Yaryna Korduba, BSc

PredictPal

A System for Visual Exploration and Prediction of Time Series Data

Master's Thesis
to achieve the university degree of
Master of Science
Master's degree programme: Computer Science

Supervisor
Univ.-Prof. Dipl.-Volksw. Dr.rer.nat. Tobias Schreck, M.Sc.

Institute of Computer Graphics and Knowledge Visualisation
Head: Univ.-Prof. Dipl.-Volksw. Dr.rer.nat. Tobias Schreck, M.Sc.

Graz, October 2024

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

I further declare that GPT-4 has solely been used for two purposes:

- As an aid in proofreading, that is, punctuation, grammar rules, word repetitions, etc.
- To simplify advanced explanatory texts to the beginner-friendly format (see Subsection 4.4.4).

03.10.2024 [illegible]

Date Signature

Abstract

This thesis presents *PredictPal*, an interactive predictive visual analytics system created to improve data-driven decision-making. PredictPal enables users to upload and interactively explore time series datasets, conduct preliminary analysis through statistical tests, and apply predictive models. Further, users can assess the results via numerical indicators and interactive visualization. The system works with univariate and multivariate datasets and supports in-sample and out-sample forecasting. Additionally, the app features a "History" section where users can browse previously obtained predictions along with their model parameters. In this section, one can also review error metrics and use the sorting functionality to identify the highest quality forecasts. PredictPal is designed to be user-friendly for beginners who need predictions in their work but do not have a deep understanding of predictive modeling. The system's beginner-friendly design includes a step-by-step analysis and prediction process, as well as simplified explanatory texts for complex terms on the UI. Through its interactive interface, the system bridges the gap between complex predictive models and intuitive data visualization, providing users with both predictive power and clear insights.

Contents

List of Figures

1. Introduction

In an era where data-driven decision-making is of great importance, the ability to foresee the future accurately has become essential for strategic planning across various industries. Time series prediction, a core of this process, incorporates the analysis of sequential data points over time to make informed forecasts about future observations.

The concept of time is multifaceted, encompassing diverse dimensions and meanings. It makes the analysis of time-related data particularly challenging. According to Aigner et al. [1], multiple categorizations of time data exist. Some of them are:

- *Scale:* qualitative or quantitative. Qualitative variables include nominal or ordinal values. Quantitative variables are based on metrics that enable numeric comparisons.
- *Frame of reference:* spatial or abstract. The spatial data model includes the "where" aspect, and this information can be used to map a data point to a screen. Abstract data is not connected to any spatial location.
- *Number of variables:* univariate or multivariate. Univariate is a data set where each time primitive is associated with only one value. In contrast, in multivariate data sets, multiple data values are connected to the same point in time.

As the data volume and the complexity of the patterns grow, analyzing the information manually and building assumptions about the future behaviors of the processes becomes more complex and error-prone. This highlights the need for user-friendly systems capable of making predictions and answering data-related questions.

Running the forecasts for data sets and simply displaying raw numeric results is often inefficient. Such results can be difficult to interpret and use in decision-making. The prediction tools address this challenge by employing visualization techniques, specifically by giving users the instruments to perform predictive visual analytics processes. The goal of visual analytics is to enable the exploration and understanding of complex data by integrating visualization, human-computer interaction, data analysis, and cognitive science [2].

Since predictive visual analytics has been gaining popularity over the last decades, multiple applications have been created to solve prediction tasks. Most such systems have been designed with expert users in mind, targeting those who possess in-depth data science knowledge and know the algorithms well. Consequently, there is a gap in the availability of user-friendly systems that are easy to use for both expert and beginner-level users. Various non-technical professions, including small business owners, healthcare workers, journalists, and marketers, could greatly benefit from simple prediction tools for better decision-making.

The primary goal of this thesis is to create a prototype of a straightforward system for quick prediction of the time series data. The system should be easy to use for both experienced data scientists and individuals who do not regularly work on prediction tasks. Users should be able to quickly verify their assumptions about the data and find the best model without requiring in-depth knowledge about algorithms.

The above goal poses the central research questions of the thesis:

RQ1: What are the key components required for building a system with the predictive visual analytics pipeline?

RQ2: How to reduce the cognitive load of a user in assessing the prediction quality and the best model?

RQ3: How to develop a comprehensive user interface for time-series analysis and prediction, which is easy to exploit for the users with beginner knowledge?

As a way to find the answers to the above questions, the PredictPal system prototype was built. PredictPal integrates the following analysis functionalities:

- visualization of the time series datasets in CSV or JSON format provided by the user,
- the "Get to know your data" section, which enables checking the hypotheses about the datasets by using statistical tests,
- the "Prediction" section, which enables users to obtain forecasts using several models (ARIMA or VAR),
- the "History" section, which allows to preserve and compare the calculated predictions for a dataset, and find the model with the smallest errors.

It is important to note that this thesis targets the prediction of abstract quantitative time series data with discrete observations recorded over equally spaced time intervals. The PredictPal prototype supports both univariate and multivariate data sets. In addition to this, PredictPal is domain agnostic. In other words, it can be used with the time series data from any field.

The upcoming chapters describe the system prototype in detail. Chapter 2 overviews related works and background in visual analytics. Chapter 3 provides information about the technical stack and architecture of PredictPal. Chapter 4 follows with a description of the interaction processes in the prototype. Chapter 5 demonstrates the example of a dataset analysis using PredictPal. Chapter 6 concludes the thesis by discussing the limitations and future work.

2. Related Work and Background

The following chapter overviews the existing approaches and systems focusing on predictive visual analytics. First, the proposed concepts of the visual analytics process are discussed. Further, the available predictive visual analytics systems for time series data are explored.

2.1. Visual Analytics Process

The visual analytics process has visualization at its core. Visualization is a field that aims to create a visual representation to help explain complex phenomena. A great definition of this term was given by McCormick et al. in 1987 [3]. They described visualization as follows:

> "Visualization is a method of computing. It transforms the symbolic into the geometric, enabling researchers to observe their simulations and computations. Visualization offers a method for seeing the unseen. It enriches the process of scientific discovery and fosters profound and unexpected insights."

Visual analytics brings the topic of visualization to new horizons. Keim et al. [4] define visual analytics as an advanced extension of information visualization that consolidates automated analysis and interactive visualization approaches. As the authors state, the main tasks of visual analytics are enabling people to:

- aggregate information and obtain insights from large, changing, and often uncertain data,
- identify the anticipated and uncover the unforeseen,
- assess the data promptly and clearly,

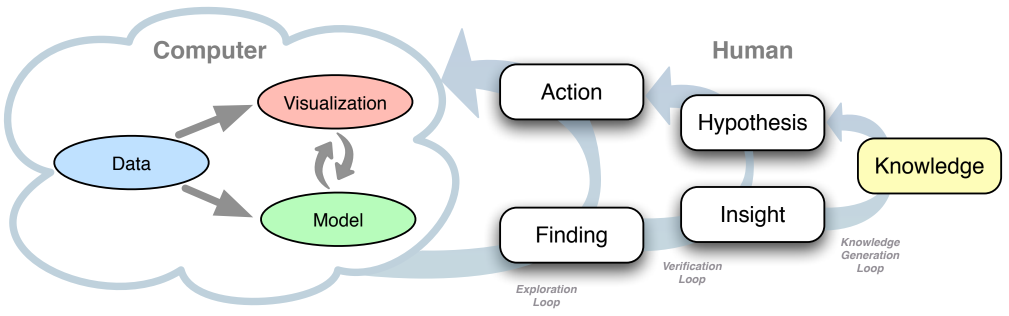


Figure 2.1.: The knowledge generation model for visual analytics. Graphic by Sacha et al. [5] with permission for usage granted under the IEEE organization, © 2014 IEEE.

- effectively convey these assessments for action.

As discussed by Sacha et al. [5], the knowledge generation model for visual analytics consists of two parts: computer and human (Figure 2.1). The computer part contains data, visualization, and analytic models. The human component describes the cognitive processes associated with an analytical session. As the authors state, these two parts complement each other since both humans and computers lack some abilities. Computers miss the creativity of human analysis, which enables people to discover hidden connections and produce unexpected insights. At the same time, humans have difficulties efficiently dealing with large amounts of information. It can be reformulated by the statement of Andrienko at al. [6] that the purpose of visual analytics is to create a synergistic collaboration of humans and computers, where each side leverages their unique strengths.

One of the topics in visual analytics that has been recently gaining interest is the topic of machine guidance for analytic activities [7]–[10]. Ceneda et al. [11] define guidance in visual analytics as a computer-assisted process that focuses on narrowing a user's knowledge gap, which blocks the efficient proceeding with the data analysis. Collins et al. [12] list a set of goals for guidance systems. Those most important for PredictPal are:

- To verify conclusions: The guidance systems may offer help in fur-

ther tests related to visual analytics. For example, the systems might provide statistical tests to prove assumptions and confirm discoveries. PredictPal, in particular, enables users to check the preliminary hypotheses about the visualized dataset. It includes checking the data stationarity, randomness, and causal relationships (in case the dataset is multivariate) by providing the results of the statistical tests on demand.
- To train users: The guidance systems may be used to improve the usability for novice users who are learning about a new visual analytics system or visualization type. It can be achieved by providing visualizations using analogy and suggested interactivity. While PredictPal does not employ an explicit guidance process to train users, it still helps users with beginner knowledge better understand the process by providing on-demand explanatory texts for different definitions on the UI. Additionally, it allows users to assess the quality of the prediction they have achieved by comparing it to other predictions in history.

The authors also name such goals as informing users, engaging, reducing mental effort, and minimizing bias.

2.2. Predictive Visual Analytics Systems for Time-Oriented Data

Predictive analytics is the process of extracting data from extensive data sets with the aim of making predictions and assumptions about future results [13]. It encompasses many methods, such as machine learning models, data mining, and statistical modeling. Predictive analytics is used in multiple spheres, including, but not limited to, healthcare and medical research, business analysis, urban planning, and finances.

Krause et al. [14] point out that model interpretation is necessarily a human task. They define three main reasons to involve humans in the modeling process:

- gaining data understanding and enabling discovery,
- building trust,

- enabling model comparison and diagnostics.

Lu et al. [15] underscore the critical role of visualization in predictive analytics. They state that many new challenges in predictive analytics appear as the models' complexity increases. Among them are the issues with data reliability, large errors in model predictions, a desire to understand the "black-box" algorithms, and the need to apply domain-specific information. As a means of facilitating the predictive analytics process, visualization is employed. It facilitates a more transparent modeling approach. Specifically, visualization enables users to gain a comprehensive overview of the data through visual representations, effectively communicate prediction outcomes, and assess their accuracy by validating the model responses.

When talking about predictive visual analytics systems for time-oriented data, one can distinguish two categories of these systems:

- innovative research prototypes - the cutting-edge experimental systems developed in academic and research settings,
- established enterprise solutions - mature, widely adopted tools used in the industry, which are reliable and provide integration capabilities.

The following subsections discuss each of these categories in depth.

2.2.1. Research prototypes

Predictive visual analytics has gained much interest in the scientific field in recent decades. Multiple systems and methods were developed for this topic. While some are designed for general purposes, others are cut out for specific exploration problems in a particular domain.

General-purpose systems

A large part of predictive visual analytics systems aims to ease the modeling process for data scientists [16]–[18]. Among the systems that inspired several decisions in PredictPal is TiMoVA-Predict, presented in 2014 by Bögl et al [19]. It is a visual analytics system supporting different types of predictions. The tool provides a rich interface with prediction capabilities (specifically, using ARIMA models), autocorrelation and partial autocorrelation plots to support the model order selection, and residual analysis plots to assess the prediction quality. One of the key system elements is the large line chart in the top left corner. It plots the actual data together with the prediction and the confidence interval and provides functionalities for interactive data exploration. The horizontal slider below the chart allows users to zoom in on the time series. The chart also provides two ways to visualize the difference between actual and predicted values: colored bars and colored areas (using diverging colors for negative and positive delta). Moreover, the interface allows the user to easily compare different models by offering the model selection history. Providing immediate visual feedback during the model selection is a distinguished benefit of TiMoVA-Predict. It brings efficiency to the process of discovering the best model for specific data.

As mentioned above, some functionalities provided by PredictPal were influenced by the TiMoVA-Predict system. Namely, it impacted the decision to highlight the difference between the target and actual values by coloring the area between them and the idea of keeping the model history for easy comparison. However, TiMoVA-Predict focus users are experts in data science who can easily understand the analysis and definitions and thus easily navigate the UI. On the contrary, PredictPal is designed to simplify the prediction process for beginners by offering a step-by-step prediction flow and easy-to-understand explanations of the definitions.

Another scientific work inspiring PredictPal was the study by Cashman et al. [20]. They developed a visual analytics workflow for exploratory model analysis and comparison. In their exploration, they considered a previously unsupported use case when the modeling tasks and the model type were unknown at the beginning of the analytics process. They introduced the notion of Exploratory Model Analysis (EMA) - the process of investigating a pool of models that can be trained on a particular data set. For PredictPal, insights gained from this work inspired the decision to incorporate several different models for selection. This approach enables users to compare the forecasts by different models and choose the most suitable option for their specific dataset.

mTSeer system designed by Xu et al. [21] also incorporates several multivariate predictive models to enable exploring, explaining, and evaluating them in comparison. The offered models include Vector Auto-Regressive (VAR), Random Forest (RF), Long Short-Term Memory Recurrent Neural Networks (LSTM), Multilayer Perceptron Models (MLP), and Convolutional Neural Networks (CNN). The system integrates the elements of guidance in the interface. For example, the app gives users a "hint" regarding the periodicity of the data when choosing the length of the input time series. To evaluate the performance of different models, mTSeer employs two measures: the Root Mean Square Error (RMSE) and the summarized feature importance. Among the views offered by mTSeer are the data selection view, the parameter view for the configuration of the model parameters, the model overview depicting the performance of various models, the inspection view enabling the comparison and interpretation of the forecasts, and other views.

PredictPal employed some concepts from mTSeer. Namely, mTSeer inspired adding a VAR model for selection and measuring the model's performance using RMSE. While mTSeer is a very well-developed system, some of its functionalities would be too complex for PredictPal since mTSeer targets users with expert knowledge.

Additionally, multiple methods and prototypes for exploring the concept drift appeared [22]–[25]. A concept denotes the underlying connection between the target variable of interest and the environment variables. For example, the virus spreading time series data could have the cure rate as the target variable and the patients' general health profile as one of the environment variables [23]. Concept drift is a phenomenon describing the unforeseen changes in the distribution of a data stream, which make the already available prediction models generate inaccurate predictions [25].

While the concept drift exploration systems focus not on the prediction itself but rather on the change in the variable relationships, concept drifts are typically captured by the accuracy change of prediction models. Visual analytics systems for concept drift exploration often include visualizations that illustrate predictions. As an example, one of the systems for the concept drift tracking in the multi-source time series data, ConceptExplorer, was proposed by Wang et al. [23]. A few interactive visualizations convey the concept drift findings. A timeline navigator view helps users quickly get an overview of the concept drifts found in multiple sources. An interactive prediction model view displays the results of the prediction models and allows sharpening insights about the findings. The concept-time view allows the users to explore and fine-tune the period of a particular concept. The concept explanation view presented with a matrix visualization depicts the correlation between the target and environment variables.

Domain-specific systems

Health sciences and healthcare. A significant number of tools have been created to assist in health explorations and healthcare. Xu et al. [26] presented the first predictive brain fiber VA system to assist neuroscientists in exploring neurodegenerative diseases. Kwon et al. presented RetainVis [27], a visual analytics system with interactive and interpretable recurrent neural networks (RNNs) on electronic medical data. Specifically, their system helps doctors predict current and future patient states, perform a "what-if" analysis, and analyze a cohort of patients. Philip et al. [28] presented a data analytics suite for exploratory predictive and visual analysis of diabetes. Using their system, a clinician can obtain an interactive survival probability chart based on the specific markers of a diabetes patient.

Finance. Finance is a sphere where predicting the future of the processes is crucial. In the last decades, multiple financial visual analytics systems have appeared. Schreck et al. [29] presented a tool to assist in analyzing extensive financial time-varying indicator data. The system includes several analytical views on the full market and specific assets and relies on the unsupervised clustering algorithm. Xie et al. [30] published an article about their Visual Analysis of the E-transaction Time-Series (VAET) prototype. The tool enables experts to explore large transaction data and gain insights into time-varying operations. Specifically, the VAET interface allows users to overview the saliency of each transaction in a time-of-saliency map, view detailed information about a transaction, point out fake transactions, and see the volume of selected operations. Jonker et al. [31] presented industry-driven visual analytics methodology for understanding financial time series models.

Economy. Predictive visual analytics systems in the economy help in forecasting complex economic processes, and support decision-making. For example, Chen et al. [32] presented RISeer. This interactive visualization system employs machine learning to explore regional industrial structure (RIS) dynamics by analyzing regional business registration activities. Their prototype exploited two time series forecasting models: RF and XGBoost. Sun et al. [33] presented DFSeer, an interactive visualization system facilitating the model selection for product demand forecasting. Wang et al. [34] proposed a forecasting method for economic statistics, analysis, decision-making, and self-regulation, where they employed LSTM models.

Urban planning. Urban planning processes include such problems as traffic prediction, public transportation optimization, environmental impact assessments, and population density projections. Multiple scientific groups have presented visual analytics systems supporting these and other tasks. For example, Kalamaras et al. [35] and Lee et al. [36] presented the tools for exploring and forecasting road traffic congestion. Alves et al. [37] presented a system for analyzing and predicting the energy data in smart cities and buildings at different time granularities.

The development and exploration of experimental prototypes in these and the other domains underscore a broader trend in the scientific community towards employing the power of data visualization and predictive analytics for advanced problem-solving. In addition, the ongoing research in predictive visual analytics holds immense potential for applications within the industry.

2.2.2. Enterprise solutions

Numerous enterprise solutions offer predictive visual analytics tools for individuals and organizations to efficiently leverage data for informed decision-making.

The list of the most popular visual analytics systems includes but is not limited to Tableau [38], Power BI [39], Qlik Sense [40], and Oracle Analytics Cloud [41]. All of the mentioned systems were also named the leaders for analytics and business intelligence platforms in 2024 by Gartner's Magic Quadrant [42]. Further, we will discuss the two most popular solutions, Tableau and Power BI, and review their predictive analytics features.

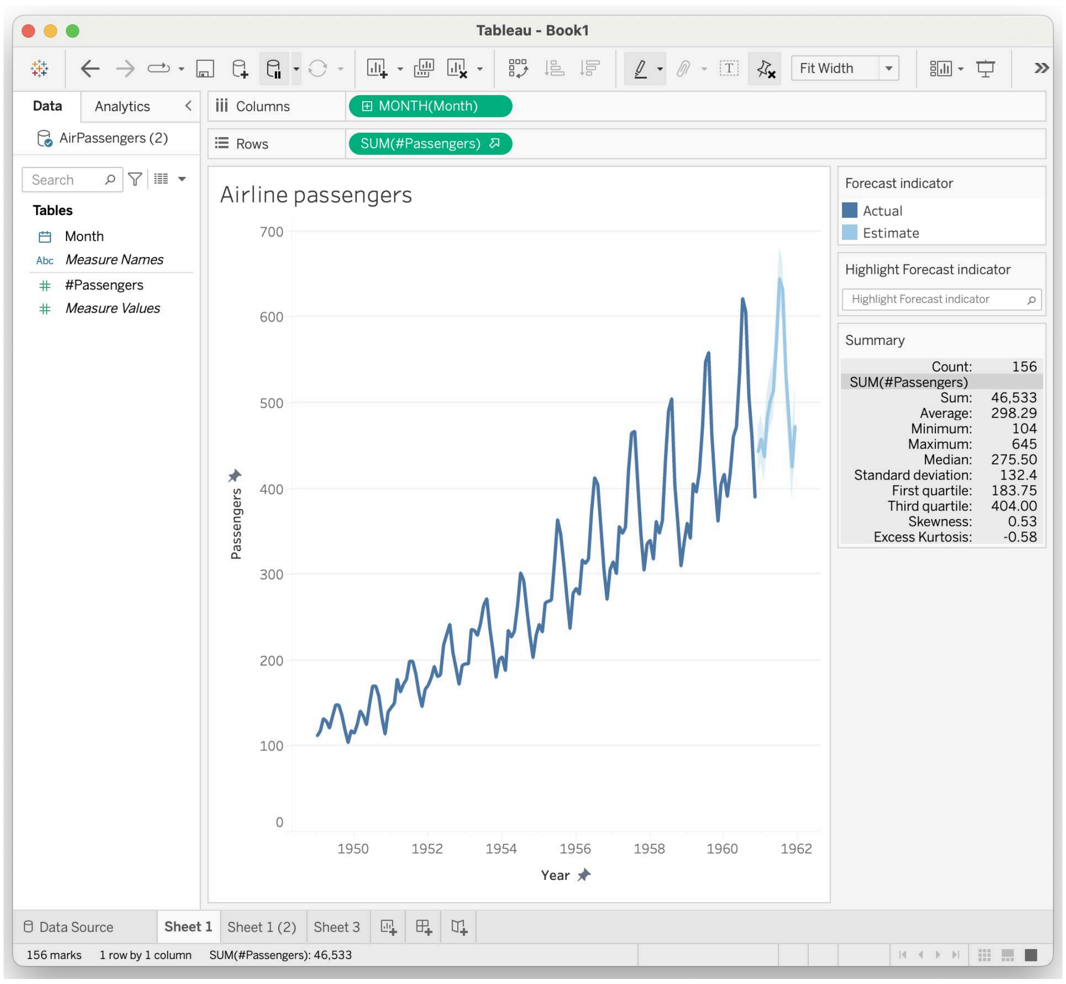
Tableau

Tableau is a suite of visual analytics products owned by Salesforce. Primarily, it provides an easy-to-use, self-service analytics cloud platform designed to deliver insights. The new company's product for business users, Tableau Pulse, brings analytics to a new level by generating proactive answers to customers' common business questions with automated insights. With Tableau Pulse, the platform can automatically detect trends, outliers, and many more interesting findings for the users' metrics. Moreover, this system is an excellent example of user guidance in the analytics process. It processes its findings in natural language using AI models and presents them to users as guided conversations or summaries in the UI. In the former case, a user types a descriptive question into an input box. The system provides a textual response and visualizations based on the known data. In the latter case, users see the "Insight summary" block, which helps them understand the important changes to the Pulse Metrics they are following. Insight summaries can be shown on the dashboards or sent via Slack or email [43].

