A Framework for Simple Integration of Time Series Analytics and Predictions into Applications

Master Thesis at the

University of Applied Sciences Ulm

Faculty of Computer Science

Master Intelligent Systems

submitted by

Patrick Beinlich

Matriculation number: 3138844

14. September 2022

Supervisor 1: Prof. Dr. Markus Goldstein

Supervisor 2: Prof. Dr. Volker Herbort

Declaration of independence

I hereby declare that this Master Thesis is entirely the result of my own work except where otherwise indicated. Only the resources given in the list of references where used. All quotations, both verbatim and in spirit, are identified as such in this work.

Altenstadt, 14. September 2022 Signature: _____

Abstract

Machine learning becomes more and more important in the modern world. Intelligent robots clean rooms or serve food and drinks in restaurants and intelligent algorithms help finding missing persons.

They also become more important in the business world, where it is important to analyse time series data and get a good prediction of the future. This can help to ensure that enough but not too many resources are available. Intelligent prediction algorithms can for example help to predict how many products probably will be sold or how the business figures of a company are developing.

Not every developer has experience in the field of machine learning and it is not always easy to find the right time series prediction model for the specific problem. The target of this master's thesis is to research, how a framework can be developed, which enables inexperienced developers to train simple time series prediction models with no knowledge and helps more experienced developers to find the right prediction model for their problem.

Contents

List of Figures V

List of Tables VII

List of Algorithms XI

Acronyms XII

Glossary XIII

1. Introduction 1

2. Theoretical Background 2

2.1. Software Architecture Patterns . . . 2
2.2. Communication Protocols . . . 4
2.2.1. Representional State Transfer [1] . . . 4
2.2.2. WebSockets . . . 4
2.3. Time Series . . . 4
2.4. Time Series Forecasting Models . . . 5
2.4.1. Autoregression . . . 5
2.4.2. Moving Average . . . 6
2.4.3. Autoregressive Integrated Moving Average . . . 6
2.4.4. Seasonal Autoregressive Integrated Moving Average . . . 7
2.4.5. Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components . . . 7
2.4.6. Exponential Smoothing . . . 8
2.4.7. Prophet . . . 8
2.4.8. Recurrent Neural Network . . . 9
2.4.9. Long Short-Term Memory . . . 9
2.4.10. Gated Recurrent Unit . . . 10
2.5. Model Performance Evaluation . . . 11
2.5.1. Mean Square Error . . . 11

2.5.2. Root Mean Square Error 11
2.5.3. Akaike Information Criterion 11
2.5.4. Bayesian Information Criterion 12

3. Software Architecture for Predictive Analytics 14
3.1. Framework Architecture 14
3.1.1. Selected Framework Architecture Pattern for Predictive Analytics Framework 14
3.1.2. Communication Protocol 19
3.1.3. Application Programming Interface 20
3.2. Understanding and Preparation of Time Series Data for the Predictive Analytics Framework 20
3.2.1. Data Preparation 21
3.2.2. Data Understanding 24
3.3. Additional Characteristics of the Predictive Analytics Framework 25
3.3.1. Dataset Format 25
3.3.2. Multivariate Forecasting 25

4. Time Series Prediction 26
4.1. Selected Time Series Prediction Models for the Predictive Analytics Framework 26
4.1.1. Selected Models 26
4.1.2. Not Selected Models 28
4.2. Model Parameter Optimization 28
4.2.1. Autoregression 29
4.2.2. Seasonal Autoregressive Integrated Moving Average 31
4.2.3. Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components 31
4.2.4. Exponential Smoothing 31
4.2.5. Long Short-Term Memory / Gated Recurrent Unit 32
4.2.6. Prophet 32

4.3. Automatic Prediction Model Selection 32
4.4. Test Datasets 33
4.4.1. Visit Counter Ulm 34
4.4.2. Covid-19 Numbers Singapore 37
4.4.3. 8-Channel EKG-Signal 38
4.4.4. Power Consumption Germany 40
4.4.5. Tesla Inc Stock Market Value 41
4.4.6. Water Temperature Well Ulm 43
4.5. Evaluation Results 43
4.5.1. Problems During Evaluation and Testing 46
4.5.2. Evaluation Conclusion 47

5. Conclusion **52**

6. References **54**

A. Dataset Decomposition **i**
A.1. Visit Counter Ulm i
A.2. Covid-19 Numbers Singapore ix
A.3. 8-Channel EKG-Signal xiii
A.4. Power Consumption Germany xxi
A.5. Tesla Inc Stock Market Value xxiii
A.6. Water Temperature Well Ulm xxviii

