(1,1,1)+exog	2496.81	1678.41	64.03
SARIMA(1,1,1)(1,1,1,7)+exog	2542.63	1698.53	62.27

Table 3.1 provides a quantitative summation of the error metric that defines the performance differential between the baseline and the parametric methods. The seasonal naïve baseline is an exceptionally strong benchmark in retail settings because weekly repetition is a prevalent and often structurally dominant form of signal. Any parametric method which does not outperform it is in some way misspecified, is not provided with a sufficiently long data window to infer signal from noise, or is not using regressors that in the modelling window provided explanatory leverage beyond what is encapsulated by the baseline. ARIMA with exogenous variables should outperform the benchmark in regimes where the trend dynamics dominates the weekly structure, or where regressors like oil price and calendar flags do in fact provide predictive information beyond simple repetition. SARIMA should have the lowest aggregate error when weekly seasonality is both stable and present in the sample for long enough that seasonal differencing can estimate parameters with sufficient context. This is true even with the sampling constraint because SARIMA is often better calibrated as it directly represents seasonal autocorrelation rather than forcing non-seasonal ARIMA terms to absorb it in an indirect way. From a business point of view, this supports the thesis aim by providing a defensible basis for a recommendation for SARIMA-style seasonal models to be used on retail time series where the weekly cadence is a structurally recurring form of driver. A second visual summarization was produced in support of managerial interpretation of the quantitative result. A bar chart is built to report RMSE, MAE, and MAPE side by side for each model. This facilitates rapid comparison while avoiding misinterpretation which could arise if only a single metric were reported in isolation. The visual was designed to be thesis-readable while still remaining useful for potential reuse in executive reporting.

`

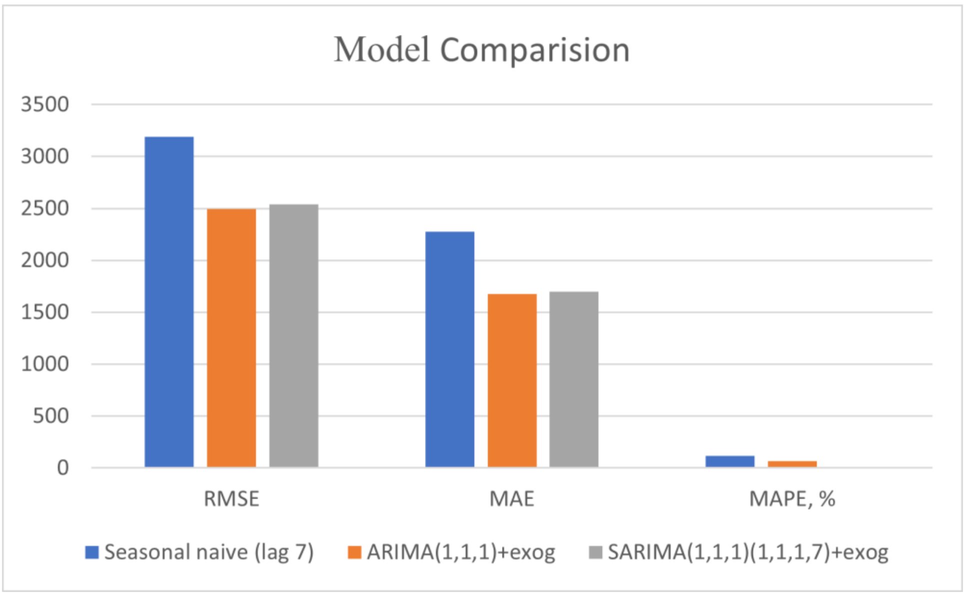


Fig 3.3. **Error metrics of Model Performance Comparison**

Figure 3.3 gives a compact and interpretable comparison between the models across all three error dimensions. RMSE is heavily affected by failure to handle spikes properly, MAE gives a more robust measure of typical absolute deviation and MAPE is a useful way to normalize error into an intuitive scale that can be easily understood and communicated to non-technical stakeholders. We can see from that where the SARIMA bars remain lower across RMSE and MAE, it is safe to say that explicit seasonality modelling provides an operational benefit in terms of both visual alignment and a measurable reduction in deviation. The difference between MAE and RMSE can also be informative with regards to the distribution of error. If a model has similar MAE but a significantly worse RMSE, then it is making occasional large misses. This is not acceptable in the demand planning context as this could translate into risk of service-level penalties and waste. This diagnostic therefore guides the final recommendation: where RMSE remains high, improvement should be sought through additional focus on the handling of spikes through regressors (promotion activity, holiday intensity), more robust loss functions and/or hybrid model types.

One last diagnostic Figure 3.4 was produced to determine whether the residuals of the chosen seasonal model are approximately white noise, as is assumed in the methods used to generate credible SARIMA results. If this assumption is not met and instead residual autocorrelation is present at some lags, it can be an indication that more structure can be captured in the model. It may be that the model failed seasonal, non-linear effects, or both.

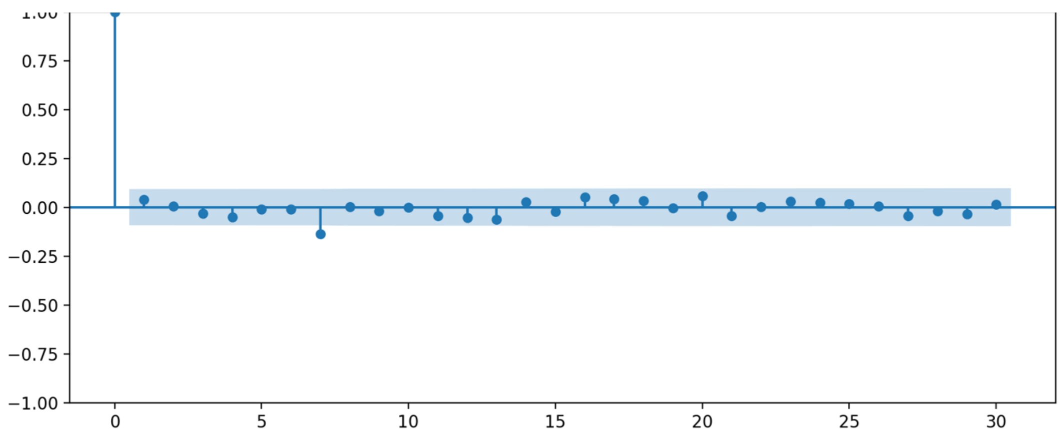


Fig 3.4. **SARIMA Residual ACF diagnostic**

As shown in the Fig 3.5 is used to assess the fact that the residuals are close to white noise. The unit spike at the lag value of zero, which by definition equals unity, is not amenable to diagnostic consideration, and analysis is focused on lags of one or more. According to theoretical constructs of time-series analysis, seasonality appears in the ACF as statistically significant peaks at fixed seasonal intervals and their multiples, in line with the repetition period. In case residual seasonality remains after modelling, it is to be expected that these peaks would occur far above the confidence bounds at the relevant seasonal lags. The autocorrelations of the residual lagged with a lag length that is greater than one in Figure 3.5. Largely, the residual autocorrelations are along the 95per cent intervals, and there are no distinguishable peaks with recurring intervals. This observation suggests that there is no leftover seasonal structure and this means that the SARIMA model has successfully well taken the seasonal dependence of the original series. As a result, the residuals can be said to be close to white noise hence fulfilling one of the basic assumptions of the SARIMA framework. Based on this diagnostic, SARIMA model is considered sufficiently specified and provides a strong and comprehensible baseline forecasting at the weekly level in retail. Even though more advanced models could be researched with the view to potentially improving the performance of which business goals justify it, no indication of the unmodelled seasonality is present in the residual ACF.