For external usage, Tableau provides Embedded Analytics APIs and developer tools. They enable the customers to integrate Tableau analytics into their own products and applications quickly.

Tableau offers two options for working on visual analytics dashboards: Tableau Cloud and Tableau Desktop. Tableau Desktop application is available for Windows and Mac [44].

Regarding the prediction tasks, Tableau provides several methods to perform and visualize the forecasts. Forecasting in Tableau uses a technique called exponential smoothing. Namely, the algorithms aim to find a repetitive pattern in measures that can be continued further in the future [46]. A way to get a fast forecast is to right-click the visualization of interest and select "Forecast" out of the available options. To predict the data, one must have at least one date or integer dimension and one measure in the



Forecast indicator
- 🟦 Actual
- 🟦 Estimate

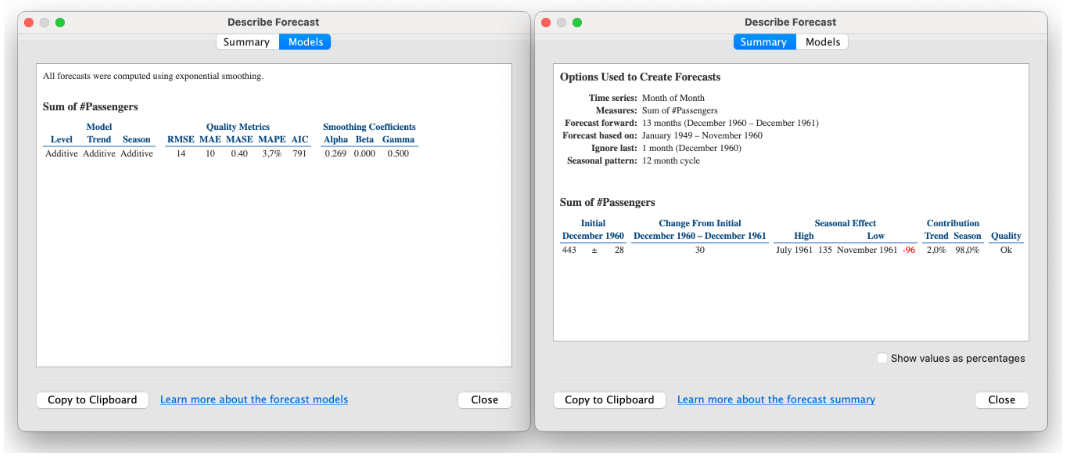
Summary

Count:	156
SUM(#Passengers)	
Sum:	46,533
Average:	298.29
Minimum:	104
Maximum:	645
Median:	275.50
Standard deviation:	132.4
First quartile:	183.75
Third quartile:	404.00
Skewness:	0.53
Excess Kurtosis:	-0.58

```
156 marks    1 row by 1 column    SUM(#Passengers): 46,533
```

Figure 2.2.: An example visualization with a forecast created using Tableau. The dataset used for this example is "Air Passengers" [45], with permission of usage granted under Database Contents License (DbCL) v1.0.

view. Users can configure forecast length and confidence interval. On save, the estimated future values of the measure will appear next to the actual values as a separate line on a chart, surrounded by an area denoting the confidence interval [47]. Tableau also provides a description of the forecast and of the model. There, among the other data, users can find information such as prediction error estimations and seasonality parameters. Figure 2.2 and Figure 4.9 display an example Tableau visualization with a forecast and



All forecasts were computed using exponential smoothing.

Sum of #Passengers

Model			Quality Metrics					Smoothing Coefficients		
Level	Trend	Season	RMSE	MAE	MASE	MAPE	AIC	Alpha	Beta	Gamma
Additive	Additive	Additive	14	10	0.40	3,7%	791	0.269	0.000	0.500

Options Used to Create Forecasts

Time series:	Month of Month
Measures:	Sum of #Passengers
Forecast forward:	13 months (December 1960 – December 1961)
Forecast based on:	January 1949 – November 1960
Ignore last:	1 month (December 1960)
Seasonal pattern:	12 month cycle

Sum of #Passengers

Initial	Change From Initial	Seasonal Effect				Contribution		
December 1960	December 1960 – December 1961	High		Low		Trend	Season	Quality
443 ± 28	30	July 1961	135	November 1961	-96	2,0%	98,0%	Ok

Figure 2.3.: A description of a forecast created using Tableau. The dataset used for this example is "Air Passengers" [45], with permission of usage granted under Database Contents License (DbCL) v1.0.

its description correspondingly.

A more sophisticated method should be used if a user wants to perform advanced statistical calculations and get more control over specific parameters. One option is to integrate Tableau with R and Python, perform statistical calculations, and then visualize them in Tableau. Another option is to use predictive modeling functions [48] to make forecasts from your data by including them in a table calculation. This functionality lets you choose targets and predictors by updating the variables. Further, you can visualize multiple models with different combinations of parameters. If you manipulate your data, for example, filter or apply aggregations or transformations, the prediction will be automatically recalculated. Predictive modeling functions offer linear regression, regularized linear regression, and Gaussian process regression.

The advantage of Tableau forecast functionality is that it offers simple and advanced forecast generation options. If the user selects a simple option (by selecting "Forecast" from the context menu), they can see the prediction quickly without needing expert statistical knowledge. Another advantage is that Tableau is very flexible regarding the visualization types to which forecasting can be applied. A user can get the forecast visualized on the line, area, and bar charts, as well as on more complex charts, such as packed bubble charts or treemaps.

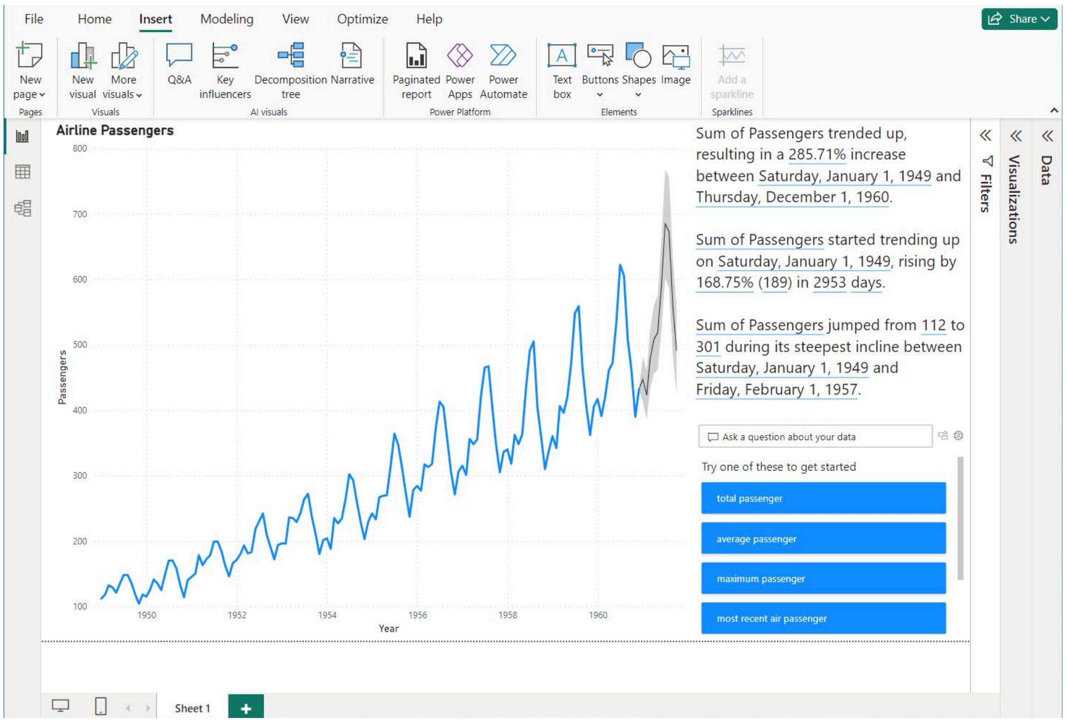
The disadvantage of Tableau is that in the web version, a user can only publish a view with forecast and see the forecast in display or edit mode. Adding or modifying the forecast in the web version is not possible. In addition to that, the multidimensional data sources are only possible to add in Tableau Desktop. Forecasting for multidimensional data sources is not supported [49].

Power BI

Power BI is a suite of business analytics and visualization tools developed by Microsoft. It offers the ability to create dashboards and reports employing interactive visualizations and business intelligence capabilities.

Power BI offers two tools for visual analytics dashboards: Power BI Service and Power BI Dashboard. Power BI Service is a cloud-based application. Its main usage purposes are sharing, collaboration, and distribution of reports and dashboards. In contrast, Power BI Desktop is a development tool for creating, modeling, and designing reports and visualizations.

Power BI allows forecasts to be applied to a data source if it contains time data. Currently, the forecasting option is only available for line chart visuals [50]. To change a forecast, a user can update the its length and the confidence interval or configure the seasonality. After the forecast is generated, it is visualized on the chart as a line, with the colored area around it depicting the confidence interval. In addition to that, one can integrate a smart narrative summary [51] and Q&A component into the visuals. They provide functionalities similar to Tableau Pulse. The smart narrative summary displays the generated insights about the visuals. They address key takeaways and point out trends. In the Q&A component, users can ask questions about the data in natural language, and Power BI will generate answers. Figure 2.4 displays the example Power BI visualization with a forecast, smart summary and Q&A component.



Sum of Passengers trended up, resulting in a 285.71% increase between Saturday, January 1, 1949 and Thursday, December 1, 1960.

Sum of Passengers started trending up on Saturday, January 1, 1949, rising by 168.75% (189) in 2953 days.

Sum of Passengers jumped from 112 to 301 during its steepest incline between Saturday, January 1, 1949 and Friday, February 1, 1957.

Figure 2.4.: An example visualization with a forecast created using Microsoft Power BI. A blue line displays the actual data, while a grey line displays the prediction. A summary block with AI-generated insights is displayed on the right side of the chart. Below it, a question block providing AI-generated answers is placed. The dataset used for this example is "Air Passengers" [45], with permission of usage granted under CC0: Public Domain license.

The benefit of Power BI is a user-friendly interface, which is easy-to-understand for beginner users. As in Tableau, users are able to create a quick forecast without the need for extensive statistical knowledge. Moreover, they can integrate the smart narrative summaries and Q&A components into the interface in a few clicks.

The huge disadvantage, however, is that the desktop application only works on Windows operational systems. At the same time, the online service only provides some of the extensive functionalities of Power BI Desktop. Another disadvantage is that users can only apply predictions to line charts, unlike in Tableau, where more charting options are available to visualize forecasts.

3. Concept and Requirements

The idea of a predictive visual analytics system accommodates two critical components: predictive modeling and interactive visualization. Predictive modeling enables forecasting future values based on historical data. Visual analytics facilitates the exploration and insight extraction through dynamic and informative user interfaces. The synergy between these two components allows users to not only view the predictions but also investigate the patterns, make conclusions about the model performance, and adjust the parameters to make it better.

Several general and product-specific objectives follow from the abovementioned functions and the project's research questions:

- to enable users to get a better understanding of their data through a variety of time series characteristics offered in the system,
- to allow users to predict future data efficiently and with a minimum cognitive load,
- to support users in analyzing the results through a set of visualization tools,
- to enable experts and beginners in data science to make efficient forecasts of their data by leading them through the process.

With the system objectives in mind, two implementation pillars must be clarified: the practical system requirements and the integrated scientific foundation. They will be discussed in-depth in the following sections.

3.1. Practical system requirements

Requirements ensure that a system meets the users' needs while preserving high performance and reliability standards. They cover functionality, user interaction, technical details, and other aspects.

3.1.1. Functional requirements

Functional requirements include various categories of product features and details that should be implemented to enable users to accomplish their tasks. A key consideration during the development of the requirements was to ensure the app interface is user-friendly and accessible for beginners. The requirements list was also influenced by the most critical features identified in related works within the field. Since this work aimed to develop a minimum viable product to answer the research questions, some standard system requirements (e.g., user authentication) were intentionally omitted.

Data ingestion and processing

- The system should support two commonly used dataset formats: CSV and JSON.
- The system should allow the users to select the data property acting as a time indicator and the data properties for the analysis.
- The system should support both univariate and multivariate datasets.
- The system should be able to perform time series transformations, namely, differencing and scaling, before applying predictive models.

Not a part of data processing requirements:

- Data preprocessing, such as handling missing data and outliers, is left out of the scope since it is an extensive topic that can be an independent project in itself.

Predictive modeling

- The system should offer at least two predictive models for selection: one for univariate dataset prediction and another for multivariate dataset prediction.
- The system should offer a preliminary analysis of the data which precedes and enables the predictive modeling process.
- The system should provide information about the prediction quality, for example, error metrics.
- The system should provide the capabilities for model customization (tuning the parameters, selecting a prediction horizon of choice, and other options).

- The system should provide the capabilities for easy comparison of different models and their parameters.
- The system should provide the opportunity to review the results of previous predictions and the parameters.
- The system should allow users to select a particular period in data for analyzing and forecasting.

Visualization

- The system should provide an interactive time series plot for data and prediction investigation.
- The system should support the visual comparison of the actual versus predicted data.
- The system should provide the ability to visually explore smaller chunks of data and individual data points in-depth.
- The system should enable the users to visually compare the trends in different variables of a multivariate dataset.

3.1.2. Non-functional requirements

Non-functional requirements specify various system aspects that are not directly related to the core functionality. The main non-functional requirements in PredictPal include such aspects:

- The system should provide an intuitive user interface that guides the user through the process of data analysis and prediction.
- The system should need a low learning curve which allows the beginner users to exploit the system easily.

3.1.3. Technical requirements

Technical requirements consist of the technical aspects that must be considered to complete the PredictPal project. These are:

- The system should be implemented as a web application.
- The implementation should follow a modular code design to allow for easy updates and maintenance.

Not a part of technical requirements:

- mobile support,
- responsive design for small screen sizes.

3.2. Scientific foundation

In the realm of predictive analytics systems, scientific concepts, such as predictive models and the theory of statistics, act as a backbone of a system's analytical capabilities. They open the opportunities to understand the data behavior and forecast the patterns.

Prior to using any kind of predictive model, users need to perform a preliminary analysis to gain a better understanding of the data. The following subsections give an overview of the aspects of data analysis employed and the predictive models used.

3.2.1. Data analysis

Before proceeding to the predictive modeling, users should normally be able to answer the following questions:

- Does the data describe a stable process, or does the character of the process change over time?
- Is the data predictable at all, or is it a set of entirely random points?
- If it is multivariate time series data, do the variables expose any causal relationships? If yes, which variables?

To enable answering these questions, PredictPal offers a set of statistical tests performed on the server. Using the available statistical tests, one can analyze such information:

- stationarity,
- white noise,
- causality.

Stationarity. Stationary time series are time series representing a process that remains in statistical equilibrium. The probabilistic properties of such a process remain stable over time. Specifically, the sequence of data points varies around a fixed constant mean and a constant variance [52].

Stationarity is a fundamental concept in predictive analytics. Multiple prediction models assume that the data they predict are stationary to allow for accurate forecasting. In the real world, however, many processes are unstable. Natural phenomena like climate patterns or population dynamics are influenced by various factors that lead them to exhibit trends or shifts in variability. Financial markets are driven by economic cycles, regulatory changes, and market sentiment, which makes them prone to price fluctuations and volatility. Similarly, in biology, economics, social sciences, or engineering, the dynamic nature of multiple systems makes them likely to exhibit nonstationary behaviors.

PredictPal verifies the stationarity of non-seasonal data using two different tests:

- Kwiatkowski–Phillips–Schmidt–Shin (KPSS),
- Augmented Dickey-Fuller (ADF).

KPSS test checks such hypotheses:

$$\begin{aligned} H_0 &: \text{The series is stationary} \\ H_1 &: \text{The series is not stationary} \end{aligned}$$

ADF test verifies such hypotheses [53]:

$$\begin{aligned} H_0 &: \text{The process contains a unit root and therefore it is nonstationary} \\ H_1 &: \text{The process does not contain a unit root and is stationary} \end{aligned}$$

The system verifies the seasonal stationarity using Canova-Hansen (CH) test. CH test verifies such hypotheses [54]:

$$\begin{aligned} H_0 &: \text{The series exhibits stationary seasonality} \\ H_1 &: \text{The series exhibits nonstationary seasonality} \end{aligned}$$

As mentioned, multiple models rely on the fact that the processes they predict are stationary by default. If it is true, the model can be applied directly to the data. In case the process is nonstationary, the prediction steps look as follows:

1. Transform the data to achieve stationarity.
2. Apply the predictive model to the stationary data.
3. Convert the predicted data back to its original nonstationary form.

One of the approaches to achieve stationarity is to repeatedly apply differencing to the original series until the differenced observations take the form of a stationary time series [52], [55]. To remove the trend from the time series that does not exhibit a seasonal pattern, we can perform $lag - 1$ differencing. We define the $lag - 1$ difference operator ∇ as following [55]:

$$\nabla X_t = (1 - B)X_t = X_t - X_{t-1},$$

where ∇X_t is the differenced data and B stands for the backshift operator:

$$BX_t = X_{t-1}.$$

To deal with the data seasonality of a period d, one needs to adjust the differencing formula by introducing $lag - d$ differencing operator ∇_d [55]:

$$\nabla_d X_t = (1 - B^d)X_t = X_t - X_{t-d},$$

where B^d represents a periodic backshift operator and the value X_{t-d} represents the value d periods ago. The stationarity tests in PredictPal are meant to give users a better understanding of their data. If a sequence needs to be converted to stationary, the app will do it under the hood before applying the predictive models.

White noise test White noise is the most fundamental example of a stationary process. It consists of a sequence of independent and identically distributed random variables $a_1, a_2, ..., a_t, ...,$ which have zero mean and constant variance σ_a^2 [52]. White noise immediately forgets its past values, making its future unpredictable.

To check whether the data is white noise, PredictPal runs the Ljung-Box test of autocorrelation. Its hypotheses are [56]:

$$H_0 : \text{The series is white noise,}$$
$$H_1 : \text{The series is not white noise.}$$

Causality test Causality is checked by running the Granger causality tests.

Time series Y_t is said to Granger cause time series X_t if the minimum predictive error variance of X_t at present can be reduced by including the past values of both Y_t and X_t, compared to only including the past values of X_t [57]. In other words, Y_t causes X_t, if

$$\sigma^2(X|\overline{X}) > \sigma^2(X|\overline{X}, \overline{Y}),$$

where $\overline{X}$ and $\overline{Y}$ represent the past values of X_t and Y_t correspondingly, $\sigma^2(X|\overline{X})$ stands for the minimum predictive error variance of X_t based only on the $\overline{X}$, and $\sigma^2(X|\overline{X}, \overline{Y})$ is a minimum error variance when both $\overline{X}$ and $\overline{Y}$ are used in prediction of X_t.

The hypotheses of a typical Granger causality test are as follows:

$$H_0 : \text{The series } Y \text{ does not Granger cause the series } X,$$
$$H_1 : \text{The series } Y \text{ Granger causes the series } X$$

3.2.2. Predictive models

The first version of PredictPal was planned as a minimum viable product to effectively demonstrate the system's capabilities with a focus on visual analytics. Therefore, the initial version includes foundational predictive models known for their simplicity and widespread availability. Specifically, the ARIMA (AutoRegressive Integrated Moving Average) and VAR (Vector Autoregression) models are offered for selection. Such a set of models allows users to forecast both univariate and multivariate datasets, which aligns well with the app's functional requirements. In the future, the system may incorporate more models for selection.

ARIMA

ARIMA is a widely used statistical method of forecasting linear univariate time series data. It consists of three parts: Autoregressive, Integrated, and Moving average.

Autoregressive (AR). AR part describes a value a_t as a function of p previous values $a_{t-1}, a_{t-2}, ..., a_{t-p}$ of the process. Here, p stands for the number of steps into the past needed to predict the next value. The form of the autoregressive model AR(p) is as follows:

$$a_t = \phi_1 a_{t-1} + \phi_2 a_{t-2} + \cdots + \phi_p a_{t-p} + w_t,$$

where a_t is stationary, $\phi_1, ..., \phi_p$ are the constants and w_t models white noise in the data [58].

Integrated (I). ARMA models can predict only stationary data. The integrated component extends them to enable handling of nonstationary processes. It involves differencing time series data d times to make it stationary. Usually, d is 0, 1, or at most 2, with $d = 0$ corresponding to stationary behavior.

Moving average (MA). This component describes a value a_t as a linear combination of past forecast errors. The moving average model of order q is of the following form:

$$a_t = w_t + w_{t-1}\theta_1 + w_{t-2}\theta_2 + ... + w_{t-q}\theta_q,$$

where $w_t, ..., w_{t-q}$ are past forecast errors, q is the number of past forecast errors to consider and $\theta_1, ..., \theta_q$ are non-zero parameters[58].

VAR

VAR (Vector Autoregressive Model) is one of the standard models for multivariate time series prediction. It extends the traditional AR (Autoregressive) model for univariate data forecasting. In the VAR model, the forecast of each variable builds as a linear function of the previous values of all data variables. The key assumption of the VAR model is that the predicted process is stationary [59]. If data exhibits nonstationary behavior, one can transform it by applying differencing and afterwards continue the modeling. The prediction then has to be converted back to the nonstationary form at the end of the process.