B. Application Programming Interface **xxviii**
B.1. Data Application Programming Interface xxviii
B.2. File Application Programming Interface xxx
B.3. Models Application Programming Interface xxx
B.4. Info Application Programming Interface xxxii
B.5. Auto Application Programming Interface xxxiii

C. Evaluation **xxxiv**
C.1. Evaluation Results xxxiv
C.1.1. Akaike Information Criterion Results xxxiv
C.1.2. Bayesian Information Criterion Results xxxvi
C.1.3. Mean Square Error Results xl

C.1.4. Root Mean Square Error Results xlv
C.1.5. Training Duration Results xlix
C.2. Parameter of the Best Model lii
C.2.1. Autoregression liii
C.2.2. Seasonal Autoregressive Integrated Moving Average lvi
C.2.3. Seasonal Autoregressive Integrated Moving Average (SARIMA) grid search lix
C.2.4. Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components lxii
C.2.5. Exponential Smoothing lxv
C.2.6. Long Short-Term Memory lxviii
C.2.7. Gated Recurrent Unit lxxi

List of Figures

1. Comparison of monolithic and microservice architecture patterns [7] . . . 3
2. Simplified architecture of a Recurrent Neural Network (RNN) [19] . . . 9
3. Simplified architecture of a Long Short-Term Memory (LSTM) [19] . . . 10
4. Simplified architecture of a Gated Recurrent Unit (GRU) [19] . . . 10
5. Survey results for the question if rather the microservice or the monolithic architecture is preferred . . . 15
6. General Predictive Analytics Framework architecture . . . 16
7. Internal Predictive Analytics Framework architecture . . . 17
8. General model module architecture . . . 18
9. Data preparation steps in the preparation feature . . . 21
10. Seasonal decompose of the visit counter for the Gloecklerstrasse Busparkplatz . . . i
11. Seasonal decompose of the visit counter for the Gloecklerstrasse Hirschstrasse . . . ii
12. Seasonal decompose of the visit counter for the Sedelhof Passage . . . iii
13. Seasonal decompose of the visit counter for the Platzgasse Herrenkellergasse . . . iv
14. Seasonal decompose of the visit counter for the Platzgasse Kohlgasse . . . v
15. Seasonal decompose of the visit counter for the Hirschstrasse Muensterplatz . . . vi
16. Seasonal decompose of the visit counter for the Hafenbad Hafengasse . . . vii
17. Seasonal decompose of the visit counter for the Hanz und Sophie Scholl Platz . . . viii
18. Seasonal decompose of the number of Covid-19 infection in Singapore . . . ix

19. Seasonal decompose of the cumulative number of Covid-19 infection in Singapore x
20. Seasonal decompose of the number of Covid-19 deaths in Singapore xi
21. Seasonal decompose of the cumulative number of Covid-19 deaths in Singapore xii
22. Seasonal decompose of Channel 1 of the ekg-signal xiii
23. Seasonal decompose of Channel 2 of the ekg-signal xiv
24. Seasonal decompose of Channel 3 of the ekg-signal xv
25. Seasonal decompose of Channel 4 of the ekg-signal xvi
26. Seasonal decompose of Channel 5 of the ekg-signal xvii
27. Seasonal decompose of Channel 6 of the ekg-signal xviii
28. Seasonal decompose of Channel 7 of the ekg-signal xix
29. Seasonal decompose of Channel 8 of the ekg-signal xx
30. Seasonal decompose of the German power consumption from 04/2021 to 04/2022 xxi
31. Seasonal decompose of the German power consumption from 01/2015 to 07/2022 xxii
32. Seasonal decompose of the Tesla Inc Stock Market Open Value xxiii
33. Seasonal decompose of the Tesla Inc Stock Market Close Value xxiv
34. Seasonal decompose of the Tesla Inc Stock Market Volume Value xxv
35. Seasonal decompose of the Tesla Inc Stock Market Low Value xxvi
36. Seasonal decompose of the Tesla Inc Stock Market High Value xxvii
37. Seasonal decompose of the water temperatur at the well at the Weinhof in Ulm xxviii