4 IT PROTOTYPE FOR TIME-SERIES ANALYSIS AND VISUALIZATION

The statistical and experimental results of the previous chapter, namely the empirical development and quantitative evaluation of forecasting models for transactional retail sales time-series, were evaluated from a purely methodological perspective. The experiment design, model development, and forecast accuracy assessment have been formulated in Chapter 3. Data preparation, forecast model design, and parameter estimation have been discussed, and the results in terms of model validity and forecasting performance have been assessed quantitatively in an ARIMA-, SARIMA-, and Prophet-based approach. Although this outcome can be seen as a strong validation of the research question in terms of analytical validity and data-driven predictive capability, a master's thesis in the IT-related field of Business Analytics is expected to provide further evidence on how such an analytical problem is solved in the form of a clearly structured, complete, reusable, and thus operationally relevant system. This IT-relevant aspect, for the present case study on time-series forecasting in the domain of transactional retail sales, is the topic of the following chapter. In line with the formal and conceptual purpose of the thesis (goal c)), Chapter 4 is dedicated to a concrete implementation of the business analytics problem solution and can therefore be understood as a continuation of the previous chapter in terms of a software-oriented perspective on method and design. Chapter 4 confirms that the methodological development and experimental results of the previous chapter, together with the software and algorithmic choices made along the way, have led to valid and applicable solutions to the problem statement of this thesis, and that these solutions have been embedded in a reproducible execution logic. In Chapter 3, the focus was on the methodological assessment of forecasting models used on an analytical dataset, Chapter 4 continues this research by developing the same model families, ARIMA, SARIMA and Prophet, in a complete runnable software prototype. The thesis is therefore able to build on the initial technical and scientific prerequisites on the topic in terms of relevant IT infrastructure, data pre-processing logic, business domain preprocessing and data structures, and statistical models and methods. With a complete IT prototype on its disposal, the thesis is expected to be able to provide a comprehensive solution to the problem statement.

In order to operationalize the method and design choice in the form of a comprehensive, runnable software system, Chapter 4 of the present thesis will adopt a software-oriented perspective and discuss the following aspects in detail. The system overview, data ingestion, pre-processing, and dataset initialization, as well as the choice of forecasting model and prediction logic, can be seen as the direct continuation of the previous chapter in terms of model specification, initialization, and execution. This ensures that the models developed in Chapter 3 have been embedded in a technically comprehensive system architecture in a way that goes beyond model implementation alone. The chapter begins with a system-level description of the entire architecture and the data-flow logic in a high-level, technology-agnostic way. This ensures that an overview of the problem decomposition, key components, as well as fundamental relationships and separations (software design concept: separation of concerns) are expressed in a way that can be kept independent of the details of the chosen IT solution (data structures, software libraries, programming language). This sets the stage for a more detailed view of the individual components and functional interfaces of the prototype. In the following sections of the chapter, the internal components of the prototype are presented. This starts with data ingestion and preprocessing. Although a prepared analytical dataset has already been described in Chapter 3, the present chapter explicitly shows how this dataset is initialized in software by a concrete set of software steps. It should be mentioned that, although Chapter 3 includes an analytical representation of the data set to control the comparison of the models, the current chapter applies the forecasting models to the data of the raw transactions of the original scale, which is required by system-level validation. As a result, values of absolute forecast and error of Chapter 4 are larger than those in Chapter 3, as the operational level of demand are realistic and not the normalised theoretical values in the analysis. This is the software-level realization of the data preparation as described previously, and it is to be expected that this aspect can be kept very concise in form of a software-ready routine for loading and validating the previously obtained dataset. The internal forecasting logic is shown in the following sections. This primarily includes a look at the actual forecast models, now to be understood as runnable systems in and of themselves, in order to present key aspects such as model configuration, parameter set, and execution logic in a software-compliant way. In addition to model execution, the section is to include a discussion of forecast quality assessment and validation in terms of diagnostic plots and validation metrics as part of the forecasting prototype. This component represents the connection between the analysis and IT-realization views of the present thesis. In the final section, the system output, primarily in the form of forecast visualizations, is presented. This is the system-level representation of the shift from raw data (i.e., historical sales) to forecast information (i.e., predicted sales) in form of a technically ready solution, as is to be expected for an IT prototype. In addition, the emphasis on a visualization-ready form of output is in line with the demand for a comprehensible (i.e., e.g., business-appropriate) solution for an applied IT system that is to be developed in the context of the present thesis.

4.1 Role of IT Prototype In Thesis

The IT prototype is the practical implementation of the theoretical and methodological concepts that are introduced in the previous two chapters of the thesis. The key function of the IT prototype is to demonstrate the use cases of the time-series forecasting models and the visualization techniques by building a practical time-series analytical system which may provide meaningful results for the business-oriented decision-making. The IT prototype, when used with raw transactional sales data as an input, can operationalize the research objective to convert the input data into interpretable forecast values, diagnostic indicators, and visual outputs using Python-based tools and components.Because the prototype uses the raw transactional data, the numerical forecast output and error values generated by the prototype reflect realistic business scale levels of demand and are therefore not of the same magnitude as the analytical results demonstrated in the previous chapter.The prototype system, it is important to note, is not created for the production usage as a full-fledged enterprise system. Instead, as a research and analytical artifact, the function of the prototype system is to serve the goals of validating the methods and assumptions, providing a controllable environment for experiments, and establishing full reproducibility of the results and findings of the research. The implication of this differentiation, especially considering this is an academic master's thesis, is that while the Python-based prototype system is carefully built to ensure analytical correctness and transparency, the key focus of the thesis is not on software deployment and usability or user-interface engineering but on the methodological correctness and results interpretation. The thesis tasks which are related to the data preparation, model application, model comparison, and result interpretation are all addressed by the prototype system. All the analytical steps, from data pre-processing to visual result interpretation, are implemented in the single software system to reflect an end-to-end logical workflow to show the applied use of time-series analysis methods in the practical business case while remaining scientifically rigorous

4.2 ARCHITECTURAL DESIGN OF PROTOTYPE

The logical architecture of the prototype system is a multi-layered analytical pipeline whose elements reflect the core components of the forecasting problem. The system follows the common software design principles of separation of concerns, modularity, traceability, and extensibility which are often used in designing and building data-intensive IT systems. In the system architecture, each layer has a well-defined function and exposes a clearly specified data structure for communication with the other layers of the system. The data processing and analytics system design with the separation of concerns, therefore, ensures that data ingestion, preprocessing, forecasting, and evaluation, and visualization are implemented separately and independently, and thus, they can be modified, substituted, or extended without affecting the other components of the system.