The basic p-lag model VAR(p) for stationary data is defined as following [60]:

$$Y_t = c + M_1 Y_{t-1} + M_2 Y_{t-2} + ... + M_p Y_{t-p} + \sigma_t, \quad t = 1, 2, ..., T \qquad (3.1)$$

where t is a point of time, M_i stand for $(n \times n)$ coefficient matrices, σ_t is a $(n \times 1)$ white noise vector process and c is an $(n \times 1)$ vector of constants.

The appropriate lag order p for the $VAR(p)$ can be identified using the model selection criteria. To do this, one has to select a maximum reasonable lag p_{max}, fit $VAR(p)$ models with lag orders $p = 0, ..., p_{max}$ and select the value which minimizes a particular model selection criteria. The most common information criteria include the Akaike (AIC) and Schwarz-Bayesian (BIC) [60].

There are different opinions regarding the maximum number of variables VAR can predict. Gorgi et al. [61] state that usually, the VAR model does not include more than five to six variables since, otherwise, the number of parameters becomes too large. Bernanke et al. [62] indicate that VAR models almost never predict more than six to eight variables to conserve degrees of freedom.

Statistical errors

The standard way to evaluate the prediction quality numerically is to calculate the statistical error between the actual test data and the test prediction. As Hyndman states [63], statistical error metrics include such categories as scale-dependent errors, percentage errors, and others. Scale-dependent errors are those errors that are on the same scale as data. They include such metrics as mean absolute error (MAE) and the root mean square error (RMSE). Percentage errors have the benefit of being independent of a scale. Therefore, they are often used to compare the forecast quality between the different datasets. The most common percentage error metric is the mean absolute percentage error (MAPE).

The initial decision was to use the percentage error metrics. These errors would offload the need to keep the data scale in mind from the users. However, the percentage error metrics have a disadvantage: they become undefined or infinite if $a_t = 0$ for any observation of the test data set. They also have extreme values in case any a_t is close to zero [63].

Since PredictPal aims to act as a universal system for time series prediction, the nature of the data cannot be known upfront. Therefore, a situation might occur when the percentage error metrics may not be estimated. For that reason, a decision was made to present two scale-dependent errors to the users: MAE and RMSE. Their formulas are as following [63]:

$$MAE = mean(|e_i|),$$

$$RMSE = \sqrt{mean(|e_i^2|)},$$

where e_i represents the error for each individual data point i, namely, the difference between the actual observed value y_i and the predicted value $\hat{y}_i$:

$$e_i = y_i - \hat{y}_i.$$

4. Implementation

PredictPal is a full-stack web application with the following core components:

- *Database*: The storage of the time series data as well as the history of its predictions
- *Backend*: Manages the time series data operations, analysis, and prediction and returns the results to the frontend for further displaying
- *Frontend*: The interface that enables the users to upload their data, visualize it, and interact with the system to get the analysis results

The overview of the architectural components of the project and their relations is displayed in Figure 4.1.

The first section of this chapter explains the key decisions regarding the project's stack. It is followed by a brief description of the database and the backend components. Further, an in-depth explanation of the main frontend components and frontend pages are given.

4.1. Key decisions about project stack

The initial decision for the project was to implement an application based on JavaScript technologies: NodeJS on the backend and JavaScript on the frontend. One of the benefits of such an approach was that JavaScript opens the door to a vast ecosystem of libraries and tools designed for building dynamic and highly interactive user interfaces with performance in the focus. Prompt performance and interactivity were crucial for an application like PredictPal, aiming to visualize the data and the analysis results. Additionally, using JavaScript for both server and client sides was intended to streamline the development process. This approach would facilitate the reuse of utilities and simplify context switching between server-side and

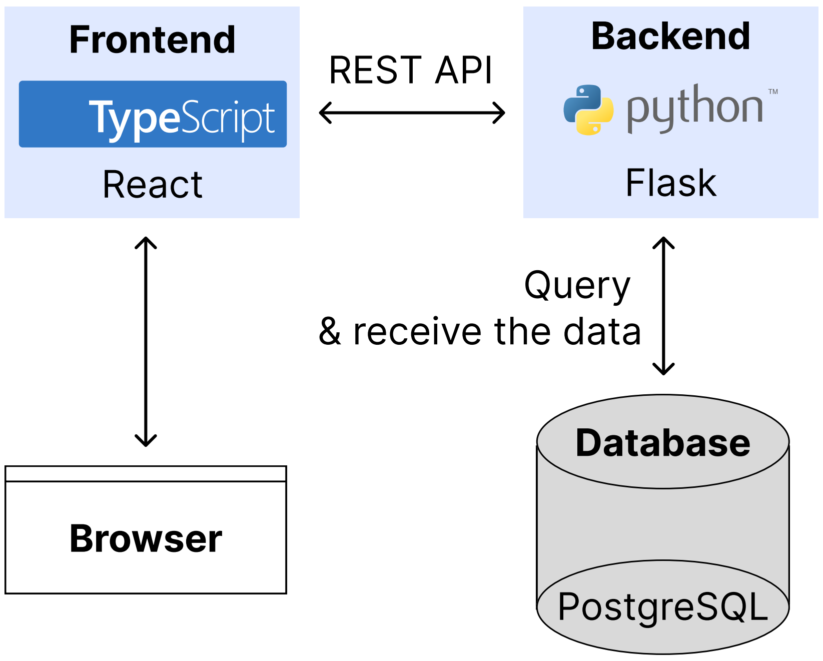


Figure 4.1.: System architecture components and their relations

client-side development.

Unfortunately, this decision proved inefficient for the PredictPal project's specific goals. The application required libraries for statistical tests, utilities, and prediction models. The preliminary investigation showed that there exist some NodeJS libraries that port statistical test packages from R, C, or Python. As an example, there is an ARIMA library for NodeJS and browsers [64]. Nevertheless, the number of such libraries is limited, and some methods necessary for the project are unavailable. In addition, there are considerably fewer community discussions related to the implementations of statistical tests and predictive modeling using NodeJS. Consequently, finding the answers to some questions or receiving community support is harder. Therefore, the decision was made to implement the project with Python on the server and JavaScript on the client side. This approach allowed the use of powerful and popular statistics libraries available in Python on the server side while preserving the benefits of the JavaScript ecosystem for building the client side.

Further, it was decided to switch from JavaScript to TypeScript language. TypeScript is a strongly typed programming language that extends JavaScript [65]. TypeScript was chosen due to its ability to minimize common errors such as misuse of variables. It not only enhances code readability but also facilitates a faster development process by preventing avoidable mistakes [65].

The project was initialized using react-flask-hello [66], a public web app boilerplate incorporating React on the frontend and Flask API on the backend. The owner of the boilerplate is 4Geeks Academy. Their boilerplate assembles an initial project structure ready for running and further implementation. It also provides common development guidelines.

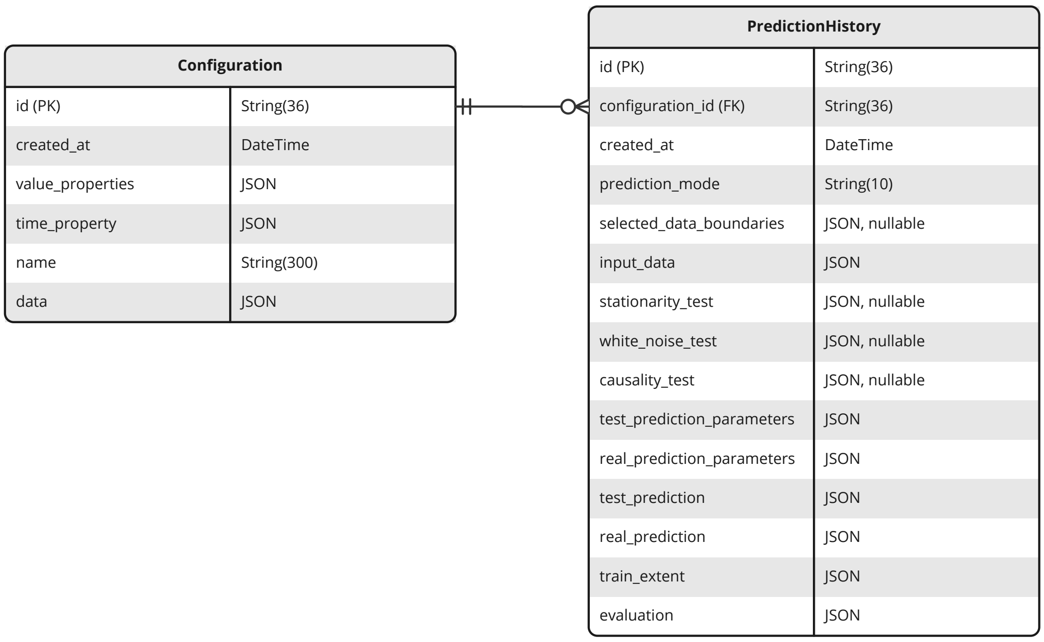
4.2. Database

To store the data, including the time series datasets, PostgreSQL [67] relational database was used. One of the reasons for choosing PostgreSQL was that PredictPal operates around time series datasets that are converted to JSON format. PostgreSQL has robust out-of-the-box support for JSON data. Specifically, it provides JSON and JSONB column types, allowing efficient storage, indexing, and querying of JSON data. In addition, PostgreSQL conforms to SQL standards, lowering the learning curve for potential project joiners.

There are two database models in the project (Figure 4.2):

- *Configuration* - contains the dataset in JSON form, as well as its selected time property name, names of the properties to analyze, and the dataset name.
- *PredictionHistory* - keeps the prediction results of a specific configuration, the name of a model used, the limits of training data, and the results of statistical tests (if any were performed before the prediction was made).

While some fields of JSON format could be extracted to become separate models themselves, the approach of having only two tables was sufficient



Configuration	
id (PK)	String(36)
created_at	DateTime
value_properties	JSON
time_property	JSON
name	String(300)
data	JSON

PredictionHistory	
id (PK)	String(36)
configuration_id (FK)	String(36)
created_at	DateTime
prediction_mode	String(10)
selected_data_boundaries	JSON, nullable
input_data	JSON
stationarity_test	JSON, nullable
white_noise_test	JSON, nullable
causality_test	JSON, nullable
test_prediction_parameters	JSON
real_prediction_parameters	JSON
test_prediction	JSON
real_prediction	JSON
train_extent	JSON
evaluation	JSON

Figure 4.2.: Database models overview

for the implementation of a minimum viable product.

4.3. General backend components

The backend was implemented using Flask, the Python micro framework for building web applications. One of the advantages of Flask is that it provides guidance but does not enforce the project layout or the tooling that the developers should use [68].

The backend routes can be divided into a few categories:

- routes handling the database CRUD (Create, Read, Update, Delete) operations to manipulate the dataset and prediction history objects,
- routes handling statistical tests and prediction models.

4.3.1. Predictive models

For prediction using ARIMA models, pmdarima [69] statistical library was used. The reason for selecting this library is that it provides the equivalent of R's *auto.arima* functionality and allows automatic detection of some parameters. For the development of the beginner-friendly PredictPal app, the ability of the model to detect some parameters internally instead of requiring them from the users was a huge benefit.

For prediction using the Vector Auto Regression (VAR) model, the statsmodels Python module was used [70]. It supports Pandas data frames and provides multiple statistical tests. Those of interest to us were the Ljung-Box test of autocorrelation and the Granger causality test.

Additionally, pmdarima provides the time series utilities and a collection of the statistical tests. The stationarity testing in PredictPal was implemented using pmdarima functions, namely *ndiffs* and *nsdiffs*, since they require minimum obligatory arguments to derive the number of differences needed. Such simplicity supported the goal of making PredictPal straightforward and friendly to beginners.

4.4. General aspects of frontend implementation

As mentioned above, the project's frontend was implemented using a client-side library called React [71]. The reason for using this library is that React is more performant than plain JavaScript or TypeScript. A great advantage of React is fast component rendering. It is achieved through the usage of Virtual DOM [72] instead of performing costly rerendering operations on the real DOM directly. A virtual DOM is a lightweight copy of a real DOM where a virtual representation of UI is kept in memory. React uses a diffing algorithm to compare the previous and the new versions of Virtual DOM. Then, it automatically updates only those elements in the real DOM that have changed instead of rebuilding the whole DOM tree.

In addition, one of the significant benefits of React is its modular design. It combines well with PredictPal's technical requirement: "The implementation should follow a modular code design to allow for easy updates and maintenance." React lets users create user interfaces out of individual parts called components, making it easier to structure and reuse the code and increase readability.

Moreover, there exists a vast ecosystem of libraries cut out to work with React. It is also worth mentioning that the React developer community is large and active.

4.4.1. State management

For the global state management, Zustand library [73] was used. Zustand is a small, fast, scalable solution based on an immutable state model.

4.4.2. Utility libraries

For support in general tasks such as manipulating arrays, objects, strings, and other data types, Lodash utility library [74] was used.

For date formatting, PredictPal employed date-fns library [75]. The advantage of date-fns, compared to other popular date libraries, such as Luxon, is that it is highly modular. The developers can import only the functions they need, which leads to a smaller bundle size than downloading the whole library. In addition to this, date-fns is consistent and reliable. It contains pure functions and always returns a new date instance instead of changing the passed value.

4.4.3. User Interface Styling

Building and styling one's own components for a project is resource-consuming. It requires diligence in unifying the behaviors between similar components, making them flexible, and considering such important aspects as accessibility. Therefore, the decision was made to avoid custom component implementation, favoring the available component systems.

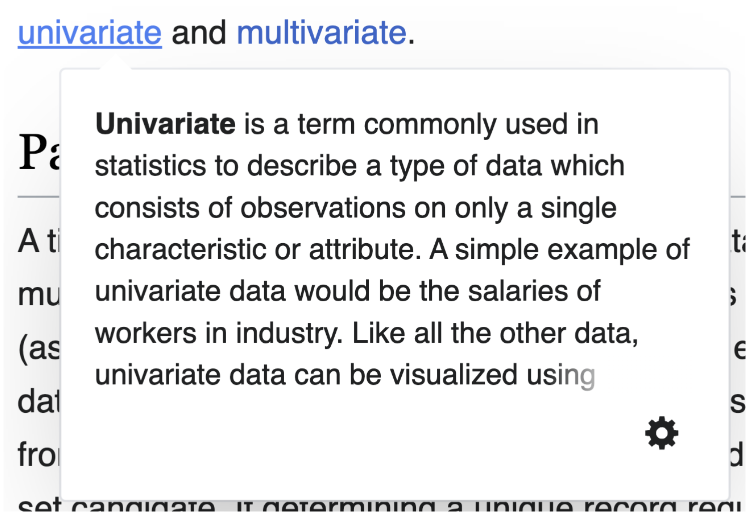
Google's Material Design [76] system was employed to provide a unified and clean user interface appearance. Material Design is a configurable system containing components, tools, and guidelines that integrate the UI best practices. One of the core values of this system is accessibility by design. Material Design offers enhanced usability for users with visual and hearing impairments. Another benefit is the default availability of design tokens for color palettes, fonts, and measurements. Material Design is easy to integrate into React code since Google offers an open-source React component library Material UI [77] with ready-to-use components.

4.4.4. Explanatory Texts

One of the goals of the PredictPal project was to make predicting easy for non-expert users. Prediction models and their parameters might be difficult to understand for people who do not work with data science problems daily. One way to improve the app's user experience and make the prediction process easier to understand was to add explanatory texts explaining the various data science concepts. The idea was that the explanatory texts should be available on demand so that they do not clutter the default app view. In such a way, those unfamiliar with a particular topic or first-time users can receive additional information. At the same time, the experienced users can ignore the explanations and start analyzing the data right away.

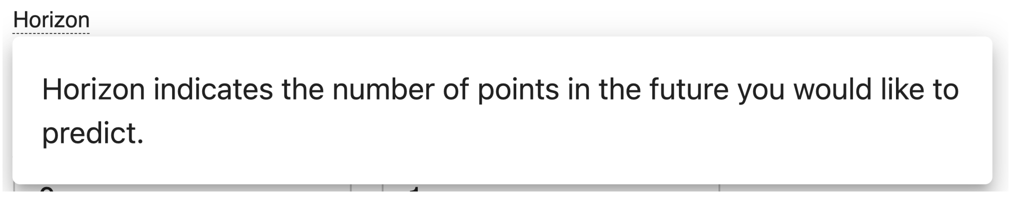
The implementation of the explanatory texts was inspired by Wikipedia (example depicted on Figure 4.3). A term that needs the explanations is highlighted to signify that it is interactive. When clicking on the text, an information popover appears containing the explanations (in Wikipedia, the interaction happens on hover). The example information popover from the PredictPal app is depicted on Figure 4.4.

Additionally, some users may already be familiar with data science terminology, so they can easily understand the explanations from the scientific papers. However, there might also be users who do not know the data science "language." Such users might find the explanatory excerpts from the scientific papers more confusing than helpful. With this in mind, it was decided to offer two styles of explanation: advanced and beginner-friendly. The


univariate and multivariate.

Univariate is a term commonly used in statistics to describe a type of data which consists of observations on only a single characteristic or attribute. A simple example of univariate data would be the salaries of workers in industry. Like all the other data, univariate data can be visualized using

Figure 4.3.: Example information popover from Wikipedia containing the explanatory text to the word "univariate." The popover appears when hovering over the word. The screenshot was taken in the Wikipedia article "Time series" [78].


Horizon

Horizon indicates the number of points in the future you would like to predict.

Figure 4.4.: Example information popover from PredictPal containing the explanatory text to the word "horizon." The popover appears on click on the word.

advanced explanations were directly cited from the scientific papers, catering to those already comfortable with the material. The beginner-friendly versions included more straightforward, extensive explanations with simpler language and multiple examples. For the generation of beginner-friendly texts, the OpenAI GPT-4o model was employed. GPT-4o ("o" stands for "omni") model is a generative pre-trained large language model (LLM), which can accept as input any combination of text, image, audio, and video and generate text, audio, and image outputs [79]. The model was given an already available advanced explanation with such a prompt:

"The browser app I develop provides an easy-to-use beginner-

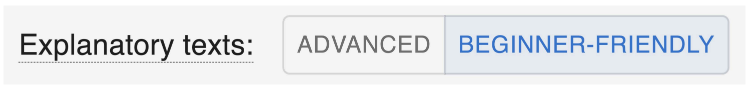


Figure 4.5.: Toggle button group to switch between advanced and beginner-friendly explanation in PredictPal. The button group is placed in the app's header.

> friendly interface for time series data analysis and prediction. Please provide a simple version of the following text in a way that is clear for beginners. Keep a similar structure of the text. Provide examples and easy comparisons to real-world processes where needed. Here is the text: <text>."

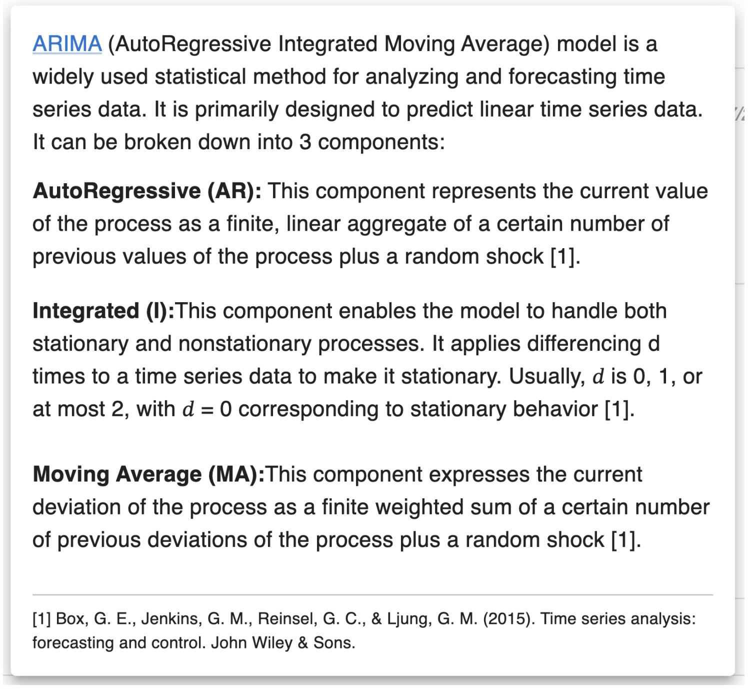
Further, the received text was manually verified and refined. The users can switch between advanced and beginner-friendly explanation types using a toggle button in the app's header (Figure 4.5). An example of an advanced and beginner-friendly information popover is provided in Figure 4.6.

4.5. Routing and pages

Routing in the client-side application is the process of navigation management between various views within the application. React apps, including PredictPal, are usually built as Single Page Apps (SPA). It means that the entire application is loaded initially, and the navigation between different routes does not require a full page reload. Instead of that, the main HTML document remains intact, and the application dynamically updates the views based on the routes or user interaction.

The routing in PredictPal is managed using the React Router library [80]. The app consists of three primary client routes, each dedicated to distinct functionalities related to uploading, managing, and analyzing datasets:

- Dataset upload route provides a form for the dataset upload.
- Datasets list route gives an overview of all the uploaded datasets.
- Dataset analysis route provides dedicated statistics analysis and predictive modeling functionalities.