List of Tables

3. Visit counter Gloecklerstrasse Busparkplatz dataset information . . . 35
4. Visit counter Gloecklerstrasse Hirschstrasse dataset information . . . 35
5. Visit counter Sedelhof Passage dataset information . . . 35
6. Visit counter Platzgasse Herrenkellergasse dataset information . . . 35
7. Visit counter Platzgasse Kohlgasse dataset information . . . 36
8. Visit counter Hirschstrasse Muensterplatz dataset information . . . 36
9. Visit counter Hafenbad Hafengasse dataset information . . . 36
10. Visit counter Hans und Sophie Scholl Platz dataset information . . . 36
11. Covid-19 confirmed cases dataset information . . . 37
12. Covid-19 confirmed cases cumulative dataset information . . . 37
13. Covid-19 deaths dataset information . . . 38
14. Covid-19 deaths cumulative dataset information . . . 38
15. Ekg channel 1 dataset information . . . 38
16. Ekg channel 2 dataset information . . . 39
17. Ekg channel 3 dataset information . . . 39
18. Ekg channel 4 dataset information . . . 39
19. Ekg channel 5 dataset information . . . 39
20. Ekg channel 6 dataset information . . . 39
21. Ekg channel 7 dataset information . . . 40
22. Ekg channel 8 dataset information . . . 40
23. Power consumption Germany 04/21 - 04/22 dataset information . . . 41
24. Power consumption Germany 01/15 - 07/22 dataset information . . . 41
25. Tesla stock open dataset information . . . 41
26. Tesla stock close dataset information . . . 42
27. Tesla stock volume dataset information . . . 42

28. Tesla stock low dataset information 42
29. Tesla stock high dataset information 42
30. Water temperature dataset information 43
31. Resolutions used for the different datasets 44
32. Dataset abbreviations used for compact representation 45
33. Best models for the visit counter datasets 4.4.1 47
34. Best models for the Covid datasets 4.4.2 48
35. Best models for the ekg datasets 4.4.3 48
36. Best models for the power consumption 4.4.4 and water temperature 4.4.6 49
37. Best models for the Tesla stock datasets 4.4.5 49
38. Akaike information criterion (AIC) evaluation results visit counter datasets xxxiv
39. AIC evaluation results Covid datasets xxxv
40. AIC evaluation results ekg datasets xxxv
41. AIC evaluation results power consumption and water temperature datasets xxxvi
42. AIC evaluation results Tesla stock datasets xxxvi
43. Bayesian information criterion (BIC) evaluation results visit counter datasets xxxvii
44. BIC evaluation results Covid datasets xxxvii
45. BIC evaluation results ekg datasets xxxviii
46. BIC evaluation results power consumption and water temperature datasets xxxviii
47. BIC evaluation results Tesla stock datasets xxxix
48. Mean Square Error (MSE) evaluation results visit counter datasets xl
49. MSE evaluation results Covid datasets xli
50. MSE evaluation results ekg datasets xlii
51. MSE evaluation results power consumption and water temperature datasets xliii
52. MSE evaluation results Tesla stock datasets xliv
53. Root Mean Square Error (RMSE) evaluation results visit counter datasets xlv

54. RMSE evaluation results Covid datasets xlvi
55. RMSE evaluation results ekg datasets xlvii
56. RMSE evaluation results power consumption and water temperature datasets xlviii
57. RMSE evaluation results Tesla stock datasets xlix
58. Training duration visit counter datasets l
59. Training duration Covid datasets l
60. Training duration ekg datasets li
61. Training duration power consumption and water temperature datasets li
62. Training duration Tesla stock datasets lii
63. Parameters of best Autoregression (AR) models of visitor datasets liii
64. Parameters of best AR models of Covid datasets liv
65. Parameters of best AR models of ekg datasets liv
66. Parameters of best AR models of power consumtion and water temperature datasets lv
67. Parameters of best AR models of tesla stock datasets lv
68. Parameters of best SARIMA models of visitor datasets lvi
69. Parameters of best SARIMA models of Covid datasets lvi
70. Parameters of best SARIMA models of ekg datasets lvii
71. Parameters of best SARIMA models of power consumption and water temperature datasets lvii
72. Parameters of best SARIMA models of Tesla stock datasets lviii
73. Parameters of best SARIMA grid search models of visitor datasets lix
74. Parameters of best SARIMA grid search models of Covid datasets lix
75. Parameters of best SARIMA grid search models of ekg datasets lx
76. Parameters of best SARIMA grid search models of power consumption and water temperature datasets lx