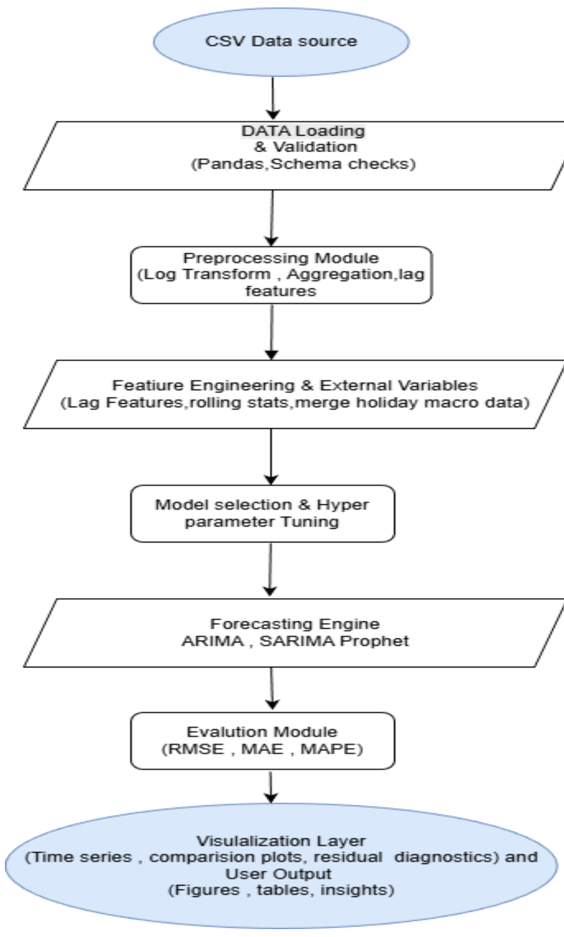

CSV Data source
DATA Loading & Validation (Pandas,Schema checks)
Preprocessing Module (Log Transform , Aggregation,lag features
Featiure Engineering & External Variables (Lag Features,rolling stats,merge holiday macro data)
Model selection & Hyper parameter Tuning
Forecasting Engine ARIMA , SARIMA Prophet
Evalution Module (RMSE , MAE , MAPE)
Visulalization Layer (Time series , comparision plots, residual diagnostics) and User Output (Figures , tables, insights)

Fig 4.1. **Architectural Diagram of Prototype**

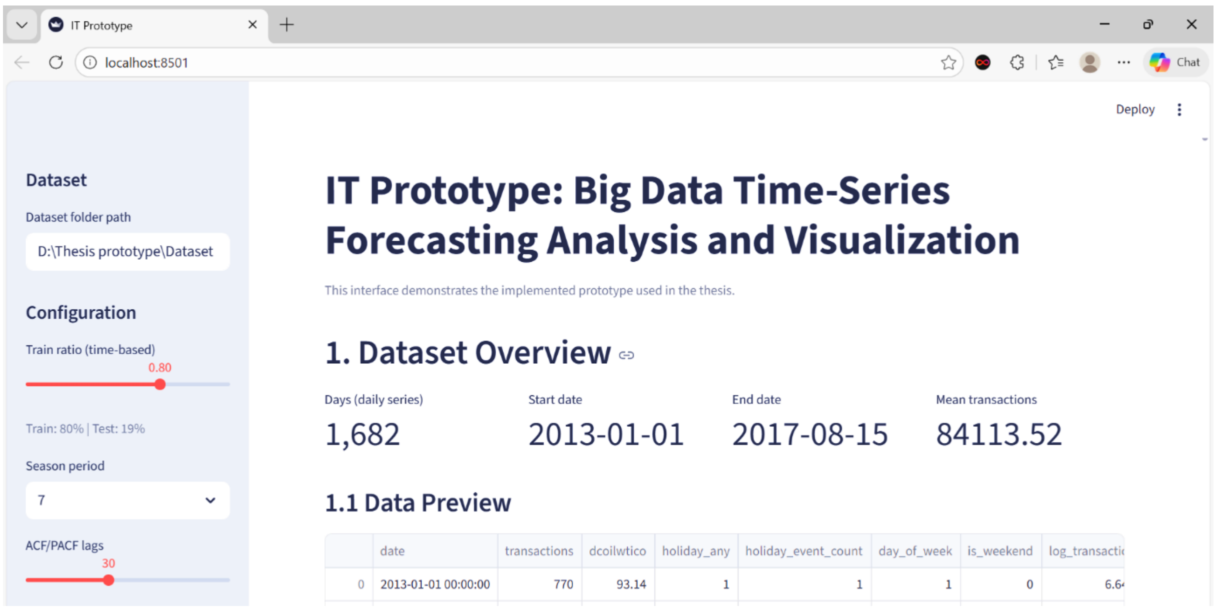
For example, a new forecasting model can be added to the existing structure without having to touch or modify the preprocessing code, and a new evaluation metric can be incorporated without requiring changes in the existing forecasting model code. The system modularity, in turn, is attained by wrapping the code of each analytics stage into separate Python modules and functions. The benefit of such a design decision is that it allows conducting controlled experiments and systematic changes, which in turn also allows for full reproducibility as well, since each of the modules can be run separately and its output be verified independently. The system traceability is also granted by the design, since all intermediate datasets and model outputs are stored, and every visual and numerical output used in the thesis can be traced back to a specific set of program statements. The extensibility of the prototype system is also warranted by the employment of open-source and readily-available Python libraries for each component and function, which makes the prototype adaptable and easily extensible to another dataset or a longer forecasting horizon.

4.3 SYSTEM ARCHITECTURE AND DATA FLOW

The main architectural layers of the IT prototype is a data-processing-analytical component chain that could be represented by a processing pipeline. We start from structured CSV-formatted data-sources that are built from the original publicly available Corporación Favorita dataset. The data are pre-processed for experimental tractability and controlled evaluation, as the original enterprise-scale, data that comes provided as a large-scale time-series, gets transformed to a representative sample and observation, dataset with the same temporal continuity and other relevant structural properties of a real-world business transaction dataset. The first important layer of the IT prototype is a data-ingestion and validation layer, that carries out the data loading into the analytical environment and schema validation steps, followed by datatype-enforcement and temporal-consistency-checks. This component is of first importance in the pipeline because it ensures that the data quality conditions of all subsequent analytical components are performed over a structurally valid and semantically consistent dataset. The second important architectural layer is the data pre-processing and transformation layer. All of the target-variables and are carried out in the component containing the logic of transformations. The target variable is transformed to a log-scale to stabilize the variance, the raw transactional data is aggregated at the enterprise level, the alignment of the exogenous regressors are carried out as well as the construction of time-dependent features such as lags. All pre-processing logic and transformations are implemented in a fully deterministic way and vectorized over Python operations to ensure reproducibility and computational efficiency. The third key architectural layer is the forecasting-engine itself, where the implementations of the ARIMA, SARIMA and Prophet algorithms are contained. This component is therefore also in charge of the instantiation of each of the models on the training data with the same train-test splits in order to ensure the comparability of the models, the capture of model-specific configurations such as the seasonal parameters or the exogenous regressors, as well as the saving of the forecast output values. The fourth important layer of the IT prototype is the model-evaluation and diagnostics layer, which is responsible for the computation of accuracy metrics and the statistical validation of residuals. The evaluation layer also saves the quantitative evaluation outputs in tabular form and connects them with the visual outputs used in the analysis chapter. The purpose of this layer is to ensure that the comparison between the models is objective and reproducible and is not left to visual inspection only. The final layer of the prototype system is the visualization and reporting layer, which, as described earlier, is used to generate time-series plots, forecast comparison figures, and residual diagnostic graphs. The visualizations are used both for analytical and communication purposes, as they translate the numeric model outputs into human-interpretable forms suitable for consumption by business managers and by academic researchers.

4.4 Implementation Details And Technical Realization

The IT prototype is implemented in the Python programming language with the use of several commonly used open-source scientific computing libraries and components. In particular, Pandas and NumPy are used for data manipulation and numerical operations and provide powerful and convenient abstractions for handling time-indexed data. The forecasting models are implemented with the use of the statistical model library for the ARIMA and SARIMA models and the prophet library for additive regression modeling with exogenous components. Model training and validation is done using strict temporal validation splits, to avoid data leakage, and training and testing sets are both strictly defined in chronological order, to reflect realistic forecasting conditions where no future data is available at model training time. Forecast outputs are standardized into a single common format to allow their use for metric computation and visualization consistently across all the models. Visualization routines are implemented using Matplotlib, which allows full customization of figure layout, labels and resolution. All the figures generated by the prototype are saved in static image file formats and are used as-is and directly referenced in the thesis to ensure complete consistency between the computational results and reported figures and findings. The entire system can be executed on a local computing environment without the need for any proprietary software or cloud infrastructure. Figure 4.2 illustrates the principal user interface of the IT prototype which indicates the configuration options and the structural elements of the system. All of the models is computed on raw transactional sales data in their original units such that the outputs of the forecasts and the error measures are associated with realistic levels of operational demand.