ARIMA (AutoRegressive Integrated Moving Average) model is a widely used statistical method for analyzing and forecasting time series data. It is primarily designed to predict linear time series data. It can be broken down into 3 components:

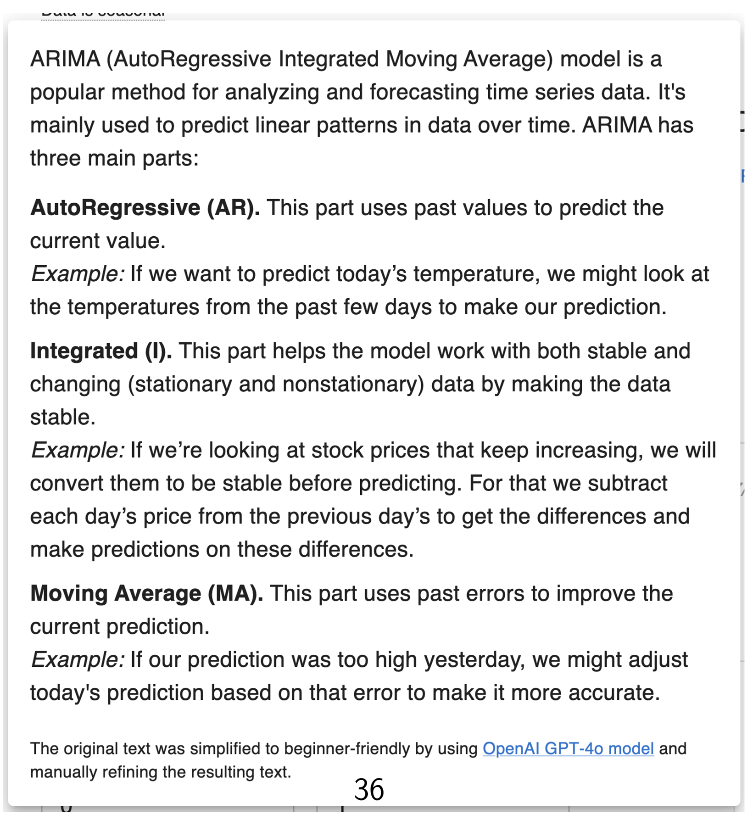
AutoRegressive (AR): This component represents the current value of the process as a finite, linear aggregate of a certain number of previous values of the process plus a random shock [1].

Integrated (I): This component enables the model to handle both stationary and nonstationary processes. It applies differencing d times to a time series data to make it stationary. Usually, d is 0, 1, or at most 2, with $d = 0$ corresponding to stationary behavior [1].

Moving Average (MA): This component expresses the current deviation of the process as a finite weighted sum of a certain number of previous deviations of the process plus a random shock [1].

[1] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.

(a) Advanced explanation



ARIMA (AutoRegressive Integrated Moving Average) model is a popular method for analyzing and forecasting time series data. It's mainly used to predict linear patterns in data over time. ARIMA has three main parts:

AutoRegressive (AR). This part uses past values to predict the current value.
Example: If we want to predict today's temperature, we might look at the temperatures from the past few days to make our prediction.

Integrated (I). This part helps the model work with both stable and changing (stationary and nonstationary) data by making the data stable.
Example: If we're looking at stock prices that keep increasing, we will convert them to be stable before predicting. For that we subtract each day's price from the previous day's to get the differences and make predictions on these differences.

Moving Average (MA). This part uses past errors to improve the current prediction.
Example: If our prediction was too high yesterday, we might adjust today's prediction based on that error to make it more accurate.

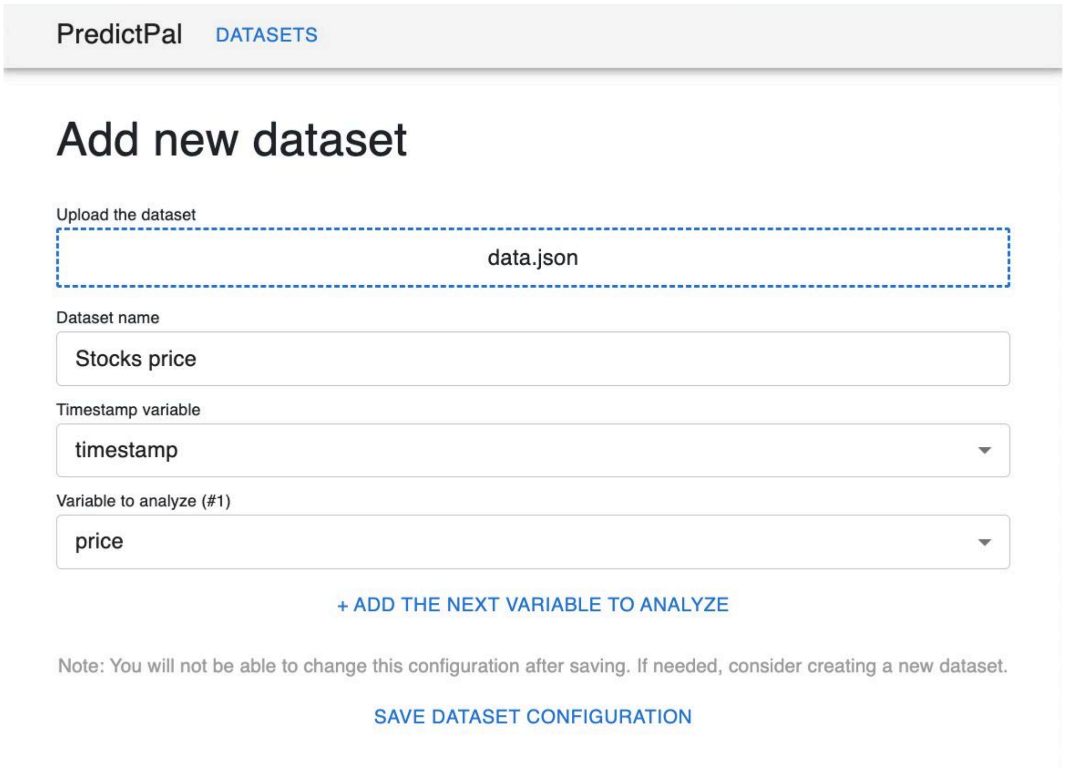
The original text was simplified to beginner-friendly by using OpenAI GPT-4o model and manually refining the resulting text.

(b) Beginner-friendly explanation

Figure 4.6.: Example of advanced and beginner-friendly information popovers explaining ARIMA model in PredictPal. The advanced explanation contains the definitions taken from scientific papers. Its alternative, the beginner-friendly explanation, is simplified using GPT-4o model. It contains more straightforward language and multiple examples.

4.6. Dataset Upload Interface

The dataset upload interface is where the data interaction starts. It provides a form for uploading the time series data of interest and selecting the variables to analyze (Figure 4.7).



PredictPal DATASETS

Add new dataset

Upload the dataset: data.json

Dataset name: Stocks price

Timestamp variable: timestamp

Variable to analyze (#1): price

+ ADD THE NEXT VARIABLE TO ANALYZE

Note: You will not be able to change this configuration after saving. If needed, consider creating a new dataset.

SAVE DATASET CONFIGURATION

Figure 4.7.: PredictPal dataset upload form accepts the datasets in CSV and JSON formats. The button "Add the next variable to analyze" allows users to define multiple variables of interest in case they upload multivariate datasets.

The application allows the data to be uploaded in CSV and JSON formats. Therefore, two different solutions were used concerning data parsing:

- JSON files: PredictPal uses the FileReader [81] interface of File API [82] for reading the files. The files are then parsed to JSON using the *JSON.parse()* method.
- CSV files: PredictPal uses Papaparse - an in-browser CSV parser, which

reads a provided local CSV file with the FileReader and converts it to JSON [83].

The form includes such fields:

- the file upload dropzone,
- the dataset name input,
- the select box for a timestamp field,
- the select box(es) for a field to analyze.

As soon as a user drags a file into the upload dropzone, the file gets parsed, and its available field names (properties of a JSON object or columns of CSV) get prefilled as the options of the field select boxes. In case the uploaded dataset is multivariate, more fields to analyze can be added by pressing a button under the select boxes.

To save the dataset configuration, a user needs to press "Save dataset configuration" button. For simplicity, in the first version of PredictPal, editing of an existing dataset configuration is not available.

4.7. Dataset Overview and Management Interface

The dataset overview interface provides the list of the uploaded dataset cards (Figure 4.8). Each card contains a name of a dataset and a deletion button on its right side. With a click on a card, a user is navigated to the dataset's analysis page.

4.8. Data Analysis and Predictive Modeling Interface

The data analysis page provides a detailed dataset overview and a set of analytics tools. The page consists of a few main blocks. The central spot of the view displays a large time series chart. It is followed by the analysis section containing two parts: the "Get to know your data" section offering preliminary data analysis, and the "Prediction" section. In addition to these always visible sections, a history section can be opened on demand in a sidebar.

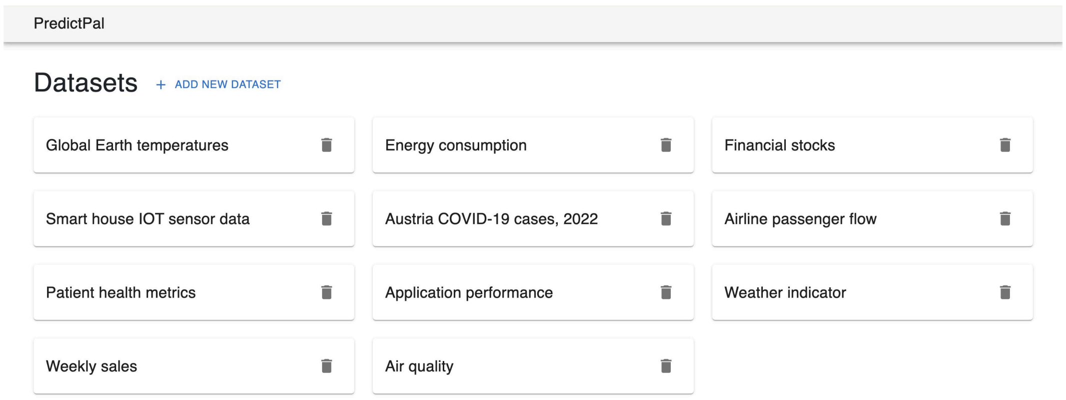


Figure 4.8.: PredictPal datasets list page. On click on a specific dataset card, the dataset's analysis page opens. On click on a dataset card's trash bin icon, the dataset is deleted.

4.8.1. Time series visualization

There are multiple ways to visualize time series data. Since PredictPal aims to provide a clear and straightforward interface, it was decided to select the line chart as the main visualization.

The visualizations were implemented using Visx library [84]. Visx is a wrapper over the state-of-art JavaScript visualization library D3 [85], which provides a collection of low-level visualization primitives for React, such as Axis, Shape (for example, Line, Area, or Bar), and others. One of the advantages of visx over the other visualization libraries is that it is positioned as "not a charting library." By combining the visualization primitives, one basically builds their charting library optimized for their specific use case [84].

Visx does provide an alternative to low-level primitives - a high-level visualization component called XYChart. This component abstracts the complexity of common visualization engineering and allows the quick creation of a line-, bar-, or similar chart. However, since the time series visualization in PredictPal required multiple custom functionalities, such as brushes, data selection, and data region mapping, the flexibility offered by low-level chart components was more beneficial. Also, the option with low-level visualization components is significantly easier to extend if needed.

Specifically, such visx primitives were used in the implementation:

- AxisBottom and AxisLeft (@visx/axis package) - the axis components with the scale ticks.
- GridRows and GridColumns (@visx/grid package) - the grid components providing the visual "skeleton" of the chart.
- LinePath (@visx/shape package) - the component representing a data line going through a set of points. It is used to plot the lines for the actual and prediction data.
- Threshold (@visx/threshold package) - the component that displays the colored area between the two lines to highlight their difference. PredictPal employs the Threshold to emphasize the difference between the actual data and the test prediction line.
- Brush (@visx/brush package) - the component that allows the selection of a data subset. PredictPal employs it for two reasons: under the main chart content to allow zooming in on specific periods and see the line pattern more clearly, and on the main chart content to select a period of interest for the analysis.
- Legend and LegendItem (@visx/legend package) - the components that allow the creation of a chart legend with the description of the line colors.
- Tooltip (@visx/tooltip package) - the component that renders a tooltip at a passed position. In PredictPal it is used to display the information by hovering at a particular point on the chart. It also displays the two tooltips on both axes with the value corresponding to the currently hovered point.

The PredictPal line chart integrates these primitives with raw SVG elements to provide multiple features (Figure 4.9). These features allow users to perform three main tasks: data viewing, interactive data exploration, and data selection for further analysis.

Data viewing

The data is rendered on a line chart using LinePath components. Up to three lines are displayed: a line with actual data and two lines with test and

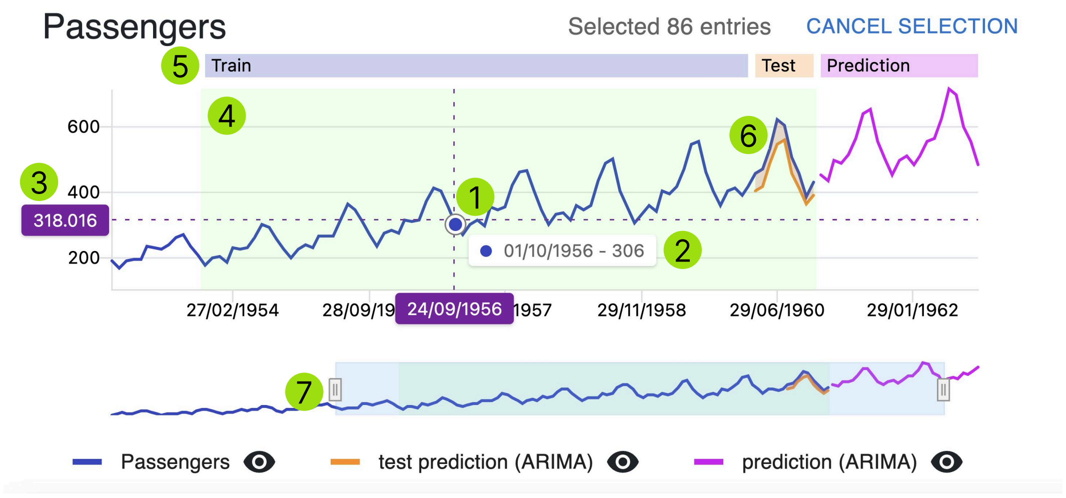


Figure 4.9.: PredictPal line chart and its elements: 1 - hovered data point, 2 - datapoint tooltip, 3 - axis tooltip with the crosshair, 4 - selected data for the analysis, 5 - data regions, 6 - colored area between the real data and test prediction emphasizing the magnitude of error, 7 - brush for zooming in a specific period. The dataset used for this example is "Air Passengers" [45], with permission of usage granted under Database Contents License (DbCL) v1.0.

real prediction. The data mapping to the colors can be found in the legend underneath the chart.

The data regions are displayed above the chart to mark what parts of the actual data were used for training and testing, and where the prediction is. These are up to three rectangles with diverging colors and the labels "train," "test," or "prediction." The data regions might differ from prediction to prediction since selecting smaller subsets of data to run prediction on is possible (it is described in depth in the "Data selection" subsection).

In case prediction lines are visible, the Threshold component highlights the area between the actual data and the test prediction. Such highlighting makes the magnitude of the test prediction error more visible and eases the visual assessment of the prediction quality.

PredictPal supports the analysis of both univariate and multivariate datasets.

For the latter, the app exploits the concept of small multiples to give users a visual overview of all the variables of interest. Small multiple design helps to compare the differences among the objects. The information slices are displayed within an eye span so that a user can compare them at a glance and have uninterrupted visual reasoning [86].

In PredictPal, small multiples display the variables from the dataset in the form of compact line charts (Figure 4.10). The vertical positioning of the small multiples helps to quickly analyze the trends among the variables over the flow of time. The small multiples also display the forecasts, if any are available. For easier visual filtering, the forecasted variables have the label "predicted" at the top right corner of their small multiples.

Hao et al. [87] in their article "Importance-Driven Visualization Layouts for Large Time Series Data" state that the perception of the importance of an object in the visualization should be supported by two display properties: position and size. As the authors suggest, the objects on the left are perceived as more important than those on the right. In addition, larger objects are treated as more important than smaller ones. Taking this into account, the detailed line chart for the currently selected variable is made significantly bigger than the block of the small multiples providing the overview of the dataset variables. As the detailed line chart is considered the main view, it is placed on the left side of the screen, while the supportive small multiples component stands on the right side next to it.

Interactive data exploration

PredictPal employs interactivity to allow the users to explore the data points of interest. Suppose a user is interested in a particular period and would like to see the behavior of the data in this range more profoundly. In that case, they can zoom the data in the x-axis and scroll it using the Brush component. One can see this component underneath the chart. All lines displayed on the line chart also appear on the brush to ease the visual perception while zooming.

In addition to zooming, users can explore a particular point of interest by hovering over it. On hover, the chart displays such elements:

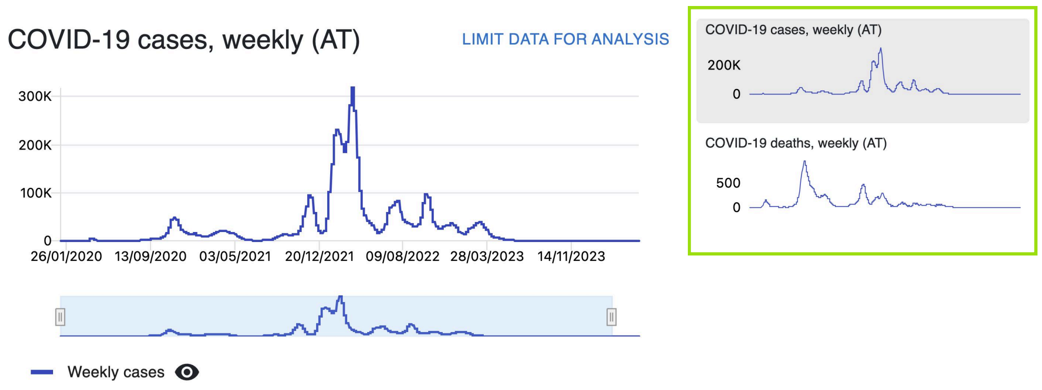


Figure 4.10.: Small multiples with the line charts (inside a green square) can be used to compare the trends in time between different variables. The example dataset "Covid 19 cases and deaths in Austria" for this figure was merged from the "Weekly deaths" and "Weekly cases" datasets provided by the World Health Organization, with permission of usage granted under CC BY 4.0 license.

- a crosshair,
- a tooltip with the corresponding point on the x-axis,
- a tooltip with the corresponding point on the y-axis,
- a line's data point, which is the closest to the hovered point.

On hover on the line's data point that is currently visible, a tooltip with its data appears.

The implementation of the hover interaction is a complex task involving multiple aspects. First, to show the crosshairs with the axis tooltips and display the closest data point, the coordinates of the currently hovered point need to be detected. To do this, we "cover" the chart contents with a custom ChartOverlays component. It consists of a Group component with a Bar component and crosshair lines inside.

The Bar component is transparent; it is stretched to a width and height of the chart's hoverable content and has a property *pointerEvents="all"*. Its purpose is to catch the hover interaction, namely mouse move and mouse leave events, in any part of the chart. These events then bubble up to Bar's parent components. Bubbling means that events that occur on an element first trigger the handlers on it, then on its parent, and then on the other ancestors all the way up [88]. The Group component then captures these events and triggers the custom handlers. The handler of the mouse over event finds the hovered coordinates using the local point function (@visx/event package). The found coordinates are then used to plot the vertical and horizontal crosshairs. The simplified code for this logic is presented in Appendix A.

Besides the crosshairs, the SVG circle components appear on hover to depict the closest data points to the hovered coordinates. At the same time, the handlers to display the data tooltips (@visx/tooltips package) are called. The tooltip positions are calculated using JavaScript event's *clientX* and *clientY* properties and SVG container boundaries. Tooltips are not SVG elements, so they must be placed outside of SVG. If tooltips are simply put next to an SVG in a common parent component, various problems might happen. First, their z-index might conflict with the z-index of another component and cause this component to overlap them. Second, one of their parent components in the tree might have overflow styles, which will cause the clipping of the tooltip to the boundaries of this parent component (for example, CSS style *overflow: hidden;* on a parent component).

To avoid these problems, we implemented the tooltips using the *useTooltip-InPortal* hook from @visx/tooltips [89]. This hook allows for rendering the tooltips inside the React Portal. Portals enable the rendering of some elements into a separate part of the DOM. They are often used for creating modals or tooltips since they allow for efficiently placing these elements above and outside other page content [90].

Data selection

In addition to viewing and exploring data, a user can select the data on the chart. The selected subset will be used instead of the complete source for analysis and prediction. To select a data span, one must click the button "Limit data for analysis" at the top of the line chart. Then, a user can select the area of interest by dragging its boundaries on the chart.

In the source code, this functionality is implemented by putting another Brush element on top of the main chart content. Namely, it is put inside the above-mentioned ChartOverlays component on top of the transparent Bar with crosshairs.

4.8.2. Exploratory data analysis

Exploratory data analysis (EDA) is an integral part of any prediction process. It helps users better understand the nature of their data and select the optimal prediction model parameters. In PredictPal, a user can perform it in the section "Get to Know Your Data," which follows the time series visualization block (Figure 4.11). The goal of PredictPal is to ease the analysis for users with beginner knowledge and lead them through the process. Therefore, the section is structured as a set of steps:

1. Provide seasonality information.
2. Check data consistency over time (stationarity).
3. Check data for randomness (white noise).
4. Check causal relationships between the variables.