77. Parameters of best SARIMA grid search models of Tesla stock datasets lxi
78. Parameters of best Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components (TBATS) models of visitor datasets lxii
79. Parameters of best TBATS models of Covid datasets lxii
80. Parameters of best TBATS models of ekg datasets lxiii
81. Parameters of best TBATS models of power consumption and water temperature datasets lxiii
82. Parameters of best TBATS models of Tesla stock datasets lxiv
83. Parameters of best Exponential Smoothing (ES) models of visitor datasets lxv
84. Parameters of best ES models of Covid datasets lxv
85. Parameters of best ES models of ekg datasets lxvi
86. Parameters of best ES models of power consumption and water temperature datasets lxvi
87. Parameters of best ES models of Tesla stock datasets lxvii
88. Parameters of best LSTM models of visitor datasets lxviii
89. Parameters of best LSTM models of Covid datasets lxviii
90. Parameters of best LSTM models of ekg datasets lxix
91. Parameters of best LSTM models of power consumption and water temperature datasets lxix
92. Parameters of best LSTM models of Tesla stock datasets lxx
93. Parameters of best GRU models of visitor datasets lxxi
94. Parameters of best GRU models of Covid datasets lxxi
95. Parameters of best GRU models of ekg datasets lxxii
96. Parameters of best GRU models of power consumption and water temperature datasets lxxii
97. Parameters of best GRU models of Tesla stock datasets lxxiii

List of Algorithms

1. Missing entry detection 23
2. AR lag determination 30

Acronyms

AI	artificial intelligence.
AIC	Akaike information criterion.
API	Application Programming Interface.
AR	Autoregression.
ARIMA	Autoregressive Integrated Moving Average.
BIC	Bayesian information criterion.
ES	Exponential Smoothing.
GRU	Gated Recurrent Unit.
LSTM	Long Short-Term Memory.
MA	Moving Average.
MSE	Mean Square Error.
NN	Neural Network.
PAF	Predictive Analytics Framework.
REST	Representional State Transfer [1].
RMSE	Root Mean Square Error.
RNN	Recurrent Neural Network.
SARIMA	Seasonal Autoregressive Integrated Moving Average.
TBATS	Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components.
TCP	Transmission Control Protocol [2].

Glossary

Application Programming Interface	An interface that allows developers easy access to functionality of an application [3].
lag(n)	n-th previous value in a time series.

Acknowledgements

I would like to thank my supervisor Prof. Dr. Markus Goldstein, as well as my external supervisors Matthias Eck and Burkhard Hoppenstedt from the eXXcellent solutions GmbH in Ulm, Germany, for their guidance and advice throughout this project.

1. Introduction

Many businesses use artificial intelligence (AI) for lots of operations like analysing data, optimizing and automating processes and customer services like individualized purchase suggestions [4]. Another area where AI is used in businesses is to forecast different types of properties like the amount of sold goods, needed resources or the price of those resources. The problem is, that not every developer has experience in the field of machine learning furthermore in the field of time series forecasting. In a survey conducted at eXXcellent solutions GmbH in Ulm and at an online survey platform, only 11 of the 26 participating developers stated to have experience in the field of machine learning and only 2 listed that they have experience with time series forecasting.

Use cases for time series forecasting models are diverse. From predictions of product costs to the number of people that got infected by a contagious disease or the course of stock market values. There are various types of time series with different properties. Some contain strong seasonality, like sensor data from a thermometer, and other strong trends, like information about the growth of bacteria colonies. There are also various types of applications those data can be used in, a stock market trading algorithm differs from an application that predicts the amount of a harvest, but both use time series forecasting models internally. For a developer with no experience this may be complex and overbearing.

Too help developers with no or only little experience in the field of time series prediction a framework could help to train several types of models and select fitting ones. This framework could even help experienced developers with the selection of a suitable model for their specific use case. It is not always needed to use the most precise model, in some cases a fast training model is preferred, even if it has a small precision penalty. The Predictive Analytics Framework (PAF) can help developers find a suitable model and provide them with the basic parameters of the time series forecasting models to help the developer to optimize the model further.

To support both inexperienced and experienced developers working with time series prediction, a framework is developed, which prepares data for the use in time series prediction models, trains a collection of models and selects the best of these models.

2. Theoretical Background

In this chapter terms are explained and information is summarized to give a quick overview of needed knowledge. It will introduce some software architecture patterns, some basics about time series, some time series forecasting models and how those models can be evaluated.