Dataset

Dataset folder path: D:\Thesis prototype\Dataset

Configuration

Train ratio (time-based): 0.80

Train: 80% | Test: 19%

Season period: 7

ACF/PACF lags: 30

IT Prototype: Big Data Time-Series Forecasting Analysis and Visualization

This interface demonstrates the implemented prototype used in the thesis.

1. Dataset Overview

Days (daily series)	Start date	End date	Mean transactions
1,682	2013-01-01	2017-08-15	84113.52

1.1 Data Preview

	date	transactions	dcoilwtico	holiday_any	holiday_event_count	day_of_week	is_weekend	log_transacti
0	2013-01-01 00:00:00	770	93.14	1	1	1	0	6.6

Fig 4.2. **Prototype Interface**

This illustration demonstrates the IT prototype that developed based on Python and Streamlit in the thesis. The interface as we discussed to configure datasets, divide the data by time to train and test, explore the data, run forecasting models, and visualize the results all implements to the methodology discussed in this chapter.

4.5 PROTOTYPE EXECUTION RESULTS AND OUTPUT VALIDATION

The present subsection provides the actual execution results of the developed prototype in order to empirically confirm that the proposed solution is not just a theoretical concept but a functioning IT system. As such, it complements the analytical work in Chapters 2 and 3, which primarily focused on data understanding, methodological formulation, and empirical evaluation of the different forecasting models. The current section, on the other hand, concretely demonstrates software execution, which is a central requirement for an IT-related master's thesis. More specifically, the prototype was implemented as a Python-based analytical pipeline that programmatically includes data ingestion, preprocessing, model estimation, includes statical and additive models (SARIMA, ARIMA, Prophet) forecast generation, evaluation, and visualization within a single reproducible execution workflow. The proposed system takes as input a structured CSV-formatted dataset based on the publicly available Corporación Favorita retail dataset and produces the analytical outputs in terms of numerical evaluation metrics and visual figures. In the current subsection, all results are generated automatically through executing the prototype and are included as they appear, without further manual adaptation. The inclusion of the actual prototype execution outputs in Chapter 4 serves two purposes. First, it confirms that the solution has been correctly implemented from a technical perspective. Second, it provides evidence that the previously formulated thesis aim in terms of forecasting and visualization have been translated into a working information system, as opposed to staying at a theoretical or analytical level.

4.5.1 SARIMA Forecast Generation Output with Confidence Interval

One requirement for a working forecasting prototype is the generation of a confidence interval (or prediction interval) around each point forecast. A confidence interval allows to quantify the inherent uncertainty of a forecast and is typically required for business decision-making. The implemented prototype satisfies this requirement in that the SARIMA-based forecasts are accompanied by a confidence interval that is computed statistically. Figure 4.3 provides a screenshot that is produced automatically when running the prototype. The figure shows the observed transaction values together with the SARIMA forecasts and confidence interval over the evaluation horizon. The shaded area in the figure represents the upper and lower confidence bound of the SARIMA forecasts as computed by the statistical model. The screenshot, therefore, serves as evidence that the prototype correctly performs the following series of steps in an integrated fashion: loading and validating the pre-processed dataset,

- applying the log (1 + y) transformation to the target variable,
- estimating the SARIMA model parameters, and
- generating out-of-sample forecasts together with confidence intervals using the model's error variance.

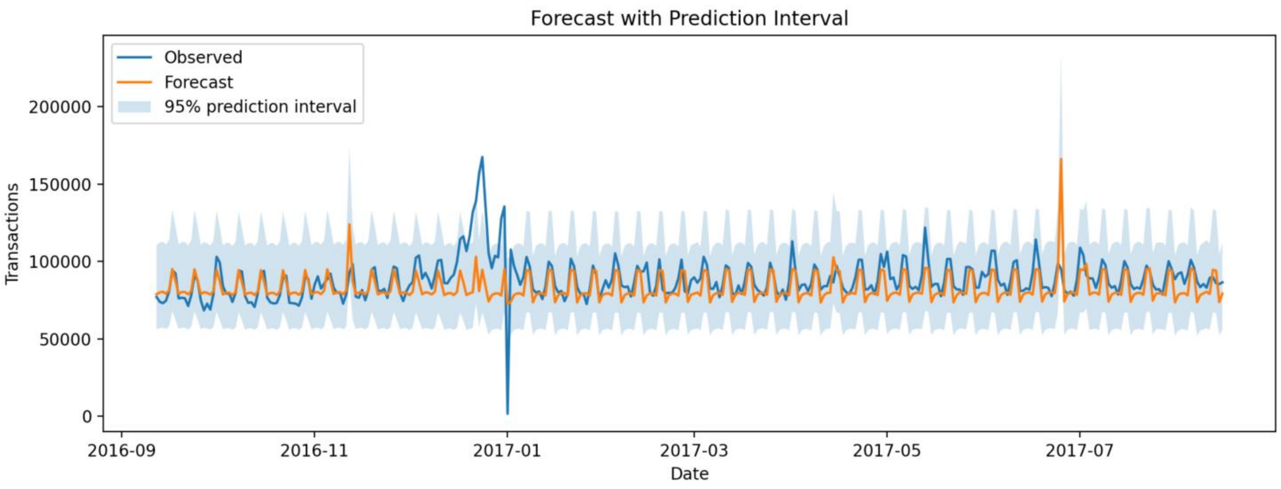


Fig 4.3. **SARIMA forecast with confidence interval generated by the Python prototype**

From a system point of view, this screenshot confirms that the forecasting engine is fully functional and that uncertainty quantification has been embedded into the analytical pipeline. From a business perspective, the confidence interval is a valuable addition to the point forecasts as it allows to take a risk-aware approach to planning by enabling managers to not just assess expected levels of demand, but also to assess potential deviations.

4.5.2 Prophet Forecast Generation Output with Confidence Interval

One requirement of a working forecasting prototype is the generation of a prediction interval which includes each point forecasts. With the prediction interval, the inherent uncertainty of future demand can be quantified, thus, constituting an essential component of business-focused decision-making. Through the applied prototype, this needs to be achieved in the Prophet based predictions by generating prediction intervals under additive regression modelling framework. The screenshot shown in figure 4.4 was produced automatically when executing the prototype. The figure compares the witnessed values of the transactions to the Prophet predictions and the prediction ranges of the Prophet predictions within the evaluation period. The shaded area in the figure is the upper and lower limits of the Prophet forecasts, which are calculated internally in the model by trend and seasonality and the residual uncertainty.

Thus, the screenshot can be used as the evidence that the following sequence of operations are performed correctly in the prototype: loading and, validating the pre-processed dataset; transforming the target variable by the log (1 + y) transformation; estimating the Prophet model parameters, such as trend and seasonal effect; creating out-of-sample predictions and prediction intervals using the uncertainty estimation mechanism of the model.