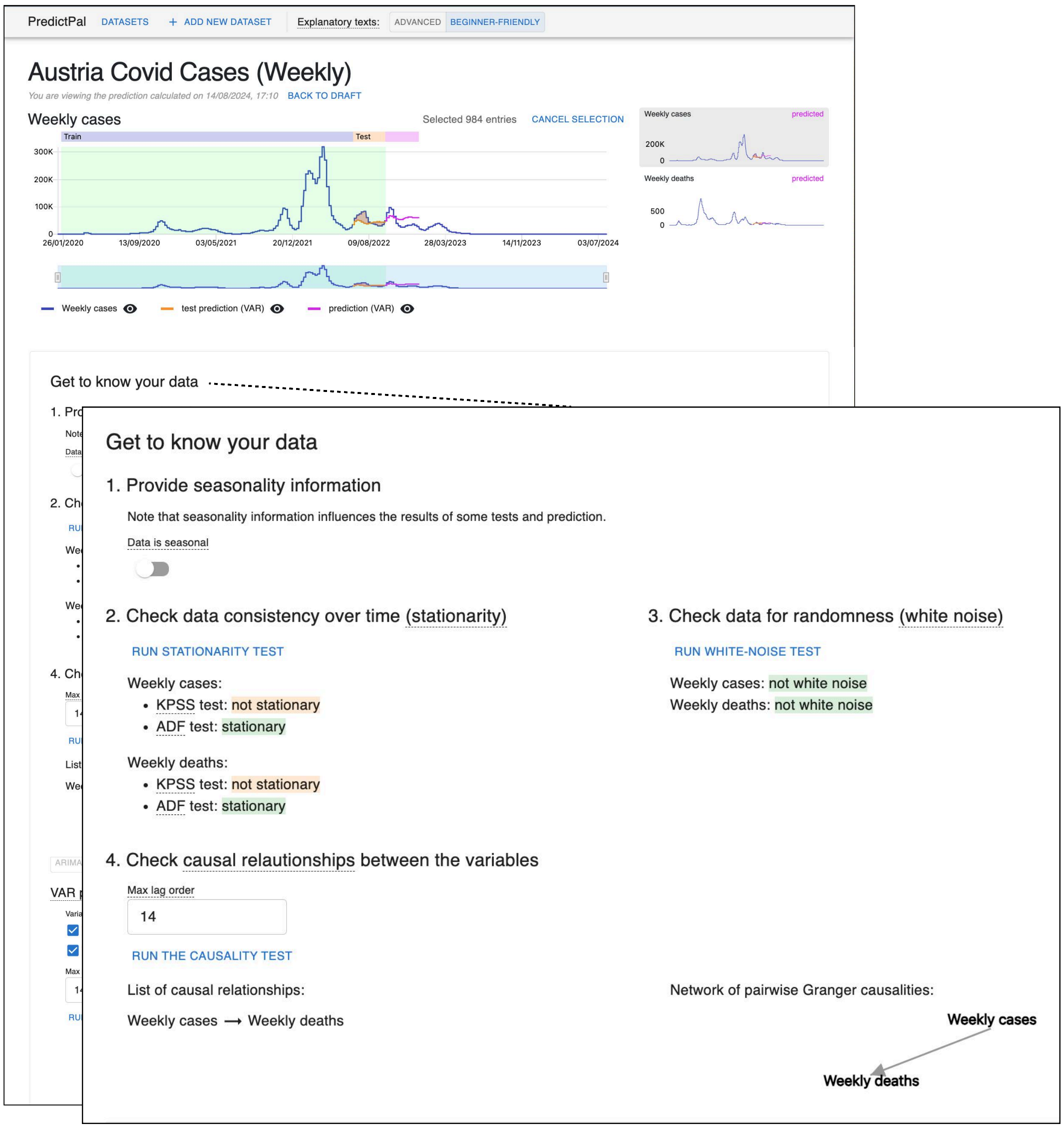
Step 4 is only available if the analyzed dataset is multivariate.

Step 1. Provide seasonality information

Step 1, "Provide seasonality information," is the starting point of the analysis. The reason is that some statistical tests, as well as the prediction model parameters, change in case the data is seasonal. If the time series exposes a seasonality pattern, a user has to toggle the "Data is seasonal" switch and provide the number of periods in a season.

Step 2. Check data consistency over time (stationarity)

Step 2 allows users to check data stationarity. To get the information, the user has to press the "Run stationarity test" button. Afterward, they will be able to see the stationarity results for each analyzed variable. This section is purely informative to help users understand the nature of the data better. If the data is not stationary, no manual action is needed. The modeling logic will convert the data to stationary under the hood.



Get to know your data

1. Provide seasonality information

Note that seasonality information influences the results of some tests and prediction.

Data is seasonal

2. Check data consistency over time (stationarity)

RUN STATIONARITY TEST

Weekly cases:
- KPSS test: not stationary
- ADF test: stationary

Weekly deaths:
- KPSS test: not stationary
- ADF test: stationary

3. Check data for randomness (white noise)

RUN WHITE-NOISE TEST

Weekly cases: not white noise
Weekly deaths: not white noise

4. Check causal relautionships between the variables

Max lag order
14

RUN THE CAUSALITY TEST

List of causal relationships:

Weekly cases ⟶ Weekly deaths

Network of pairwise Granger causalities:

Figure 4.11.: Section "Get to know your data" for performing preliminary data analysis

Step 3. Check data for randomness (white noise)

Step 3 offers an opportunity to test a special case of data stationarity - white noise. White noise is a random process that immediately forgets its past and, thus, cannot be predicted. It is important to know this information about every data field to decide whether it makes sense to proceed with a prediction.

To retrieve the information, a user must press the "Run white noise test" button. The results are displayed as a list of fields and their test outcomes ("white noise" / "not white noise"). For clarity, the results are highlighted with two different colors: yellow (warning) for "white noise", indicating that the data prediction will not by meaningful, and green (success) for "not white noise".

Step 4. Check causal relationships between the variables

As a last step, if the dataset is multivariate, PredictPal suggests checking the causal relationships between the variables. This information will give users an understanding of whether one variable can help predict the other variable. It will also show what variables do not have any influence on each other's patterns. Such insights will be of significant help when deciding whether it makes sense to perform a prediction using the VAR model and what variables to include in the modeling process.

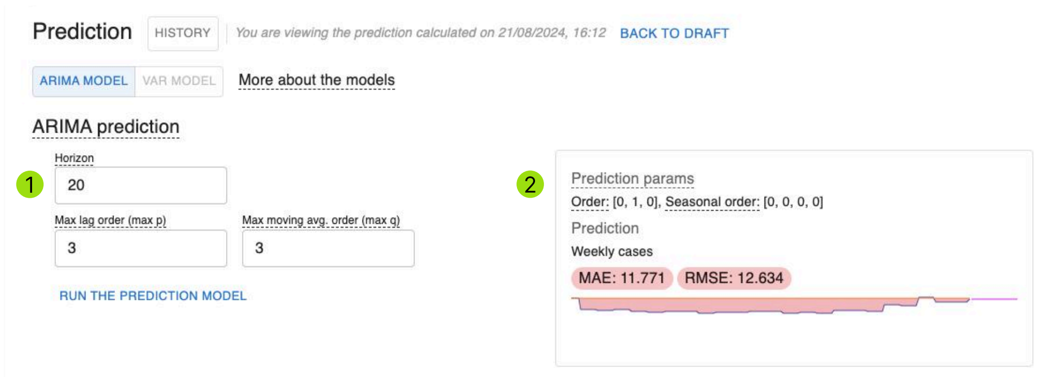
4.8.3. Prediction

After the "Get to know your data" section, the user reaches the most important part of the system - the "Prediction" section. Based on the nature of the data and the insights from the analysis, a user can choose between two types of predictive models, ARIMA or VAR, using a toggle. VAR is only available for selection if the data are multivariate.

ARIMA

PredictPal offers three inputs for configuring the ARIMA model (Figure 4.12). The horizon defines the number of points the user would like to predict. Max lag order (max p) helps to avoid overly complex models, where considering too many past data values leads to overfitting. Max moving average order (max q) helps to ensure that the algorithm disregards the models that incorporate too many past error terms.

In addition, the algorithm takes into account the data from the "Provide seasonality information" block. Namely, if the data is seasonal, it passes this information together with the number of periods in season to the model.
After the request is executed, the selected model parameters and prediction errors are shown on the evaluation card in the "Prediction" section. The prediction errors are displayed in the form of color-coded chips. The higher the value of an error among the other error values of this type, the more saturated red is used as a background. In addition, the sparkline chart is shown under the prediction errors. It helps to understand the shape of the prediction line in comparison to the actual data. The more comprehensive visual representation of prediction is shown on the main line chart at the top of the page.



You are viewing the prediction calculated on 21/08/2024, 16:12

ARIMA prediction

Horizon: 20

Max lag order (max p): 3

Max moving avg. order (max q): 3

Prediction params

Order: [0, 1, 0], Seasonal order: [0, 0, 0, 0]

Prediction

Weekly cases

MAE: 11.771 RMSE: 12.634

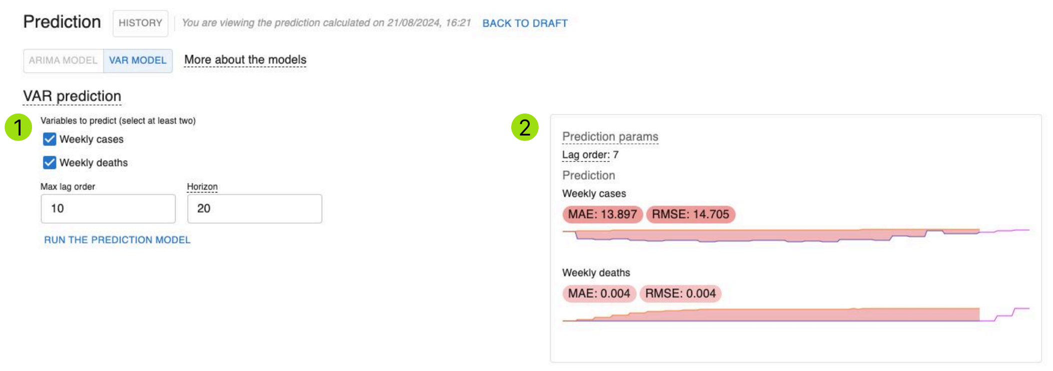
Figure 4.12.: Interface for prediction with ARIMA model. ARIMA prediction form (1) allows to configure a horizon, maximum lag order and maximum moving average order of a model. The evaluation card (2) displays the selected parameters, the errors for the variable prediction and the sparkline chart for comparison of the actual and the predicted data.

VAR

Before using the VAR model, it is essential to understand what variables to employ in the prediction. A user can get these insights from the block "4. Check causal relationships between the variables", which was described above.

In the "Prediction" block, one can select the variables of interest using the checkbox group (Figure 4.13). At least two variables have to be selected. A user should also specify the maximum lag order and define the prediction horizon.

Similarly, as in the ARIMA block, the best model parameters and results are presented on the evaluation card. The prediction error chips and the sparklines depicting the predicted lines in comparison to actual data are displayed for each variable separately. The complete visual representation of predictions is shown in the main line chart at the top of the page.


Prediction HISTORY | You are viewing the prediction calculated on 21/08/2024, 16:21 BACK TO DRAFT

ARIMA MODEL VAR MODEL More about the models

VAR prediction

1 Variables to predict (select at least two)

✓ Weekly cases

✓ Weekly deaths

Max lag order: 10

Horizon: 20

RUN THE PREDICTION MODEL

2 Prediction params

Lag order: 7

Prediction

Weekly cases

MAE: 13.897 RMSE: 14.705

Weekly deaths

MAE: 0.004 RMSE: 0.004

Figure 4.13.: Interface for prediction with VAR model. VAR prediction form (1) allows to select the variables for prediction, and configure maximum lag order and the horizon. The evaluation card (2) displays the selected parameters, the list of predicted variables with their prediction errors and the sparkline charts for comparing the actual and the predicted data.

4.8.4. Prediction History

One of the initial system requirements was to provide the capabilities for easy comparison of different models and their parameters. A related requirement was to supply the opportunity of reviewing the results of previous predictions and the parameters.

To cover both of these objectives, PredictPal offers the "History" section. Prediction history is placed inside a sidebar, which shows up on demand at the right side of the screen. To open the sidebar, a user has to press the button "History," which is located inside the "Prediction" section of the UI.

The "History" section presents a list of evaluation cards similar to those displayed in the "Prediction" section after the forecast is made. Each card displays the model type (ARIMA or VAR) and the time when the forecast was made. It also shows prediction parameters, a list of predicted variables with the error metrics, and the sparkline charts depicting the forecasts in comparison to the real data.

By clicking on a card, a user can select the prediction of interest. Its results are then shown on the main line chart. The label "You are viewing the prediction calculated on <time>" appears above the line plot. The sections "Get to Know Your Data" and "Prediction" display the historical values of the form inputs, as well as the results of statistical tests and the prediction. The button "Back to draft" on the right side of the label can be used to deselect the history item and start predicting anew.

To facilitate efficient navigation of the history list and enable users to quickly identify the best predictions, the "History" section offers a sorting functionality. By default, the items are sorted by date in descending order. A sorting popup opens when one clicks on the sorting button on top of the list. It offers descending and ascending sorting by each variable's prediction errors (MAE or MRSE) or by date.

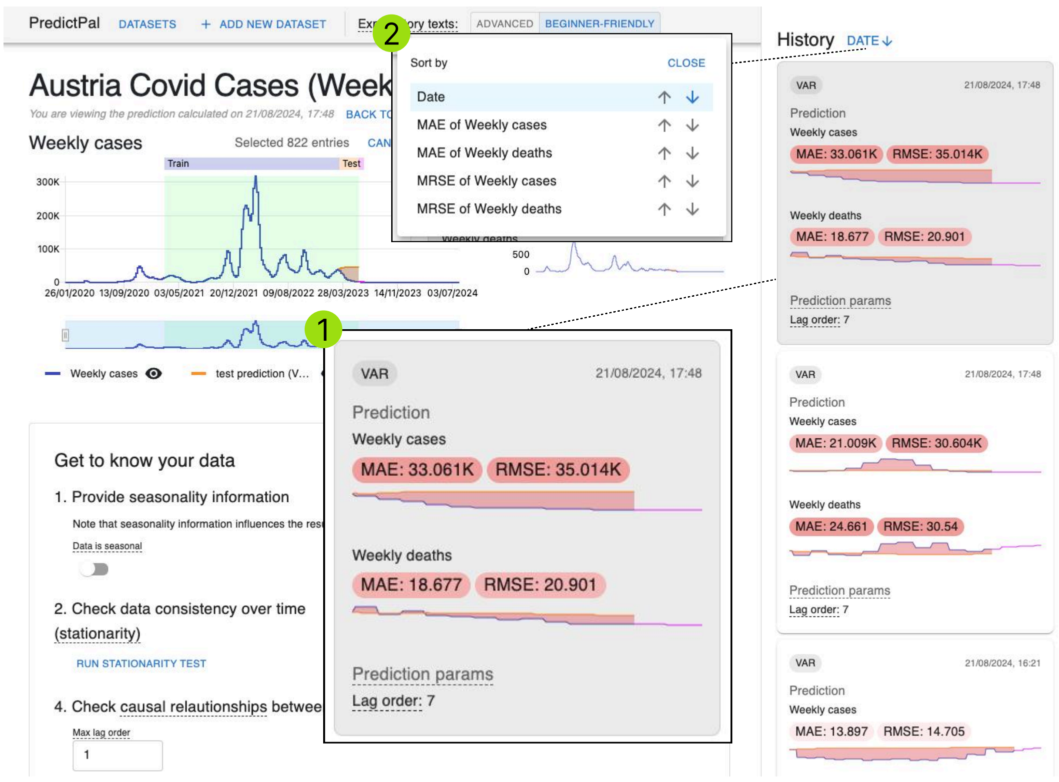


Figure 4.14.: PredictPal history. A prediction history card (1) displays the main information about the prediction, such as the selected model, the time of prediction, the prediction parameters and the predicted variables along with their error metrics. The sorting popup (2) allows to sort the prediction list by different kinds of errors and by date.

5. Evaluation

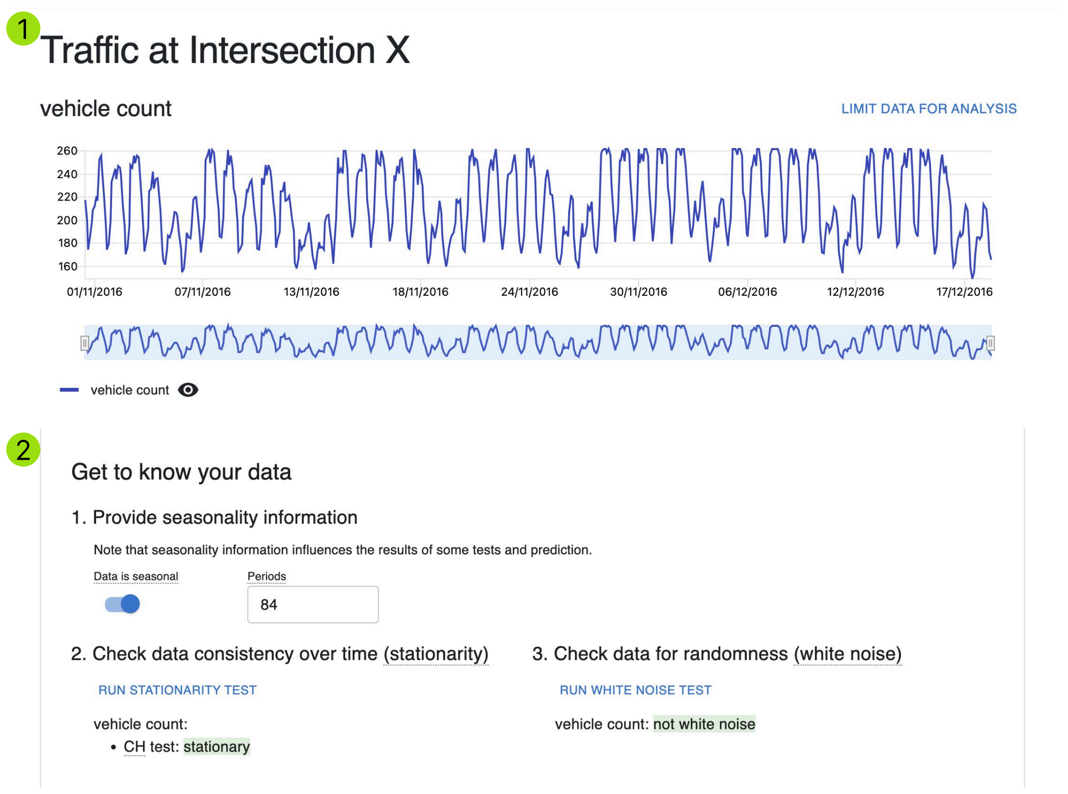
The evaluation of this project is conducted through a demonstration involving three fictional users performing prediction tasks across datasets from various fields. The first two illustrated scenarios show that the system is helpful for the prediction of the datasets from the different spheres. The third example demonstrates the use case when the system cannot provide a high-quality prediction. It also becomes clear that some user interface and algorithmic improvements could further optimize the system's performance (see Chapter 6).

5.1. Vehicle traffic prediction

John Doe, a municipal office worker responsible for traffic management, needs to plan upcoming roadworks at intersection X. In order to schedule the roadworks during times of lower traffic, John would like to obtain a forecast of the traffic flow for the next week. Using PredictPal, he inputs a historical univariate traffic dataset containing the bi-hourly amounts of cars going through the crossing.

Upon examining the line chart in the analysis view (Figure 5.1), John notices a consistent daily pattern: the flow increases and stays high during the day, and decreases at night. Therefore, in the "Provide seasonality information" step, the user marks the data as seasonal and provides 12 (the daily number of data records) as the number of periods in a season. Further, the user checks the data stationarity and randomness and proceeds to data prediction using the ARIMA model. The user assumes that reasonable parameters for the model would be a maximum lag order of 12 records and a maximum moving average of 6 records.

After obtaining the forecast, John reviews the evaluation card with the



Get to know your data

1. Provide seasonality information

Note that seasonality information influences the results of some tests and prediction.

Data is seasonal: ●

Periods: 84

2. Check data consistency over time (stationarity)

RUN STATIONARITY TEST

vehicle count:
- CH test: stationary

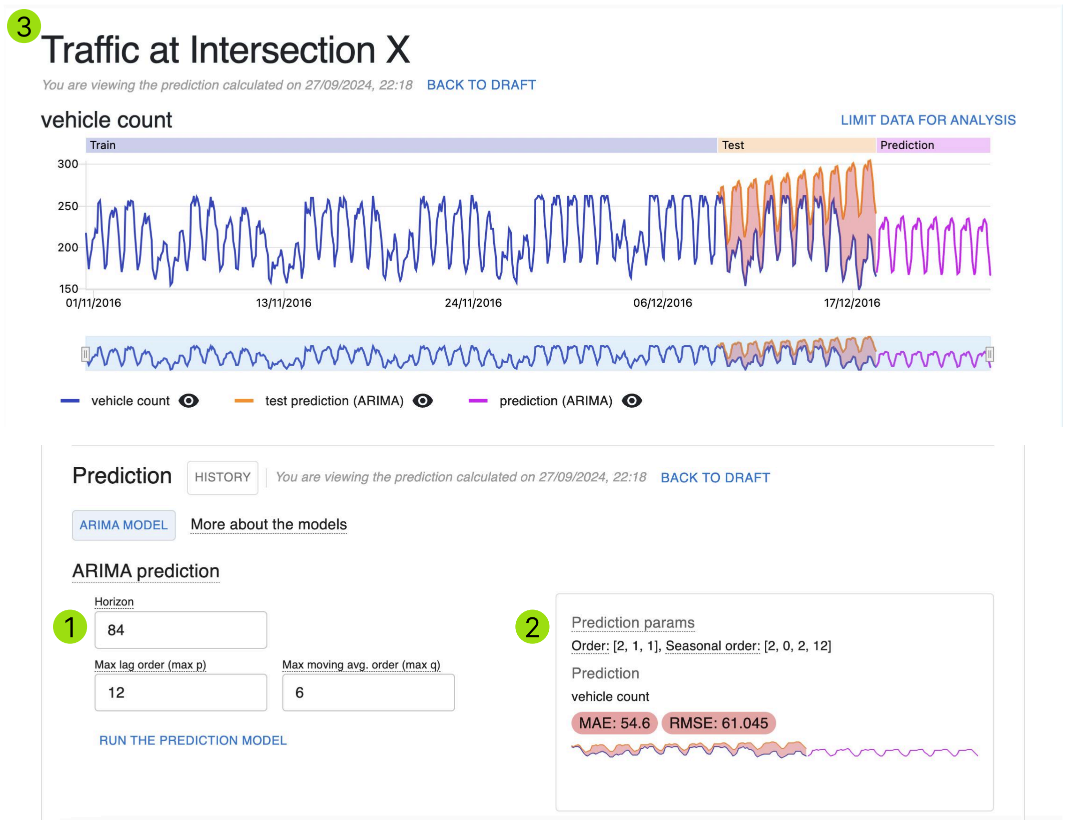
3. Check data for randomness (white noise)

RUN WHITE NOISE TEST

vehicle count: not white noise

Figure 5.1.: On the line chart (1), the user investigates the traffic at intersection data and its pattern. One can notice that every day the traffic goes up in the morning and goes down in the evening, with the minimum values at night. In "Get to know your data" section (2), the user marks the data as seasonal with 12 periods in season (the number of records in one day) and performs the preliminary analysis.

detailed information (Figure 5.2). Upon examining the error chips in the evaluation card, he notices that the prediction errors seem quite large, considering the dataset's magnitude. John then navigates to the main chart view, where he observes an unexpected upward trend in the line for test prediction. He also spots a significant red threshold indicating the error between the actual data and the test prediction. Moreover, the out-of-sample prediction captures some pattern, but it differs from that observed in the real data.