2.1. Software Architecture Patterns

Most applications follow certain architectural patterns, that define how the software is built internally. The two most commonly used software architecture patterns are the monolithic and the micro service architecture pattern [5]. Besides those two popular architecture patterns there are several different other patterns which could be used for the implementation of the Predictive Analytics Framework (PAF). One of those is the nanoservice architecture pattern.

The following sections will describe the general idea behind those software architecture patterns and their general benefits and drawbacks.

Monolithic Architecture

In the monolithic architecture all components of the application are combined in a single program [5]. All components like the data access layer, the UI, the business logic are developed and deployed in a single program as displayed in figure 1.

Because all parts of the application are contained in one program, the complexity and therefore the costs of the software is initially lower [6]. And because there are no communications between different programs which are a security risk, the safety of the monolithic software architecture is also better compared to other solutions [6].

The larger an application becomes, the more the disadvantages of the monolithic architecture patterns become apparent. Because all components are contained in one application, they are stronger linked together, which prevents for example the scalability [6] of components and the independent development.

Microservice Architecture

In contrast to the monolithic software architecture the microservice architecture pattern separates the different component into independent, often stateless, services

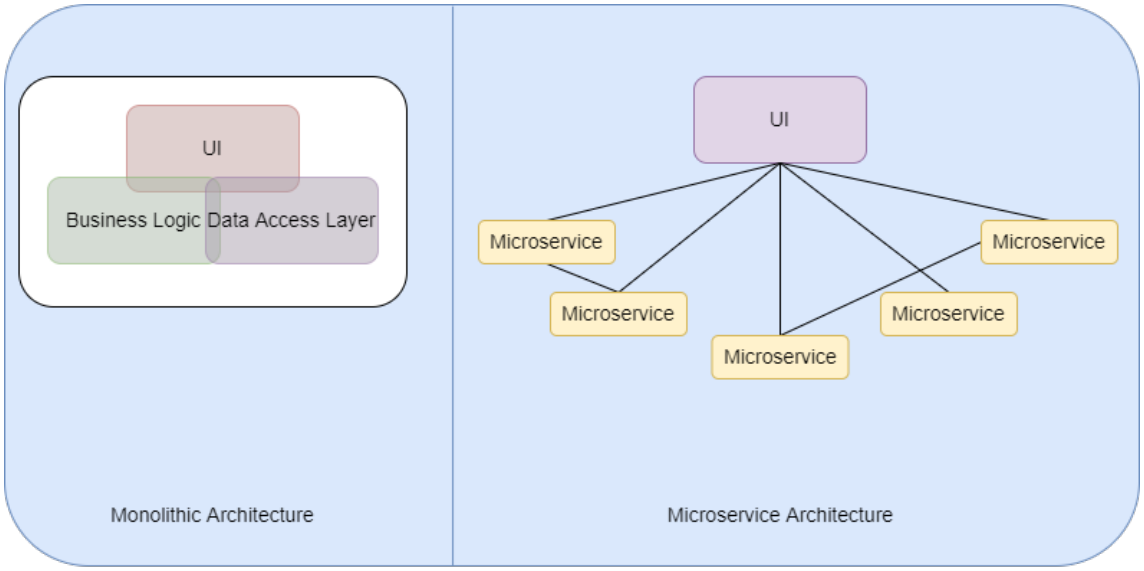

Figure 1: Comparison of monolithic and microservice architecture patterns [7]

[5], like shown in figure 1. Those different components communicate with each other most of the times through web Application Programming Interfaces (APIs) [6] which use websockets or follow the Representional State Transfer [1] (REST) pattern and can be placed on different real machines.

The benefits of this approach is that the separated services are easier to maintain, scale and share [6]. Each component of the application runs in at least one service, if those services are stateless they can be easily scaled by adding a loadbalancer and additional instances of the utilized services to eliminate bottlenecks. Because the different services are independently running applications which communicate over web APIs, they can be developed more independently and exchanged more easily without the need to adjust the opposite site [6].

Because each service is basically an own application the complexity of the complete program and therefore the cost is increased in comparison to the monolithic architecture pattern. In addition to this, the communication over web APIs add additional security risks to the program [6].

Nanoservice Architecture

The nanoservice software architecture pattern takes the general idea of the microservice pattern to another level. Instead of only separating the application into

several logical microservices it is split into nanoservices, where each of these services only perform a single function [6]. The goal behind this pattern is to amplify the existing benefits of microservices by splitting the application into even smaller independent services [8].