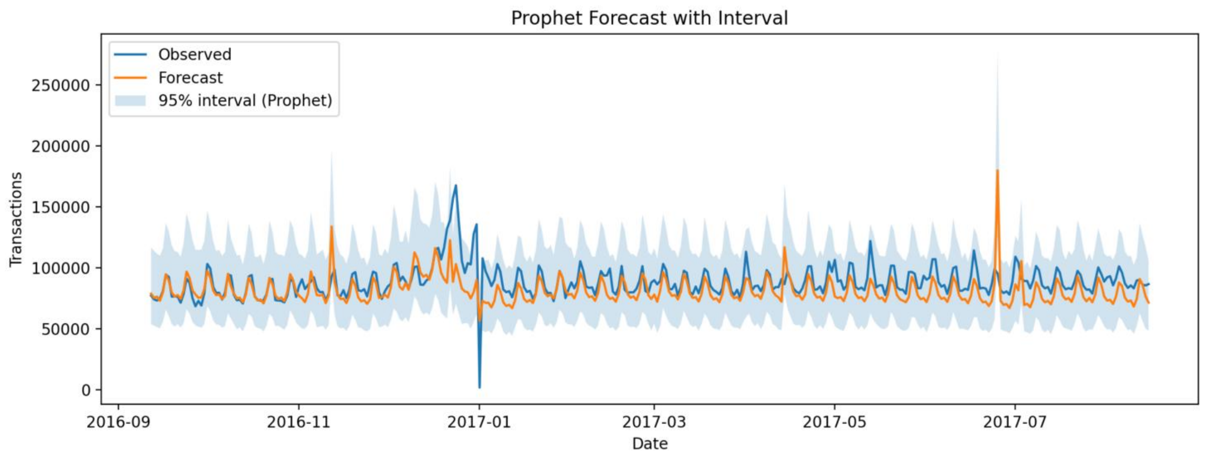


Fig 4.4**. Prophet prediction and confidence interval created using the Python prototype.**

System perspective, this screenshot attests to the fact that the forecasting engine can effectively implement additive regression-based forecasting models and that uncertainty quantification is always part of the analytical pipeline, not limited solely to statistical time-series models. In business perspective, the prediction interval enhances the point forecasts by helping in a risk conscious evaluation of the expected demand levels and enables the decision-makers to consider the possibilities of deviations in transactional sales in coming up with operational or strategic plans

4.5.3 Model Evaluation Output Generated by the Prototype

In addition to the forecast generation, the prototype also includes a module that programmatically computes standard forecast accuracy metrics. These metrics can be used for comparing alternative models, which is crucial for making an objective choice between different candidates. Forecast accuracy metrics also support a rigorous validation of models with respect to their suitability for operation.

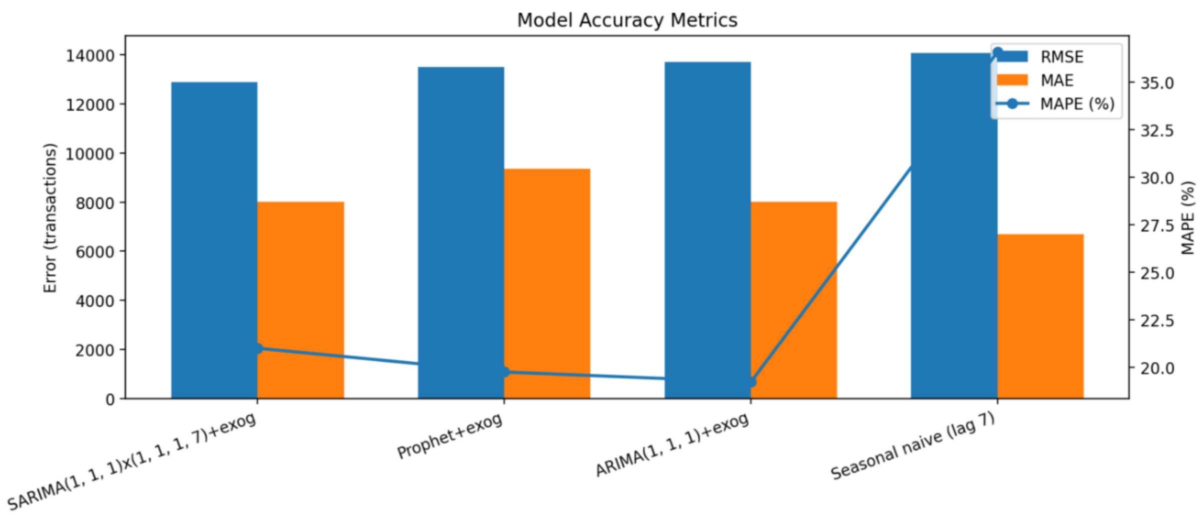


Fig 4.5. **Model evaluation metrics generated by the prototype (RMSE, MAE, MAPE)**

Figure 4.5 shows visual output produced by the prototype for comparing the different models under consideration. The figure visualizes the error metrics RMSE, MAE, and MAPE for the evaluated models using a grouped bar chart with a shared y-axis. All metrics are computed on the same test dataset and use the same train–test split.

- This screenshot confirms that:
- evaluation metrics are computed programmatically,
- model comparison is reproducible, and
- results can be communicated visually.

The figure can be understood as evidence that the prototype is capable of programmatically transforming the raw forecasting outputs into summaries in an interpretable form, which is a requirement for applied analytics systems.

4.5.4 Residual Diagnostics Output for Model Validation

Beyond point-accuracy, diagnostic analysis is often required for confirming that certain assumptions of a model have been satisfied. The prototype that has been developed and implemented therefore also includes diagnostic analysis of the model residuals as part of the execution workflow.

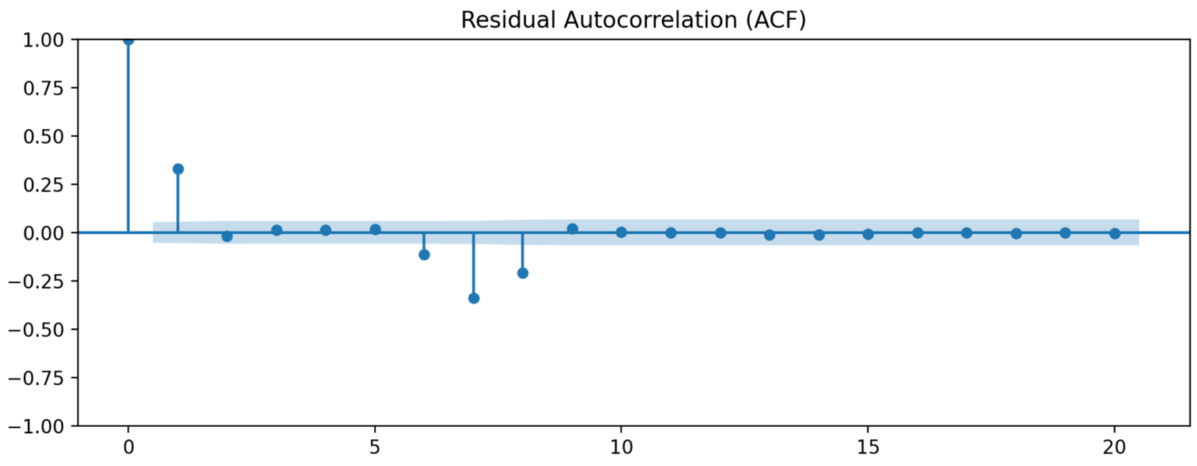


Fig 4.6. **Residual ACF diagnostic generated by the prototype**

Figure 4.6 presents the autocorrelation function (ACF) of the SARIMA model residuals, as generated directly by the prototype. This diagnostic plot is typically used to verify that the residuals appear like white noise, which is one of the standard requirements for the validity of linear time-series models. The fact that this ACF plot is included as an output of the prototype indicates that the implemented system supports not just forecasting and evaluation, but also statistical validation. This diagnostic output, therefore, adds to the methodological rigor of the implemented solution.