Traffic at Intersection X

You are viewing the prediction calculated on 27/09/2024, 22:18 BACK TO DRAFT

Prediction

You are viewing the prediction calculated on 27/09/2024, 22:18 BACK TO DRAFT

ARIMA prediction

Horizon: 84

Max lag order (max p): 12

Max moving avg. order (max q): 6

Prediction params

Order: [2, 1, 1], Seasonal order: [2, 0, 2, 12]

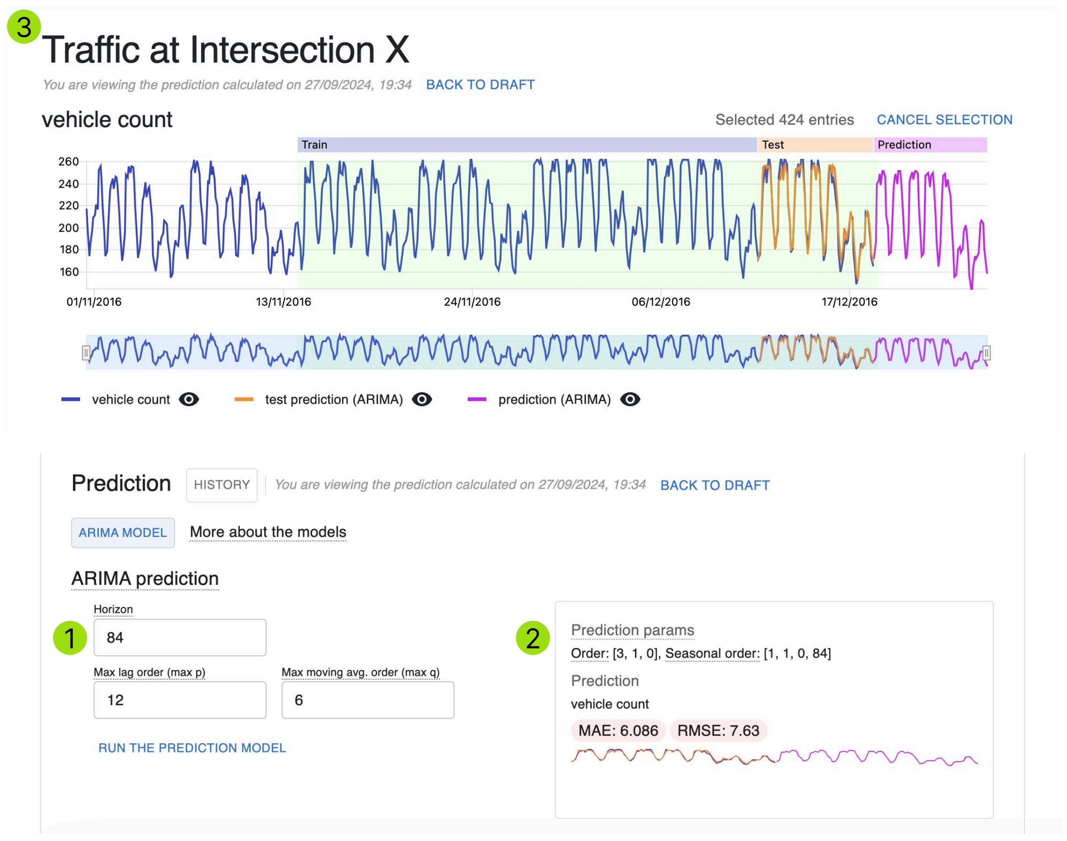
Prediction

vehicle count

MAE: 54.6 RMSE: 61.045

Figure 5.2.: The user performs the prediction for the traffic using ARIMA model with daily seasonality 12 (the number of records in a day), maximum lag order 12 (the number of records in a day) and maximum moving average order 6 (1). In the evaluation card (2), the user examines the selected parameters (order and seasonal order); the user also observes that the MAE and RMSE error chips report quite large values and have a saturated red background. On the line chart (3), the user can observe an unexpected upward trend of the test prediction. In addition, the pattern of the out-of-sample prediction does not fully resemble the real data pattern, where the last two days of a week report smaller values in comparison to the first five.

John re-examines the behavior of the actual data and notices that while a similar pattern repeats daily, the vehicle flow on the weekend is smaller compared to the weekdays. He assumes that accounting for weekly seasonality might produce a more accurate prediction. Therefore, he changes the number of periods to 84 (the number of records in a week). Additionally,



Horizon: 84

Max lag order (max p): 12

Max moving avg. order (max q): 6

Order: [3, 1, 0], Seasonal order: [1, 1, 0, 84]

MAE: 6.086 RMSE: 7.63

Figure 5.3.: The user performs the prediction for the traffic using ARIMA model with weekly seasonality 84 (the number of records in a week), maximum lag order 12 (the number of records in a day) and maximum moving average order 6 (1). In the evaluation card (2), the user examines the selected parameters (order and seasonal order); the user also observes that the MAE and RMSE error chips have a lighter red background in comparison to the previous forecast, and are reporting smaller values. On the line chart (3), the user can see that the area between actual and predicted data lines is considerably smaller in comparison to the previous forecast.

John selects a smaller subset of data including five weeks. Afterward, the user checks the newly generated prediction (Figure 5.3) and observes that both numeric prediction errors and the red threshold area on the line chart became less profound.

To verify that the latest prediction is of better quality than the previous ones, John opens the "History" section and sorts the cards by prediction error. The user employs the most recent traffic prediction result (Figure 5.3) as the best one to schedule the road works in the times with the smallest vehicle load.

5.2. Insects population forecast

Jane Doe is an epidemiologist tasked with monitoring insect populations in her city. Her responsibilities include forecasting mosquito numbers based on weather conditions to develop effective control strategies. Using PredictPal, she inputs a historical multivariate dataset with daily data on mosquito indicators, temperature, and rainfall.

By examining the line chart and the small multiples in the analysis view (Figure 5.4), Jane observes that the temperatures and the mosquito indicator numbers are consistently high at the middle of the year but decrease at the beginning and the end of a year. While the similarities in the amount of rain are not as pronounced, a slight yearly seasonal pattern is still visible. Therefore, in the analysis "Provide seasonality information" step, Jane marks the data as seasonal. She enters 365 (days) as the number of periods in the season. Further, the user notices that the first season reports lower mosquito indicator values than the subsequent seasons. Therefore, Jane opts to exclude it from the training data by selecting the training subset on the line chart. Afterward, she checks the stationarity and verifies that the data is not white noise.

Since the dataset is multivariate, the "Get to know your data" section includes a step that allows checking for the causal relationships between the variables. It takes 7–10 days for a mosquito to grow from an egg to an adult. Based on this, Jane assumes that it takes approximately 10 days for favorable weather conditions, such as large amounts of rain or high temperatures, to be reflected in the growing number of mosquitoes. She selects 10 (days) as the maximum lag order she would like to test for the causal relations between the variables. The results of the causality test indicate that there is a unidirectional causal connection from the temperature to the mosquito indicator and from the rain to the mosquito indicator.

Based on this information, the user selects the VAR model to proceed with the prediction of a few variables simultaneously. Initially, she focuses on forecasting the mosquito indicator together with the temperature (Figure 5.5) since both variables exhibit clear seasonal patterns. Subsequently Jane conducts another prediction employing the rain variable (Figure 5.6) to check whether she can receive a better prediction than the previous one. Last, but not least, Jane decides to check the prediction employing all available variables (Figure 5.7).

To determine the best prediction, she opens the history section and sorts the forecasts by RMSE of the mosquito indicator. The sorting indicates that the first prediction using mosquito indicator and the mean temperature (Figure 5.5) had the lowest prediction error. Jane Doe proceeds with developing the plan for controlling the mosquito population at different stages of its growth using on the generated prediction.

5.3. Cryptocurrency price prediction

Alice Smith invests money into cryptocurrencies to have an additional source of income. She is interested in Binance Coin (BNB), which has recently gained popularity. Alice wants to see a forecast of the BNB price growth for the next quarter to understand whether investing in this cryptocurrency is a good idea. In addition, she would like to know the best time for such an investment. Using PredictPal, she inputs a historical univariate BNB price dataset with daily data granularity.

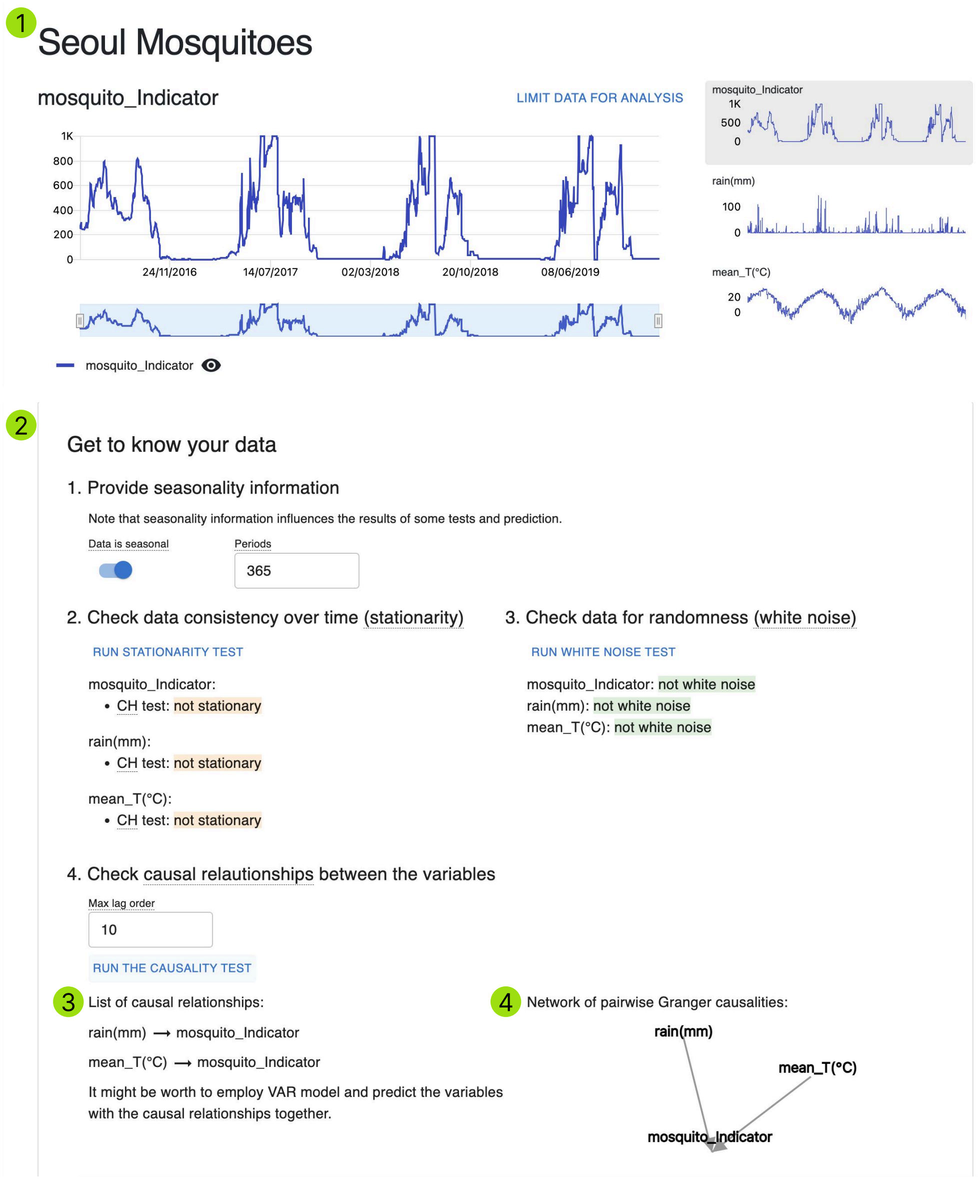
Upon examining the line chart in the analysis view (Figure5.8), Alice notices that the price was minimal for almost three years (2018 - 2021) and then started growing fast at the beginning of 2021. Alice does not see any seasonal similarities, so she does not mark the data as seasonal. Further, the user proceeds with the "Get to know your data" section. There she obtains the information that the data is not stationary and not white noise.

Afterward, the user proceeds with the prediction using the ARIMA model. She decides to test ARIMA with the following parameters: maximum p (lag order) of 20 days and maximum q (moving average order) of 10 days. After performing the prediction, Alice notices that the prediction error between the test prediction and the actual data is quite large (Figure 5.9). The test prediction line does not imitate any pattern and goes straight above. Even though the model's out-of-sample forecast looks better than the test prediction, Alice does not think she can trust it due to the large reported error. She tries to limit the data in the analysis, but the prediction does not improve.

Alice concludes that predicting cryptocurrency prices using an ARIMA model is difficult and likely will not provide high-quality results. There are several reasons for this. First of all, cryptocurrency price changes are a highly volatile process. Unpredictable price swings follow long periods of trending behavior, as observed in the BNB price data for 2021. In addition, ARIMA is a linear model. It assumes that the future price is a linear combination of past prices and prediction errors. However, the price of cryptocurrencies is often influenced by complex nonlinear combinations of multiple external factors such as market sentiment, geopolitical events, technological advancements, or regulatory changes. These factors are susceptible to unpredictable events that can drastically influence the process. ARIMA's linearity makes it incapable of capturing sharp price movements caused by sudden news or events.

Last but not least, ARIMA models tend to perform better when the data exhibits seasonal patterns. While some cryptocurrencies may display periodic trends, these trends are often inconsistent and not reliable enough.

To summarize, the ARIMA model might be too simple for such kind of data. One approach to achieving a better forecast is to explore more complex models, like LSTM or GARCH. These models can capture long-term dependencies and nonlinear patterns in time series data. This characteristic makes them more suitable for volatile, complex data like cryptocurrency prices.



Get to know your data

1. Provide seasonality information

Note that seasonality information influences the results of some tests and prediction.

Data is seasonal: on

Periods: 365

2. Check data consistency over time (stationarity)

RUN STATIONARITY TEST

mosquito_Indicator:
- CH test: not stationary

rain(mm):
- CH test: not stationary

mean_T(°C):
- CH test: not stationary

3. Check data for randomness (white noise)

RUN WHITE NOISE TEST

mosquito_Indicator: not white noise
rain(mm): not white noise
mean_T(°C): not white noise

4. Check causal relautionships between the variables

Max lag order: 10

RUN THE CAUSALITY TEST

List of causal relationships:

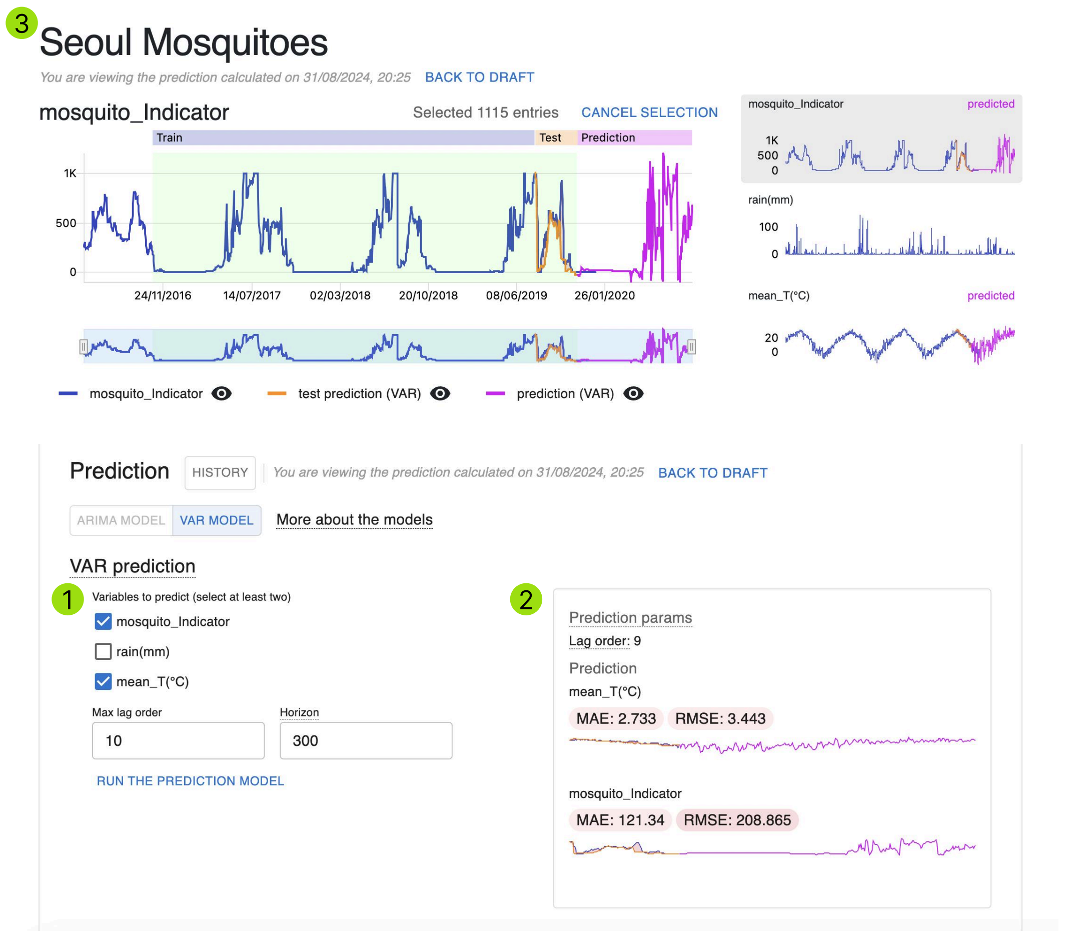
rain(mm) ⟶ mosquito_Indicator

mean_T(°C) ⟶ mosquito_Indicator

It might be worth to employ VAR model and predict the variables with the causal relationships together.

Network of pairwise Granger causalities:

Figure 5.4.: The user investigates the mosquito dataset (1) and conducts the preliminary analysis (2). To detect the causal relationships between variables of a dataset, the user performs the causality test with the lag order of 10. The identified causal relationships are displayed in two different views: the raw list of causal relationships (3) and the network of pairwise Granger causalities (4). The list view provides a precise summary of all detected pairwise causalities. The network graph view simplifies identifying the overall complex dependency structure within the dataset and detect the clusters of related variables. The dataset used for this example is "Mosquito Indicator in Seoul, Korea" [91], with permission of usage granted under CC0 1.0 Universal license.