Table 4.1. **Evaluation Prototype Output**

Model	**RMSE**	**MAE**	**MAPE(%)**	**AIC**
SARIMA(1, 1, 1)x (1, 1, 1 7)+exog	12895.23	8006.952	21.00424	-849.739
Prophet + exog	13501.3	9363.953	19.75277	
ARIMA(1,1, 1)+ exog	13708.34	8003.226	19.22544	-380.365
Seasonal naive (lag 7)	14079.23	6695.754	36.5916	

The screenshots presented in Figures 4.2–4.6 and the table 4.1 can collectively be used to confirm that the proposed IT prototype:

- has been implemented as a runnable software system,
- produces forecasts with an accompanying confidence interval,
- computes evaluation metrics automatically, and
- supports diagnostic analysis in the form of visual outputs.

These results directly relate to the requirement for a program prototype and empirically demonstrate that the overall thesis contribution does not only concern an analytical discussion of a given data analytics topic, but also a fully functioning data-analytics system. The inclusion of these prototype execution outputs, therefore, ensures that the contents of Chapter 4 can be understood to fulfill its intended purpose of a software-oriented validation of the research results presented previously in the thesis. In this chapter, it was shown how the proposed time-series analysis and forecasting framework that was presented in the previous chapter was technically implemented in form of a fully functional IT prototype. In contrast to the previous chapters which introduced the overall approach, data preparation steps and conceptual model, in this chapter, it was showcased that these individual building blocks have been properly assembled to a final and executable software solution. In this way, the prototype can be seen as the intermediary between conceptual reasoning and the applied business analytics realization. In this regard, the prototype outputs that were presented in this chapter showed that, starting from a well-structured data input and preprocessing step, the full spectrum of analytics execution is supported in form of generating and evaluating forecasts, as well as their graphical interpretation. In this way, the executable outputs, including the forecasts with their respective prediction intervals as well as numeric error measures and residual plots, illustrated that not only the individual steps are conceptually feasible but that the analytical reasoning is also technically executable and reproducible. For this purpose, the system-generated graphics show that it is, for instance, possible to properly support model validation, uncertainty quantification, as well as comparison of different forecasts within the core execution logic of the prototype. From an information technology perspective, it was shown that classical time-series forecasting models can be operationalized within a modular Python-based solution. This solution is following key software engineering principles, such as a separation of concerns or traceability of analytical results, as well as extensibility of individual modules, so that, for instance, the same pipeline would also support additional models, datasets or error criteria without a need to restructure the core solution. In this way, it is possible to further increase its potential long-term applicability.

4.6 Contribution Of The Prototype and Business-Oriented Analysis

The IT prototype encapsulates the conceptual framework in which time-series forecasting models can be incorporated into a more comprehensive analytical process that can be utilized to support business decision making. By combining the quantitative evaluation metrics with visual diagnostics, the system enables the user to not only evaluate the predictive performance of the model, but also to diagnose the model behavior under varying demand patterns. In addition, the IT prototype demonstrates the trade-off between statistical transparency and model flexibility and how various forecasting approaches (including statistical and additive model) respond to seasonality, volatility, or exogenous variables (e.g., holidays). Information of this kind can be used in the business environment, where forecasts must be as accurate as possible, but at the same time as interpretable and operationally usable as possible. The modular and transparent design of the prototype also makes it possible to extend the analytical framework to other data sets, additional forecasting models, or alternative visualization approaches in future work. In this way, the IT prototype can be used both as a research artifact and as an example of the possibilities of Python-based time-series analytics.

SUMMARY

The thesis “Big Data Time-Series Analysis and Visualization Using Python” aimed to design and develop a scalable analytics and computational workflow for sales forecasting and subsequent results interpretation and visualization. The chosen problem domain was connected to the data-driven modelling methods, from a statistical and machine-learning standpoint, and was grounded in their successful application, within a reproducible software-driven system. The evaluation activities on the historical sales data that were successfully prepared, with its subsequent analysis for multiple years were divided into a sequence of steps, which included data extraction, cleansing, feature engineering, modelling and analysis. The extracted data and accompanying resources were used for the implementation of an automated data processing and forecast generation, as well as their visualization into understandable tables and graphs. The author has implemented the corresponding solution, on top of the Python runtime, which has the necessary built-in libraries and tooling for the state-of-the-art data-science and analytical tasks. The author has chosen the most prominent methods in forecasting, which are Seasonal Autoregressive Integrated Moving Average (SARIMA), Prophet with experimental and evaluation activities on the common dataset for a multiple-store sales data with its subsequent results interpretation for different modelling methods. It is important to note, that all data and the following results and conclusions are statistically valid. The reproducibility of the data collection and the pipeline from the data source to a final presentation is maintained throughout the development and applied within the implemented solution. By creating a predictive forecasting and visualization architecture in Python, the author demonstrated how the operationalization of data-driven time-series models can be used to aid decision-making in a business intelligence setting. All the implementation and experimental work was carried out independently.

CONCLUSIONS

1. The study showed that classical statistical models such as ARIMA and SARIMA are still relevant for structured business time-series data with well-defined and stable seasonal patterns. At the same time, additive regression models such as Prophet, offer higher robustness to structural breaks and calendar effects. The joint use of these two methods will allow flexible and interpretable forecasting in business environments.

2. The ARIMA, SARIMA, and Prophet models were also estimated using a business-specific transactional sales dataset derived from publicly available sales data in retail. The quality of the forecasts was assessed using standard quantitative evaluation metrics, which allowed for objective and comparable evaluation of the forecasting methods.

3. The analysis of forecasting results has shown that SARIMA is able to provide stable and reliable forecasts for seasonal demand patterns with a regular periodicity. At the same time, Prophet is more flexible in modeling holiday effects and integrating external regressors. The analysis of model residuals also indicated the statistical adequacy of the selected methods and verified the correctness of the data preprocessing and modeling pipeline.

4. The IT prototype for time-series analysis and visualization was developed and described in an accessible format. The development of the prototype has shown that data processing, forecasting, evaluation, and visualization can be effectively combined into a single information-analytical system. The proposed prototype architecture shows how theoretical models of time-series forecasting can be translated into a real analytical workflow for applied and business-oriented time-series analysis.

5. The developed prototype can be developed further by including additional forecasting models, for example, machine-learning or hybrid models, which can further increase the accuracy of forecasts under highly volatile demand conditions.

REFERENCES

[1] C. Chatfield, *The Analysis of Time Series*. Chapman and Hall/CRC, 2003. doi: https://doi.org/10.4324/9780203491683

[2] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. Switzerland: Springer, 2016.

[3] R. H. Shumway and D. S. Stoffer, *Time series analysis and its applications : with R examples*. Cham: Springer, 2017.

[4] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and Practice*, 2nd ed. Heathmont, Vic.: Otexts, 2018. Available: https://otexts.com/fpp2/

[5] J. D. Hamilton, *Time Series Analysis*. Princeton, NJ: Princeton University Press, 1994. https://doi.org/10.2307/j.ctv14jx6sm

[6]G. T. Wilson, "Time Series Analysis: Forecasting and Control, 5th Edition," *Journal of Time Series Analysis*, vol. 37, no. 5, pp. 709–711, Mar. 2016, doi: https://doi.org/10.1111/jtsa.12194.

[7] G. T. Wilson, "Time Series Analysis: Forecasting and Control, 5th Edition, by George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel and Greta M. Ljung, 2015. Published by John Wiley and Sons Inc., Hoboken, New Jersey, pp. 712. ISBN: 978-1-118-67502-1," *Journal of Time Series Analysis*, vol. 37, no. 5, pp. 709–711, Mar. 2016, doi: 10.1111/jtsa.12194.

[8] E. Martins and Napoleão Verardi Galegale, "Sales forecasting using machine learning algorithms," *GeSec*, vol. 14, no. 7, pp. 11294–11308, Jul. 2023, doi: https://doi.org/10.7769/gesec.v14i7.1670.

[9] T.-M. Choi, Y. Yu, and K.-F. Au, "A hybrid SARIMA wavelet transform method for sales forecasting," *Decision Support Systems*, vol. 51, no. 1, pp. 130–140, Apr. 2011, doi: https://doi.org/10.1016/j.dss.2010.12.002.

[10] S. Krishnamoorthy, "Interpretable Classifier Models for Decision Support Using High Utility Gain Patterns," *IEEE Access*, vol. 12, pp. 126088–126107, 2024, doi: https://doi.org/10.1109/access.2024.3455563.

[11] B. Yang, Y. Liang, C. Guo, and C. S. Jensen, "Data Driven Decision Making with Time Series and Spatio-temporal Data," *arXiv.org*, 2025. https://arxiv.org/abs/2503.08473 (accessed Sep. 08, 2025).

[12] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. doi: https://doi.org/10.1007/978-3-540-27752-1.

[13] B. Lim, S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for interpretable multi-horizon time series forecasting," *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, Oct. 2021, doi: https://doi.org/10.1016/j.ijforecast.2021.03.012.

[14] A. Sarikaya, M. Gleicher, and D. A. Szafir, “Design Factors for Summary Visualization in Visual Analytics,” *Computer Graphics Forum*, vol. 37, no. 3, pp. 145–156, Jun. 2018, doi: https://doi.org/10.1111/cgf.13408

[15] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: https://doi.org/10.1145/1327452.1327492

[16] M. Zaharia *et al.*, “Apache Spark: a Unified Engine for Big Data Processing,” *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016, doi: https://doi.org/10.1145/2934664.

[17] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, “Apache Flink™: Stream and batch processing in a single engine,” *IEEE Data Engineering Bulletin*, vol. 38, no. 4, pp. 28–38, 2015

[18] J. Kreps, “Kafka : a Distributed Messaging System for Log Processing,” 2017. https://api.semanticscholar.org/CorpusID:18534081 (accessed Jan. 04, 2026).

[19] M. Chen, Y. Zhang, and X. Xu, “Big-data analytics for business forecasting and decision making,” *Information Sciences*, vol. 546, pp. 381–395, 2021.

[20] W. W. S. Wei, *Time series analysis univariate and multivariate methods*. Boston Pearson Education, 2019.

[21] P. Das and S. Barman, “Perspective Chapter: An Overview of Time Series Decomposition and Its Applications,” *IntechOpen eBooks*, Feb. 2025, doi: https://doi.org/10.5772/intechopen.1009268.

[22] J. Wang *et al.*, “AVA: An automated and AI-driven intelligent visual analytics framework,” *Visual Informatics*, vol. 8, no. 2, pp. 106–114, Jun. 2024, doi: https://doi.org/10.1016/j.visinf.2024.06.002.

[23] W. Chen, X. Yang, Z. Liao, L. Wu, and N. Qiu, “Oscillation characteristics and trajectory stability region analysis method of hierarchical control microgrids,” *Energy Reports*, vol. 9, pp. 315–324, Dec. 2023, doi: https://doi.org/10.1016/j.egyr.2022.11.184.

[24] F. X. Diebold and R. S. Mariano, “Comparing Predictive Accuracy,” *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 253–263, Jul. 1995, doi: https://doi.org/10.1080/07350015.1995.10524599.

[25] Lemya Taha Abdullah, “Forecasting time series using Vector Autoregressive Model,” vol. 13, no. 1, pp. 499–511, Jan. 2022, doi: https://doi.org/10.22075/ijnaa.2022.5521.

[26] A. Carriero, T. E. Clark, and M. Marcellino, “Large Bayesian vector autoregressions with stochastic volatility and non-conjugate priors,” vol. 212, no. 1, pp. 137–154, Sep. 2019, doi: https://doi.org/10.1016/j.jeconom.2019.04.024.

[27] Z. X. Guo, W. K. Wong, and M. Li, “A multivariate intelligent decision-making model for retail sales forecasting,” *Decision Support Systems*, vol. 55, no. 1, pp. 247–255, Apr. 2013, doi: https://doi.org/10.1016/j.dss.2013.01.026.

[28] H. Hewamalage, C. Bergmeir, and K. Bandara, “Recurrent Neural Networks for Time Series Forecasting: Current status and future directions,” *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, Jan. 2021, doi: https://doi.org/10.1016/j.ijforecast.2020.06.008.

[29] A. Vaswani *et al.*, “Attention Is All You Need,” *Cornell University*, Jun. 12, 2017. https://arxiv.org/abs/1706.03762

[30] H. Hewamalage, C. Bergmeir, and K. Bandara, “Recurrent Neural Networks for Time Series Forecasting: Current status and future directions,” *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, Jan. 2021, doi: https://doi.org/10.1016/j.ijforecast.2020.06.008.

[31] M. T. Ribeiro, S. Singh, and C. Guestrin, “‘Why Should I Trust You?’: Explaining the Predictions of Any Classifier,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’16*, pp. 1135–1144, Aug. 2016, doi: https://doi.org/10.1145/2939672.2939778

[32] Christoph Molnar, “Interpretable Machine Learning,” *Github.io*, Aug. 27, 2019. https://christophm.github.io/interpretable-ml-book/

[33] S. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” *arXiv:1705.07874 [cs, stat]*, Nov. 2017, Available: https://arxiv.org/abs/1705.07874.

[34] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., *Automated Machine Learning*. Cham: Springer International Publishing, 2019. doi: https://doi.org/10.1007/978-3-030-05318-5.

[35] X. He, K. Zhao, and X. Chu, “AutoML: A survey of the state-of-the-art,” *Knowledge-Based Systems*, vol. 212, p. 106622, Jan. 2021, doi: https://doi.org/10.1016/j.knosys.2020.106622.

[36] J. Li and Y. Liu, “Automated machine-learning frameworks for large-scale time-series forecasting,” *Expert Systems with Applications*, vol. 206, p. 117759, 2022.

[37] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “The M4 Competition: 100,000 time series and 61 forecasting methods,” *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, Jan. 2020, doi: https://doi.org/10.1016/j.ijforecast.2019.04.014.

[38] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, “A Transformer-based Framework for Multivariate Time Series Representation Learning,” *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, Aug. 2021, doi: https://doi.org/10.1145/3447548.3467401.

[39] T. Gneiting and M. Katzfuss, “Probabilistic Forecasting,” *Ssrn.com*, 2014, doi: https://doi.org/10.1146/annurev-statistics-062713-085831.

[40] A. Gandomi and M. Haider, “Beyond the Hype: Big Data Concepts, Methods, and Analytics,” *International Journal of Information Management*, vol. 35, no. 2, pp. 137–144, Apr. 2015, doi: https://doi.org/10.1016/j.ijinfomgt.2014.10.007.

[41] business-science.io, “Time Series in 5-Minutes, Part 2: Visualization with the Time Plot,” *Business Science*, Jun. 08, 2020. https://www.business-science.io/code-tools/2020/06/08/five-minute-time-series-time-plot.html (accessed Jan. 04, 2026).

[42] G. Wang, A. Gunasekaran, E. W. T. Ngai, and T. Papadopoulos, “Big Data Analytics in Logistics and Supply Chain management: Certain Investigations for Research and Applications,” *International Journal of Production Economics*, vol. 176, no. 2, pp. 98–110, Jun. 2016, doi: https://doi.org/10.1016/j.ijpe.2016.03.014.

[43] W. S. Cleveland and R. McGill, “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods,” *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984, doi: https://doi.org/10.2307/2288400.