Seoul Mosquitoes

You are viewing the prediction calculated on 31/08/2024, 20:25

mosquito_Indicator — Selected 1115 entries

Prediction

VAR prediction

Variables to predict (select at least two)

[✓] mosquito_Indicator

[ ] rain(mm)

[✓] mean_T(°C)

Max lag order: 10

Horizon: 300

Prediction params

Lag order: 9

Prediction

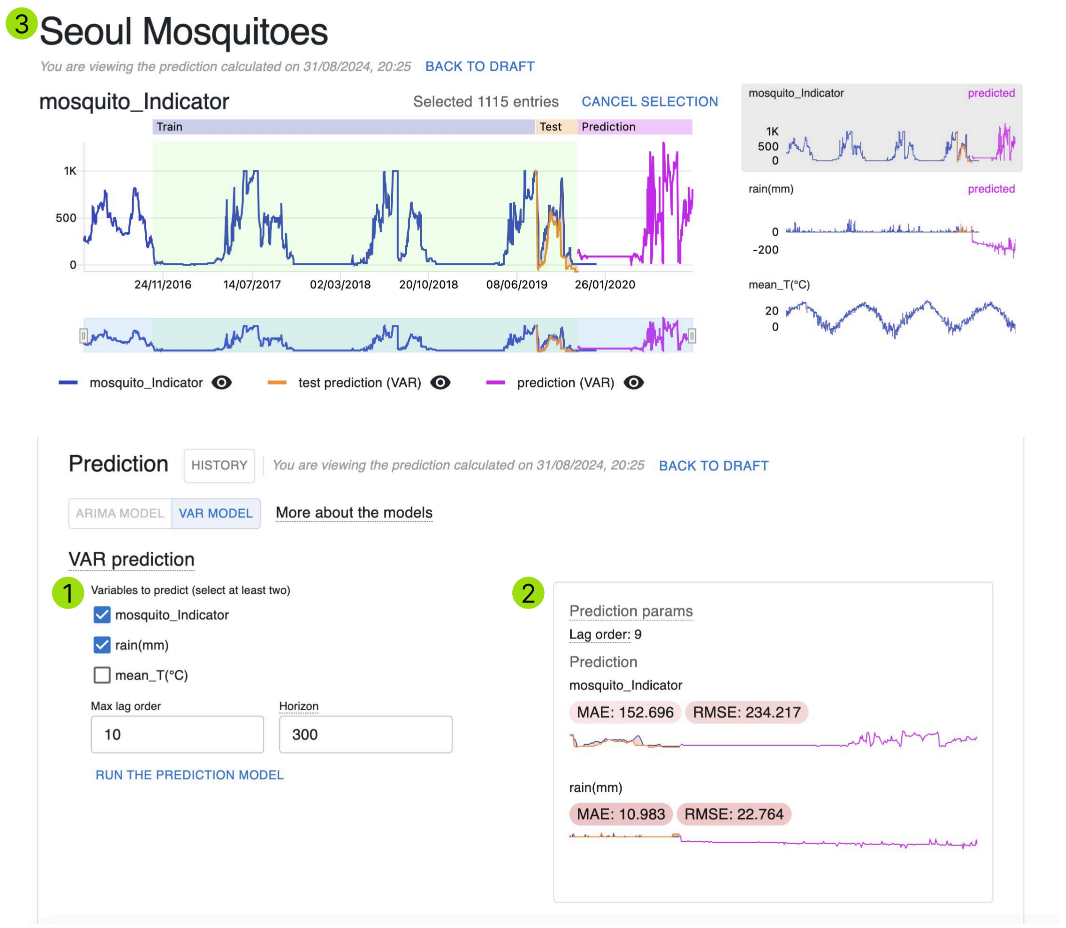
mean_T(°C)

MAE: 2.733 RMSE: 3.443

mosquito_Indicator

MAE: 121.34 RMSE: 208.865

Figure 5.5.: The user selects a subset of data and performs the prediction for mosquito indicator and temperature using VAR model (1). The prediction result is shown in the evaluation card together with the error chips (2), on the line chart (3) and on the small multiples with the label "predicted" (4). From the line chart, it is visible that the prediction repeats the actual data pattern quite well. The dataset used for this example is "Mosquito Indicator in Seoul, Korea" [91], with permission of usage granted under CC0 1.0 Universal license.



Seoul Mosquitoes

You are viewing the prediction calculated on 31/08/2024, 20:25 BACK TO DRAFT

mosquito_Indicator Selected 1115 entries CANCEL SELECTION

Prediction HISTORY You are viewing the prediction calculated on 31/08/2024, 20:25 BACK TO DRAFT

ARIMA MODEL VAR MODEL More about the models

VAR prediction

Variables to predict (select at least two)

- [✓] mosquito_Indicator
- [✓] rain(mm)
- [ ] mean_T(°C)

Max lag order: 10

Horizon: 300

RUN THE PREDICTION MODEL

Prediction params

Lag order: 9

Prediction

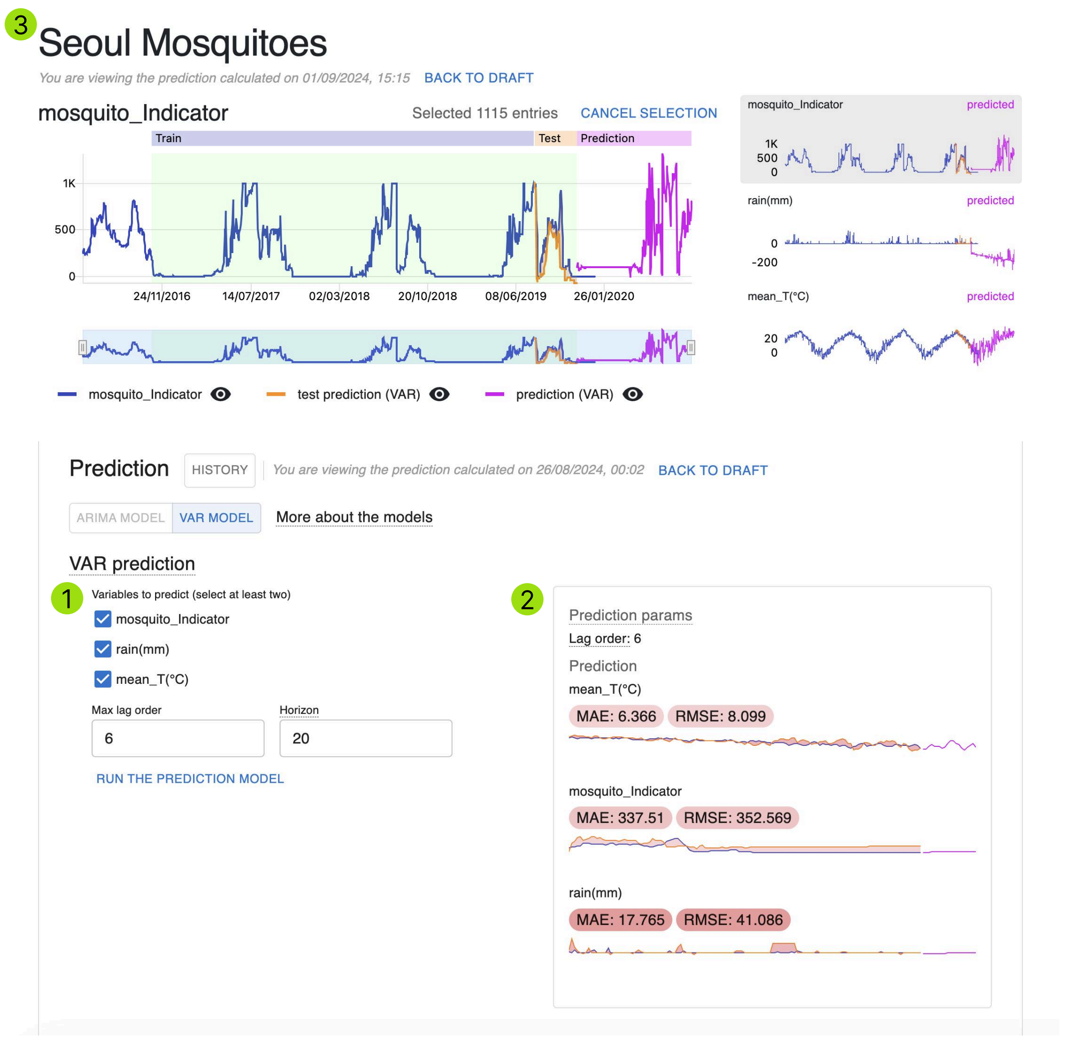
mosquito_Indicator

MAE: 152.696 RMSE: 234.217

rain(mm)

MAE: 10.983 RMSE: 22.764

Figure 5.6.: The user performs the prediction for mosquito indicator and rain using VAR model (1). The prediction result is shown in the evaluation card together with the error chips (2), on the line chart (3) and on the small multiples with the label "predicted" (4). From the line chart and the error chips, it is visible that the error between the test prediction and the actual data is slightly bigger in comparison to Figure 5.5, although the predicted line still resembles the repetitive pattern of the actual data. The dataset used for this example is "Mosquito Indicator in Seoul, Korea" [91], with permission of usage granted under CC0 1.0 Universal license.



You are viewing the prediction calculated on 01/09/2024, 15:15

Selected 1115 entries

You are viewing the prediction calculated on 26/08/2024, 00:02

Variables to predict (select at least two)

- ✓ mosquito_Indicator
- ✓ rain(mm)
- ✓ mean_T(°C)

Max lag order: 6

Horizon: 20

Prediction params

Lag order: 6

Prediction

mean_T(°C)

MAE: 6.366 RMSE: 8.099

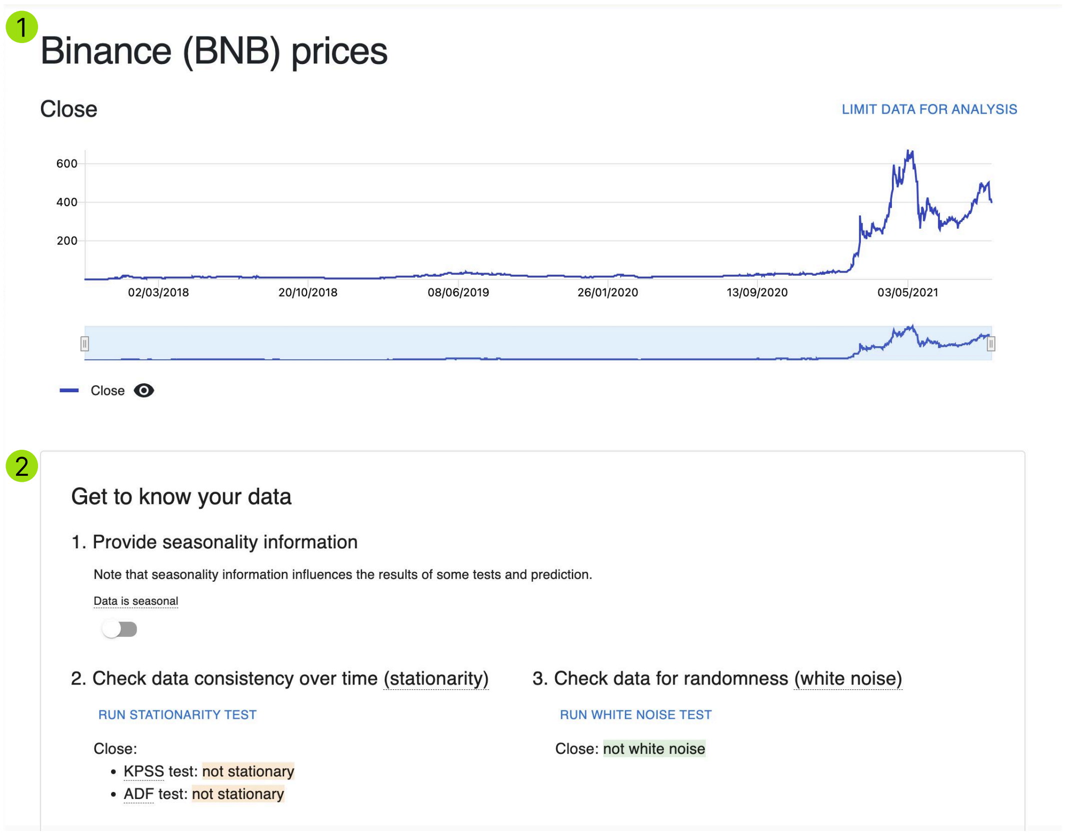
mosquito_Indicator

MAE: 337.51 RMSE: 352.569

rain(mm)

MAE: 17.765 RMSE: 41.086

Figure 5.7.: The user performs the prediction for mosquito indicator, temperature, and rain using VAR model (1). The prediction result is shown in the evaluation card together with the error chips (2), on the line chart (3) and on the small multiples with the label "predicted" (4). On the small multiples, it is visible that the rain (mm) prediction has a large error. The dataset used for this example is "Mosquito Indicator in Seoul, Korea" [91], with permission of usage granted under CC0 1.0 Universal license.



Get to know your data

1. Provide seasonality information

Note that seasonality information influences the results of some tests and prediction.

Data is seasonal

2. Check data consistency over time (stationarity)

RUN STATIONARITY TEST

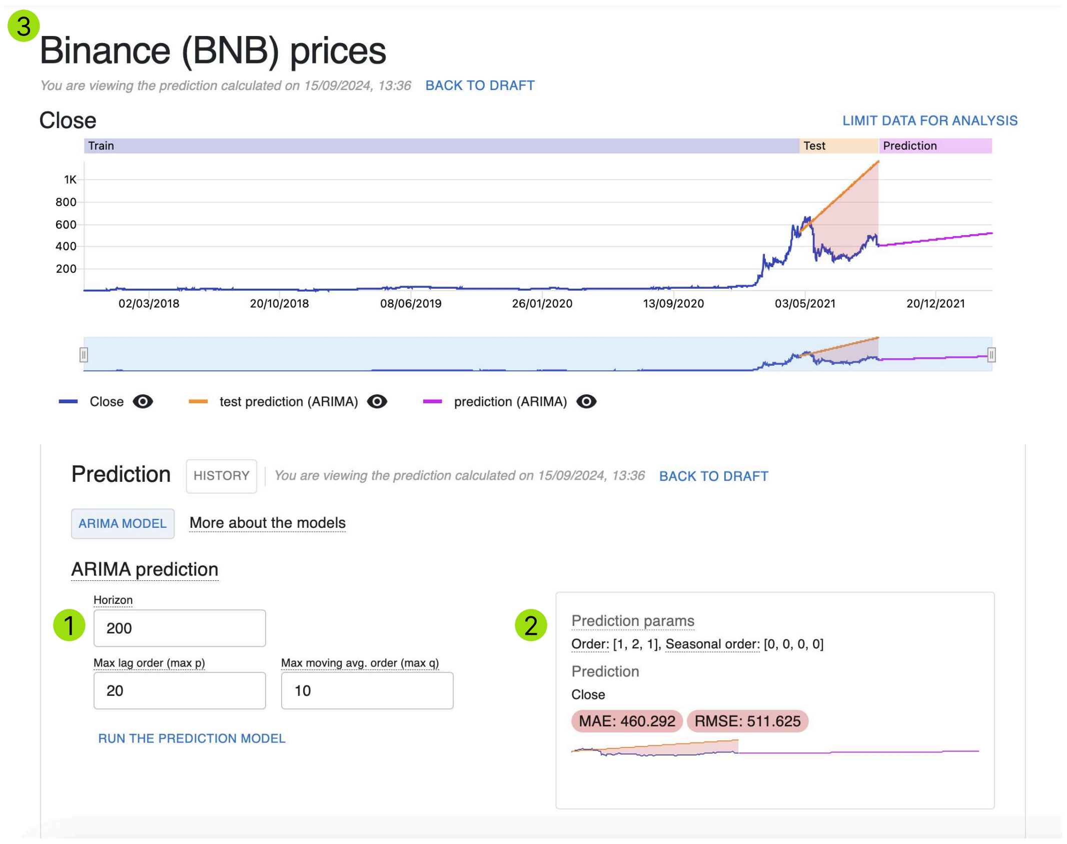
Close:
- KPSS test: not stationary
- ADF test: not stationary

3. Check data for randomness (white noise)

RUN WHITE NOISE TEST

Close: not white noise

Figure 5.8.: The user investigates the BNB price dataset (1) and performs its preliminary analysis (2) in "Get to know your data" section. The dataset used and adjusted for this example is "Cryptocurrency Historical Prices [Updated Daily]" [92], with permission of usage granted under CC0 1.0 Universal license.


3

Binance (BNB) prices

You are viewing the prediction calculated on 15/09/2024, 13:36 BACK TO DRAFT

Close

Prediction HISTORY *You are viewing the prediction calculated on 15/09/2024, 13:36* BACK TO DRAFT

ARIMA MODEL More about the models

ARIMA prediction

1

Horizon: 200

Max lag order (max p): 20

Max moving avg. order (max q): 10

RUN THE PREDICTION MODEL

2

Prediction params

Order: [1, 2, 1], Seasonal order: [0, 0, 0, 0]

Prediction

Close

MAE: 460.292 RMSE: 511.625

Figure 5.9.: The user investigates the BNB price dataset (1) and performs its preliminary analysis (2) in "Get to know your data" section. The dataset used and adjusted for this example is "Cryptocurrency Historical Prices [Updated Daily]" [92], with permission of usage granted under CC0 1.0 Universal license.

6. Limitations and Future Work

Like any minimum viable product, PredictPal has a set of limitations that need to be acknowledged. These limitations are related to various aspects, such as technological constraints, design choices, and predictive modeling challenges. Identifying these areas helps establish the foundation for future research and implementation efforts to enhance the currently available functionality.

6.1. Algorithmic aspects

Seasonality information. PredictPal system offers a set of steps for preliminary data analysis. While this analysis gives users some understanding of the nature of their data, incorporating additional aspects of analysis would be beneficial. As the first improvement, Step 1 ("Provide seasonality information") could include seasonal decomposition visualized in the form of three small line charts: trend, seasonal component, and residuals. The visualization will help users determine whether the data exhibits any trend or seasonality and select the proper number of seasonal periods based on that.

ACF and PACF charts. Integrating the autocorrelation (ACF) and partial autocorrelation (PACF) charts as a part of the analysis in PredictPal could be beneficial. These charts show the correlation of a time series with its past values at different lags and are often used in the analysis before the prediction. ACF displays the overall correlation, while PACF displays solely the correlation between two lagged terms by removing the influence of shorter lags. ACF and PACF help in the lag selection for multiple models. The residual autocorrelation can also be used for model diagnostics to check whether the model adequately captures the time series dynamics. ACF and PACF charts were not employed in the first version of the project because of the complexity of their interpretation. Understanding these charts requires advanced knowledge. They may be confusing for beginner-level users, which does not align with the project's goals and requirements. However, the ACF and PACF integration in PredictPal can be revisited in future versions. Ideally, one could find a straightforward metaphor to explain the data presented in these charts in an easy way.

Available models. PredictPal currently offers predictions using two models: ARIMA and VAR. While these models are suitable for performing simple forecasts and demonstrating the app's functionality, incorporating a set of better-performing models could bring great value to the users. As a part of future work, models such as LSTM, Prophet, or transformer models can be added to the app. A more extensive model selection will facilitate the discovery of better forecasts and help users understand the differences between various models. Furthermore, in such a way, the app will support the learning process.

Prediction evaluation methods. There are multiple ways to assess the prediction quality. At the moment, PredictPal calculates two error metrics: MAE and RMSE. In the future, more evaluation methods can be added. For instance, PredictPal could add a possibility to run a white noise test on the residuals of the prediction data. If the residuals form a white noise process, it indicates that a model is a good fit for the data.

6.2. Technical aspects

User authentication. The first version of PredictPal is a minimum viable product. It lacks the crucial features needed to transform it into a real product ready to be hosted in the cloud. An essential aspect is integrating the user model and the authentication mechanisms. It must include the ability to save a list of private analyzed datasets per user. Currently, all datasets added to the system are public by default.

Performance on very large datasets. One of the important technical limitations of PredictPal is the system's performance with large datasets. This issue is visible in the rendering of line charts. The app is considerably slower with very large data, sometimes causing the web page to freeze, especially when one tries to interact with the line chart. In addition to this, visualizing a huge and noisy time series creates visual clutter, making it difficult to discover the underlying patterns and trends. The future version of the system could employ some optimizations, such as data smoothing. Conventional methods for smoothing data include Gaussian, median, and other filters. In 2019, Rosen et al. [93] presented a method for smoothing line charts called TopoLines, which could also be employed.

Allowed data formats. Another limitation lies in the data formats that the current functionality supports. Currently, two data formats are allowed: CSV and JSON. In the next iteration, this list could be extended to allow XLSX and Pickle files.

6.3. Usability aspects

Available views. Currently, PredictPal offers only the analysis view to display results. Users can conduct the modeling process and discover the most accurate forecast within this view. In the next version, the app can be extended to include a presentation view. This view would solely contain the selected prediction for demonstration purposes, excluding the "Get to know your data" and "Prediction" sections.

Analysis results retrieval. Besides demoing the data, some users might want to retrieve it and use it in their work. Therefore, the functionality of downloading the prediction in the form of a PDF or an image, as well as in the form of CSV and JSON datasets, is of great importance. In addition, an option to publish their prediction with a publicly accessible URL can be added.

Visualization. Visualization components also have areas for improvement in future versions. First, PredictPal could incorporate additional chart types, such as bar charts, scatter plots, and others. Second, an option to change the prediction horizon by dragging the end of the existing prediction line on a chart would enhance the user experience. In such a way, the forecast on the chart could be updated immediately as the user drags, providing a smooth, user-friendly interaction. However, the speed of the prediction recalculation on the server might become a bottleneck for such implementation. Therefore, the feasibility of this feature has to be explored.

7. Conclusion

This thesis presented the design, implementation, and evaluation of PredictPal, the predictive visual analytics system for forecasting time series data. The developed application not only bridges the gap between complex statistical models and user-friendly interfaces but also integrates ways to make the prediction process more accessible for beginner users.

The project aimed to answer a set of research questions related to the predictive visual analytics process.

RQ1: *What are the key components required for building a system with the predictive visual analytics pipeline?*
Any predictive visual analytics system requires several components for smooth analysis and insights extraction. The functionality of PredictPal was built based on an understanding about the usual forecast process and research into the already available predictive visual analytics systems. The system consists of such main components:

- Data visualization: Visualizing data on the line chart on top of the screen helps users quickly understand the nature of the data and make initial assumptions about the future behavior of the observed process. The chart also displays predicted data, enabling users to visually assess the credibility of the forecast.
- Exploratory data analysis ("Get to know your data" section): The EDA component provides the ability to perform various statistical tests prior to forecasting. In such a way, it enables users to understand the nature of their data better.
- Prediction: The prediction component includes model selection and parameter configuration to perform the prediction.
- Model evaluation: The model evaluation card displays the model parameters along with the numerical prediction errors to help users

evaluate the forecast quality. It also shows the sparkline charts with the test and real prediction compared to the actual data in the same period so that the users can clearly see the pattern difference.

RQ2: *How to reduce the cognitive load of a user in assessing the prediction quality and the best model?*
Many approaches can be employed to reduce a user's cognitive load when assessing the prediction quality. PredictPal integrated such strategies:

- Providing aggregate error metrics: PredictPal provides the summary error statistics, namely, MAE and MRSE, to give users a quick understanding of model performance in numbers. The error values are color-coded using the different saturation of red to describe the magnitude of the errors compared to the previous predictions.
- Visualizing the errors on the chart: The line chart displays the actual and predicted values on the main chart in different colors, highlighting the difference, which represents the error, in red. This way, the magnitude of the error can be perceived easier.
- Employing interactivity in the visualization: Interactive features in the line charts allow the users to zoom in on the periods of interest and explore separate data points.
- Providing users with prediction history: The history of predictions allows users to navigate between different models easily, compare the forecasts, sort them by the error magnitudes, and select the best model parameters for their specific data.