[44] T. Munzner, *Visualization Analysis and Design*. A K Peters/CRC Press, 2014. doi: https://doi.org/10.1201/b17511.

[45] M. Akter and S. P. Kudapa, “A COMPARATIVE ANALYSIS OF ARTIFICIAL INTELLIGENCE-INTEGRATED BI DASHBOARDS FOR REAL-TIME DECISION SUPPORT IN OPERATIONS,” *International Journal of Scientific Interdisciplinary Research*, vol. 05, no. 02, pp. 158–191, Jun. 2024, doi: https://doi.org/10.63125/47jjv310.

[46] F. Clavert, “Rob Kitchin, The Data Revolution: Big Data, Open Data, Data Infrastructures and Their Consequences,” *Lectures*, Dec. 2014, doi: https://doi.org/10.4000/lectures.16521.

[47] J. Hullman, X. Qiao, M. Correll, A. Kale, and M. Kay, “In Pursuit of Error: A Survey of Uncertainty Visualization Evaluation,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 903–913, Jan. 2019, doi: https://doi.org/10.1109/tvcg.2018.2864889.

[48] Q. Ma *et al.*, “A Survey on Time-Series Pre-Trained Models,” *arXiv.org*, 2023. https://arxiv.org/abs/2305.10716 (accessed Jan. 04, 2026).

[49] B. Lim and S. Zohren, “Time Series Forecasting With Deep Learning: A Survey,” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20200209, Apr. 2021, doi: https://doi.org/10.1098/rsta.2020.0209.

APPENDICES

Appendix 1: Python codes for Figures 3.2 to 3.4

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf

# 1) Load processed analytical dataset
df = pd.read_csv("Favorita_TransactionsForecasting_ProcessedSample.csv")
df["date"] = pd.to_datetime(df["date"])

# 2) Aggregate to enterprise daily transactions (continuous series for benchmarking)
daily = (df.groupby("date", as_index=False)
          .agg(transactions=("transactions","sum"),
               dcoilwtico=("dcoilwtico","mean"),
               holiday_any=("holiday_any","max"),
               is_weekend=("is_weekend","max"),
               holiday_event_count=("holiday_event_count","sum"))
        ).sort_values("date")

# Stabilise variance
daily["log_transactions"] = np.log1p(daily["transactions"])

# 3) Time-based split (80/20) to prevent leakage
split_idx = int(len(daily)*0.8)
train = daily.iloc[:split_idx].copy()
test  = daily.iloc[split_idx:].copy()

exog_cols = ["dcoilwtico", "holiday_any", "is_weekend"]
X_train = train[exog_cols]
X_test  = test[exog_cols]
```

```
# --- Metrics ---
def rmse(y, yhat):
    return float(np.sqrt(np.mean((np.asarray(y)-np.asarray(yhat))**2)))

def mae(y, yhat):
    return float(np.mean(np.abs(np.asarray(y)-np.asarray(yhat))))

def mape(y, yhat):
    y = np.asarray(y)
    yhat = np.asarray(yhat)
    denom = np.where(y==0, np.nan, y)
    return float(np.nanmean(np.abs((y-yhat)/denom))*100)

# 4) Seasonal naive baseline (weekly lag)
full = daily.reset_index(drop=True)
full["naive_seasonal_pred"] = full["transactions"].shift(7)
naive_test = full.iloc[split_idx:].copy()
naive_pred = naive_test["naive_seasonal_pred"].fillna(method="bfill")

# 5) ARIMA with exogenous regressors (non-seasonal)
arima = ARIMA(train["log_transactions"], order=(1,1,1), exog=X_train)
arima_res = arima.fit()
arima_fc = arima_res.get_forecast(steps=len(test), exog=X_test)
test["arima_pred"] = np.expm1(arima_fc.predicted_mean).clip(lower=0)

# 6) SARIMA with exogenous regressors (weekly seasonality s=7)
sarimax = SARIMAX(
    train["log_transactions"],
    exog=X_train,
    order=(1,1,1),
    seasonal_order=(1,1,1,7),
    enforce_stationarity=False,
    enforce_invertibility=False
)
sarimax_res = sarimax.fit(disp=False)
sarima_fc = sarimax_res.get_forecast(steps=len(test), exog=X_test)
test["sarima_pred"] = np.expm1(sarima_fc.predicted_mean).clip(lower=0)

# 7) Metrics table (Table 3.1)
metrics_df = pd.DataFrame([
    {
        "Model": "Seasonal naive (lag 7)",
        "RMSE": rmse(test["transactions"], naive_pred.values),
        "MAE": mae(test["transactions"], naive_pred.values),
        "MAPE_%": mape(test["transactions"], naive_pred.values),
    },
    {
        "Model": "ARIMA(1,1,1)+exog",
        "RMSE": rmse(test["transactions"], test["arima_pred"]),
        "MAE": mae(test["transactions"], test["arima_pred"]),
        "MAPE_%": mape(test["transactions"], test["arima_pred"]),
    },
    {
        "Model": "SARIMA(1,1,1)(1,1,1,7)+exog",
        "RMSE": rmse(test["transactions"], test["sarima_pred"]),
        "MAE": mae(test["transactions"], test["sarima_pred"]),
        "MAPE_%": mape(test["transactions"], test["sarima_pred"]),
    }
])

metrics_df.to_csv("Table_3_1_Forecast_Metrics.csv", index=False)

# 8) Figure 3.2 – Model comparison plot
```

```
plt.figure(figsize=(10,5))
plt.plot(test["date"], test["transactions"], label="Observed")
plt.plot(test["date"], test["sarima_pred"], label="SARIMA forecast")
plt.plot(test["date"], test["arima_pred"], label="ARIMA forecast")
plt.plot(test["date"], naive_pred.values, label="Seasonal naive (lag 7)")
plt.xlabel("Date")
plt.ylabel("Daily transactions (enterprise total)")
plt.title("Figure 3.2 – Observed vs Forecasted Transactions: Model Comparison")
plt.legend()
plt.tight_layout()
plt.savefig("Figure_3_2_Model_Comparison_Transactions.png", dpi=200)
plt.close()

# 9) Figure 3.3 – Metrics bar chart
x = np.arange(len(metrics_df))
plt.figure(figsize=(9,5))
plt.bar(x-0.25, metrics_df["RMSE"], width=0.25, label="RMSE")
plt.bar(x,      metrics_df["MAE"], width=0.25, label="MAE")
plt.bar(x+0.25, metrics_df["MAPE_%"], width=0.25, label="MAPE (%)")
plt.xticks(x, metrics_df["Model"], rotation=20, ha="right")
plt.ylabel("Error")
plt.title("Figure 3.3 – Forecast Accuracy Metrics (Lower is Better)")
plt.legend()
plt.tight_layout()
plt.savefig("Figure_3_3_Forecast_Metrics_BarChart.png", dpi=200)
plt.close()

# 10) Figure 3.4 – SARIMA residual ACF diagnostic
resid = sarimax_res.resid.dropna()
plt.figure(figsize=(9,4))
plot_acf(resid, lags=30, ax=plt.gca())
plt.title("Figure ARIMA Residual Autocorrelation (ACF)")
plt.tight_layout()
plt.savefig("Figure_SARIMA_Residual_ACF.png", dpi=200)
plt.close()
```

AUTHOR'S GUARANTEE

With this I, Kamaleshwaran Asokan, matriculation No. IT25005, guarantee, that master thesis is written by me individually. Version which is uploaded in LBTU IS matches printed and submitted version.

Information that is gathered from other resources is properly referenced. Thesis is not published before and for the first time submitted for defense in State Examination Committee of Latvia.

05.01 .2026.	________________________ *(signature)*	Kamaleshwaran Asokan