RQ3: *How to develop a comprehensive user interface for time-series analysis and prediction, which is easy to exploit for the users with beginner knowledge?*

One major problem contributing to the steep learning curve of many predictive analytics systems for beginners is the assumption that all users possess expert knowledge by default. When starting to use such systems, beginners often feel overwhelmed, since they must not only familiarize themselves with the interface and features but also overcome the lack of knowledge about certain scientific concepts. This often requires them to spend time researching each new term or algorithm independently.

To ease the prediction process for users with beginner knowledge, PredictPal integrated such aspects:

- Clear and intuitive charts: The system employed simple and familiar line charts instead of integrating more complex and confusing visualization types.
- Step-by-step process: The analysis and prediction pipeline guides users through the process, eliminating the need to think about the next steps.
- Simple wording: Avoiding complex terminology in the step names makes it easier for a beginner to understand the essence of the process.
- Explaining the concepts: Explanatory texts available upon clicking a word provide the definitions of the terms and algorithms. Moreover, the system provides different explanations depending on the user's level of expertise. Definitions from the scientific papers are presented to expert users. Users with beginner knowledge are offered simplified alternatives with easy wording, no formulas, and multiple examples.

In summary, PredictPal successfully addresses the challenges of predictive visual analytics by making the forecasting process more accessible to users with different expertise levels. The system's straightforward design, explanatory texts and powerful analytical tools ensure that even novices in data science can effectively perform the prediction process. Further improvements could extend PredictPal's functionality, ensuring that the system continues developing as a valuable tool for users with different levels of knowledge in time series forecasting.

Bibliography

[1] W. Aigner, S. Miksch, H. Schumann, and C. Tominski, *Visualization of Time-Oriented Data*, Second Edition. Springer, 2023, ISBN: 978-1-4471-7527-8. DOI: 10.1007/978-1-4471-7527-8. [Online]. Available: https://timeviz.net (cit. on p. 1).

[2] S. Miksch and W. Aigner, "A matter of time: Applying a data–users–tasks design triangle to visual analytics of time-oriented data," *Computers & Graphics*, vol. 38, pp. 286–290, 2014 (cit. on p. 2).

[3] B. McCormick, T. DeFanti, and M. Brown, "Definition of visualization," *SIGGRAPH Comput. Graph*, vol. 21, no. 6, pp. 3–3, 1987 (cit. on p. 4).

[4] D. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, *Mastering the information age solving problems with visual analytics*. Eurographics Association, 2010 (cit. on p. 4).

[5] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. A. Keim, "Knowledge generation model for visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1604–1613, 2014 (cit. on p. 5).

[6] G. Andrienko, N. Andrienko, W. Chen, R. Maciejewski, and Y. Zhao, "Visual analytics of mobility and transportation: State of the art and further research directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 8, pp. 2232–2249, 2017 (cit. on p. 5).

[7] D. Koop, C. E. Scheidegger, S. P. Callahan, J. Freire, and C. T. Silva, "Viscomplete: Automating suggestions for visualization pipelines," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1691–1698, 2008 (cit. on p. 5).

[8] D. J. Lehmann, F. Kemmler, T. Zhyhalava, M. Kirschke, and H. Theisel, "Visualnostics: Visual guidance pictograms for analyzing projections of high-dimensional data," in *Computer Graphics Forum*, Wiley Online Library, vol. 34, 2015, pp. 291–300 (cit. on p. 5).

[9] F. Sperrle, D. Ceneda, and M. El-Assady, "Lotse: A practical framework for guidance in visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1124–1134, 2022 (cit. on p. 5).

[10] D. Ceneda, N. Andrienko, G. Andrienko, *et al.*, "Guide me in analysis: A framework for guidance designers," in *Computer Graphics Forum*, Wiley Online Library, vol. 39, 2020, pp. 269–288 (cit. on p. 5).

[11] D. Ceneda, T. Gschwandtner, T. May, *et al.*, "Characterizing guidance in visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 1, pp. 111–120, 2016 (cit. on p. 5).

[12] C. Collins, N. Andrienko, T. Schreck, *et al.*, "Guidance in the human-machine analytics process," *Visual Informatics*, vol. 2, no. 3, pp. 166–180, 2018 (cit. on p. 5).

[13] D. T. Larose, *Data mining and predictive analytics*. John Wiley & Sons, 2015 (cit. on p. 6).

[14] J. Krause, A. Perer, and E. Bertini, "Using visual analytics to interpret predictive machine learning models," *arXiv preprint arXiv:1606.05685*, 2016 (cit. on p. 6).

[15] Y. Lu, R. Garcia, B. Hansen, M. Gleicher, and R. Maciejewski, "The state-of-the-art in predictive visual analytics," in *Computer Graphics Forum*, Wiley Online Library, vol. 36, 2017, pp. 539–562 (cit. on p. 7).

[16] H. Yeon, H. Son, and Y. Jang, "Visual performance improvement analytics of predictive model for unbalanced panel data," *Journal of Visualization*, vol. 24, pp. 583–596, 2021 (cit. on p. 7).

[17] J. Krause, A. Perer, and K. Ng, "Interacting with predictions: Visual inspection of black-box machine learning models," 2016, pp. 5686–5697 (cit. on p. 7).

[18] S. Das, D. Cashman, R. Chang, and A. Endert, “Beames: Interactive multimodel steering, selection, and inspection for regression tasks,” *IEEE Computer Graphics and Applications*, vol. 39, no. 5, pp. 20–32, 2019 (cit. on p. 7).

[19] M. Bögl, W. Aigner, P. Filzmoser, *et al.*, “Visual analytics methods to guide diagnostics for time-series model predictions,” in *Proceedings of the 2014 IEEE VIS Workshop on Visualization for Predictive Analytics*, vol. 1, 2014 (cit. on p. 7).

[20] D. Cashman, S. R. Humayoun, F. Heimerl, *et al.*, “A user-based visual analytics workflow for exploratory model analysis,” in *Computer Graphics Forum*, Wiley Online Library, vol. 38, 2019, pp. 185–199 (cit. on p. 8).

[21] K. Xu, J. Yuan, Y. Wang, C. Silva, and E. Bertini, “Mtseer: Interactive visual exploration of models on multivariate time-series forecast,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 2021, pp. 1–15 (cit. on p. 9).

[22] J. Demšar and Z. Bosnić, “Detecting concept drift in data streams using model explanation,” *Expert Systems with Applications*, vol. 92, pp. 546–559, 2018 (cit. on p. 9).

[23] X. Wang, W. Chen, J. Xia, *et al.*, “Conceptexplorer: Visual analysis of concept drifts in multi-source time-series data,” in *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2020, pp. 1–11 (cit. on p. 9).

[24] H. Galmeanu and R. Andonie, “Concept drift visualization of svm with shifting window,” *arXiv preprint arXiv:2406.13754*, 2024 (cit. on p. 9).

[25] W. Yang, Z. Li, M. Liu, *et al.*, “Diagnosing concept drift with visual analytics,” in *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*, IEEE, 2020, pp. 12–23 (cit. on p. 9).

[26] C. Xu, T. Neuroth, T. Fujiwara, R. Liang, and K.-L. Ma, “A predictive visual analytics system for studying neurodegenerative disease based on dti fiber tracts,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 4, pp. 2020–2035, 2021 (cit. on p. 10).

[27] B. C. Kwon, M.-J. Choi, J. T. Kim, *et al.*, "Retainvis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 299–309, 2018 (cit. on p. 10).

[28] N. Y. Philip, M. Razaak, J. Chang, M. O'Kane, B. K. Pierscionek, *et al.*, "A data analytics suite for exploratory predictive, and visual analysis of type 2 diabetes," *IEEE Access*, vol. 10, pp. 13 460–13 471, 2022 (cit. on p. 10).

[29] T. Schreck, T. Tekušová, J. Kohlhammer, and D. Fellner, "Trajectory-based visual analysis of large financial time series data," *ACM SIGKDD Explorations Newsletter*, vol. 9, no. 2, pp. 30–37, 2007 (cit. on p. 10).

[30] C. Xie, W. Chen, X. Huang, Y. Hu, S. Barlowe, and J. Yang, "Vaet: A visual analytics approach for e-transactions time-series," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1743–1752, 2014 (cit. on p. 10).

[31] D. Jonker, R. Brath, and S. Langevin, "Industry-driven visual analytics for understanding financial timeseries models," IEEE, 2019, pp. 210–215 (cit. on p. 10).

[32] L. Chen, Y. Ouyang, H. Zhang, S. Hong, and Q. Li, "Riseer: Inspecting the status and dynamics of regional industrial structure via visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 1, pp. 1070–1080, 2022 (cit. on p. 11).

[33] D. Sun, Z. Feng, Y. Chen, *et al.*, "Dfseer: A visual analytics approach to facilitate model selection for demand forecasting," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13 (cit. on p. 11).

[34] L. Wang and L. Zhao, "Digital economy meets artificial intelligence: Forecasting economic conditions based on big data analytics," *Mobile Information Systems*, vol. 2022, no. 1, p. 7 014 874, 2022 (cit. on p. 11).

[35] I. Kalamaras, A. Zamichos, A. Salamanis, *et al.*, "An interactive visual analytics platform for smart intelligent transportation systems management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 487–496, 2017 (cit. on p. 11).

[36] C. Lee, Y. Kim, S. Jin, *et al.*, “A visual analytics system for exploring, monitoring, and forecasting road traffic congestion,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 11, pp. 3133–3146, 2019 (cit. on p. 11).

[37] Â. P. Alves, A. M. Milani, and I. H. Manssour, “Visual analytics system for energy data in smart cities and buildings,” in *2020 IEEE International Smart Cities Conference (ISC2)*, IEEE, 2020, pp. 1–8 (cit. on p. 11).

[38] Salesforce, Inc., *Tableau from Salesforce*. [Online]. Available: `https://www.tableau.com` (visited on 07/27/2024) (cit. on p. 11).

[39] Microsoft, *Power BI*. [Online]. Available: `https://www.microsoft.com/en-us/power-platform/products/power-bi` (visited on 07/27/2024) (cit. on p. 11).

[40] QlikTech International AB, *Qlik sense*. [Online]. Available: `https://www.qlik.com/us/products/qlik-sense` (visited on 08/06/2024) (cit. on p. 11).

[41] Oracle, *Analytics cloud*. [Online]. Available: `https://docs.oracle.com/en-us/iaas/analytics-cloud/index.html` (visited on 08/06/2024) (cit. on p. 11).

[42] K. Schlegel, A. Ganeshan, D. Pidsley, *et al.*, *Magic quadrant for analytics and business intelligence platforms*, 2024. [Online]. Available: `https://www.gartner.com/doc/reprints?id=1-2HWG6WTS&ct=240621&st=sb` (visited on 07/27/2024) (cit. on p. 12).

[43] I. Booth and H. Wang, *Tableau pulse: Proactive answers to your common business questions with automated insights*. [Online]. Available: `https://www.tableau.com/blog/tableau-pulse-automated-business-insights` (visited on 07/27/2024) (cit. on p. 12).

[44] Salesforce, Inc., *Technical specifications*. [Online]. Available: `https://www.tableau.com/products/techspecs` (visited on 07/27/2024) (cit. on p. 12).

[45] R. Nimer, *Air passengers*. [Online]. Available: `https://www.kaggle.com/datasets/rakannimer/air-passengers?resource=download` (visited on 07/27/2024) (cit. on pp. 13, 14, 16, 41).

[46] Salesforce, Inc., *How forecasting works in Tableau*. [Online]. Available: `https://help.tableau.com/current/pro/desktop/en-us/forecast_how_it_works.htm` (visited on 07/27/2024) (cit. on p. 12).

[47] Salesforce, Inc., *Create a forecast*. [Online]. Available: `https://help.tableau.com/current/pro/desktop/en-us/forecast_create.htm` (visited on 07/27/2024) (cit. on p. 13).

[48] Salesforce, Inc., *How predictive modeling functions work in Tableau*. [Online]. Available: `https://help.tableau.com/current/pro/desktop/en-us/predictions_overview.htm` (visited on 07/27/2024) (cit. on p. 14).

[49] Salesforce, Inc., *Forecasting*. [Online]. Available: `https://help.tableau.com/current/pro/desktop/en-us/forecasting.htm` (visited on 07/27/2024) (cit. on p. 15).

[50] Microsoft, *Use the analytics pane in Power BI Desktop*. [Online]. Available: `https://learn.microsoft.com/en-us/power-bi/transform-model/desktop-analytics-pane#apply-forecasting` (visited on 07/27/2024) (cit. on p. 15).

[51] Microsoft, *Create smart narrative summaries*. [Online]. Available: `https://learn.microsoft.com/en-us/power-bi/visuals/power-bi-visualization-smart-narrative` (visited on 07/27/2024) (cit. on p. 15).

[52] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*, Fifth Edition. John Wiley & Sons, 2015 (cit. on pp. 21, 22).

[53] M. Arltová and D. Fedorová, "Selection of unit root test on the basis of length of the time series and value of ar (1) parameter," *Statistika: Statistics & Economy Journal*, vol. 96, no. 3, 2016 (cit. on p. 21).

[54] M. Giacalone, R. Mattera, and E. Nissi, "Economic indicators forecasting in presence of seasonal patterns: Time series revision and prediction accuracy," *Quality & Quantity*, vol. 54, no. 1, pp. 67–84, 2020 (cit. on p. 21).

[55] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*, Third Edition. Springer, 2002 (cit. on p. 22).

[56] G. M. Goerg, "Testing for white noise against locally stationary alternatives," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 6, pp. 478–492, 2012 (cit. on p. 23).

[57] C. W. Granger, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969 (cit. on p. 23).

[58] R. H. Shumway, D. S. Stoffer, and D. S. Stoffer, *Time series analysis and its applications*. Springer, 2000, vol. 3 (cit. on p. 24).

[59] J. M. Haslbeck, L. F. Bringmann, and L. J. Waldorp, "A tutorial on estimating time-varying vector autoregressive models," *Multivariate Behavioral Research*, vol. 56, no. 1, pp. 120–149, 2021 (cit. on p. 25).

[60] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," *Modeling Financial Time Series with S-PLUS®*, pp. 385–429, 2006 (cit. on p. 25).

[61] P. Gorgi, S. J. Koopman, and J. Schaumburg, "Time-varying vector autoregressive models with structural dynamic factors," *Tinbergen Institute, The Netherlands and Aarhus University, Denmark*, vol. 17, 2017 (cit. on p. 25).

[62] B. S. Bernanke, J. Boivin, and P. Eliasz, "Measuring the effects of monetary policy: A factor-augmented vector autoregressive (favar) approach," *The Quarterly Journal of Economics*, vol. 120, no. 1, pp. 387–422, 2005 (cit. on p. 25).

[63] R. J. Hyndman, "Measuring forecast accuracy," *Business Forecasting: Practical problems and Solutions*, pp. 177–183, 2014 (cit. on pp. 25, 26).

[64] A. Zemlyansky, *ARIMA. Time-series forecasting in browsers and Node.js*. [Online]. Available: `https://github.com/zemlyansky/arima` (visited on 07/14/2024) (cit. on p. 28).

[65] Microsoft, *Typescript*. [Online]. Available: `https://www.typescriptlang.org` (visited on 07/04/2024) (cit. on p. 29).

[66] 4Geeks Academy, *Webapp boilerplate with react js and flask api*. [Online]. Available: `https://github.com/4GeeksAcademy/react-flask-hello` (visited on 07/04/2024) (cit. on p. 29).

[67] The PostgreSQL Global Development Group, *PostgreSQL: The world's most advanced open source relational database*. [Online]. Available: https://www.postgresql.org (visited on 07/05/2024) (cit. on p. 29).

[68] Pallets, *Flask*. [Online]. Available: https://github.com/pallets/flask (visited on 07/05/2024) (cit. on p. 30).

[69] T. G. Smith, *Pmdarima: ARIMA estimators for Python*. [Online]. Available: https://alkaline-ml.com/pmdarima (visited on 07/05/2024) (cit. on p. 31).

[70] J. Perktold, S. Seabold, and J. Taylor, *Statsmodels*. [Online]. Available: https://www.statsmodels.org/stable (visited on 07/05/2024) (cit. on p. 31).

[71] Meta Platforms, Inc., *React - a JavaScript library for building user interfaces*. [Online]. Available: https://react.dev (visited on 07/04/2024) (cit. on p. 31).

[72] I. Meta Platforms, *Virtual DOM and internals*. [Online]. Available: https://legacy.reactjs.org/docs/faq-internals.html#gatsby-focus-wrapper (visited on 08/16/2024) (cit. on p. 31).

[73] Zustand contributors, *Zustand*. [Online]. Available: https://docs.pmnd.rs/zustand (visited on 07/05/2024) (cit. on p. 32).

[74] Lodash core team and contributors, *Lodash*. [Online]. Available: https://lodash.com (visited on 07/05/2024) (cit. on p. 32).

[75] S. Koss, *Date-fns*. [Online]. Available: https://date-fns.org (visited on 07/05/2024) (cit. on p. 32).

[76] Google, *Material design*. [Online]. Available: https://m2.material.io (visited on 07/04/2024) (cit. on p. 33).

[77] MUI, *Material UI*. [Online]. Available: https://mui.com/material-ui (visited on 07/05/2024) (cit. on p. 33).

[78] Wikipedia contributors. "Time series." (2024), [Online]. Available: https://en.wikipedia.org/wiki/Time_series (visited on 07/27/2024) (cit. on p. 34).

[79] OpenAI, *Hello GPT-4o*. [Online]. Available: https://openai.com/index/hello-gpt-4o (visited on 07/13/2024) (cit. on p. 34).

[80] I. Remix Software, *React router*. [Online]. Available: https://reactrouter.com/en/main (visited on 08/18/2024) (cit. on p. 35).

[81] W3C, *The FileReader API*. [Online]. Available: https://www.w3.org/TR/FileAPI/#dfn-filereader (visited on 07/05/2024) (cit. on p. 37).

[82] W3C, *File API*. [Online]. Available: https://www.w3.org/TR/FileAPI (visited on 07/05/2024) (cit. on p. 37).

[83] M. Holt, *Papa Parse*. [Online]. Available: https://www.papaparse.com (visited on 07/05/2024) (cit. on p. 38).

[84] Airbnb, Inc., *Visx: Visualization components for React*. [Online]. Available: https://airbnb.io/visx (visited on 07/05/2024) (cit. on p. 39).

[85] M. Bostock and Observable, Inc., *D3 by Observable*. [Online]. Available: https://d3js.org (visited on 07/05/2024) (cit. on p. 39).

[86] E. R. Tufte, *Envisioning information*. Graphics Press LLC, 1990 (cit. on p. 42).

[87] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck, "Importance-driven visualization layouts for large time series data," in *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, IEEE, 2005, pp. 203–210 (cit. on p. 42).

[88] Mozilla Corporation, *Event bubbling*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Building_blocks/Event_bubbling (visited on 07/27/2024) (cit. on p. 44).

[89] Airbnb, Inc., *@visx/tooltip*. [Online]. Available: https://airbnb.io/visx/docs/tooltip (visited on 07/27/2024) (cit. on p. 44).

[90] I. Meta Platforms, *createPortal*. [Online]. Available: https://react.dev/reference/react-dom/createPortal (visited on 07/27/2024) (cit. on p. 44).

[91] kukuroo3, *Mosquito indicator in Seoul, Korea*. [Online]. Available: https://www.kaggle.com/datasets/kukuroo3/mosquito-indicator-in-seoul-korea (visited on 09/01/2024) (cit. on pp. 59–62).

[92] U. Buttar, *Cryptocurrency historical prices [updated daily]*. [Online]. Available: https://www.kaggle.com/datasets/usamabuttar/cryptocurrency-historical-prices-updated-daily (visited on 09/16/2024) (cit. on pp. 63, 64).

[93] P. Rosen, A. Suh, C. Salgado, and M. Hajij, “Topolines: Topological smoothing for line charts,” *arXiv preprint arXiv:1906.09457*, 2019 (cit. on p. 67).

Appendices

A. Line chart hovered point detection

This appendix presents a simplified example of the JSX component for detecting the coordinates of a hovered point on a time series line chart. ChartOverlays component includes the transparent Bar component for capturing the hover events and two Line components that the crosshair. ChartOverlays component is put on top of the main chart content inside an SVG element.

```
import { localPoint } from '@visx/event';
import { Group } from '@visx/group';
import { Bar, Line } from '@visx/shape';
import { useState, useCallback } from 'react';

const ChartOverlays = ({ width, height }) => {
  const [pointerCoordinates, setPointerCoordinates] =
    useState({ x: undefined, y: undefined });

  const onHover = useCallback((event) => {
    const { x, y } = localPoint(event.target, event);
    // Save the coordinates in the state
    // to use them for crosshair and tooltip positioning
    setPointerCoordinates({ x, y });
    // Continue handling the interaction
  }, []);

  const onLeave = useCallback((event) => {
    setPointerCoordinates({ x: undefined, y: undefined });
    // Continue handling the interaction
  }, []);

  return (
    <Group onMouseMove={onHover} onMouseLeave={onLeave}>
```

```
      <Bar // Rectangle for capturing the hover events
        width={width}
        height={height}
        fill="transparent"
        pointerEvents="all"
      />
      <Group pointerEvents="none">
        <Line // Vertical crosshair
          from={{ x: pointerCoordinates.x, y: 0 }}
          to={{ x: pointerCoordinates.x, y: height }}
          stroke="black"
          strokeDasharray="3,6"
        />
        <Line // Horizontal crosshair
          from={{ x: 0, y: pointerCoordinates.y }}
          to={{ x: width, y: pointerCoordinates.y }}
          stroke="black"
          strokeDasharray="3,6"
        />
      </Group>
    </Group>
  );
};
```