This approach does not only increase the benefits of the microservice architecture pattern, but also its drawbacks. Additionally to that the resource consumption is increased because of the large number of services [6].

2.2. Communication Protocols

Two simple and often used communication types used for the communication between different applications or application parts are interfaces developed using the REST design principle and WebSockets [9]. These are also the candidates for the protocol used to enable the using application to access the PAF.

2.2.1. Representional State Transfer [1]

REST is fundamentally just a collection of design principles that uses the HTTP protocol as a communication method [9]. It is unidirectional, which means that only the client can send messages to the server and the server answers this request. REST is also stateless, meaning that every request is a new connection with no information of prior requests [9].

2.2.2. WebSockets

WebSockets are other than REST statefull, which can help when two application exchange information frequently [9]. Additionally it only works over the Transmission Control Protocol [2] (TCP) protocol and supports bidirectional communication, which allows for continuous connections [9].

2.3. Time Series

A time series is a series of information over time. This can be values like temperature or prices of goods, but also states like the current state of a traffic light. In the context of this work only the series of decimal or integer values are relevant. Those can be used for example to represent the gathered data from sensors like temperature or

humidity development or to represent the development over time of values like prices of goods or number of persons.

For training of time series forecasting models some properties of the time series are important. Those are if the time series is stationary, if it contains seasonality and if there are correlations in the series [10]. Stationarity means, that the time series has a constant mean and variance over the complete time series, the seasonality contains information about recurring behaviour of the time series, like a spike every year and the correlation shows how much a value is influenced by its predecessors, also called lags [11]. A lag(n) is the n-th previous value, for example lag(5) is the 5th value before the current. The information of the correlation can also be used to retrieve information about seasonality. If for monthly data the lags 12, 24, 36, ... have a high correlation, this would indicate a yearly seasonality.

2.4. Time Series Forecasting Models

Time series forecasting models are important so the datasets can be used to predict the future. These models are one of the core parts of the Predictive Analytics Framework (PAF). This chapter introduces a collection of possible candidates that could be implemented in the PAF.

2.4.1. Autoregression

Autoregression (AR) uses the previous lags to create an prediction for future values [12].

It uses a constant value c (often the mean value) and adds and error ϵ and p lags multiplied with a coefficient β to change their effect on the prediction.

$y_t = c + \epsilon_t + \sum_{i=1}^{p} \beta_i y_{t-i}$ [11]

p = number of lags (previous values) used

ϵ = error

c = constant

y = real value

β = weight of the value

Those coefficients β can be used to determine how much influence which lag has on the prediction, which can be used to help with seasonality if for example the impact of the value of the same day the previous week is weighted more than the

previous days value.

2.4.2. Moving Average

Moving Average (MA) models work like AR models, with the exception that not the previous values but the errors are used to derive the new values [13].

$y_t = c + \epsilon_t + \sum_{i=1}^{q} \beta_i \epsilon_{t-i}$ [11]

q = number of order (previous values) used

ϵ = error

c = constant

β = weight of the value

Equal to AR, the Moving Average (MA) model contains a weight for each order to map recurring patterns like seasonality.

2.4.3. Autoregressive Integrated Moving Average

The Autoregressive Integrated Moving Average (ARIMA) model uses a combination of AR and MA. This would make it only usable for stationary time series. To negate this problem, the AR part of this model does not use the original values, but differences them [11].

$y_t = c + \epsilon_t + \sum_{i=1}^{p} \alpha_i y'_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-1}$[11]

c = constant

ϵ = error

p = number of lags (previous values) used

q = number of order (previous errors) used

y = real value

y' = differenced value

α = weight of the value

β = weight of the value

These models are called ARIMA(p,d,q) models with [11]:

- p = order of AR part
- d = degree of first differencing
- q = order of MA part

Because the Autoregressive Integrated Moving Average (ARIMA) model basically contains the AR and MA model, they can be used with it too. By setting only the p order an AR model and by only using the q order a MA model is created.

2.4.4. Seasonal Autoregressive Integrated Moving Average

An extension of the Autoregressive Integrated Moving Average (ARIMA) model is the SARIMA model, where additional terms for seasonality are added. These models are described as SARIMA(p,d,q)(P,D,Q)m or SARIMA(p,d,q)(P,D,Q,M) [11]:

- m,M = number of observations per year
- P = order of AR part
- D = degree of first differencing
- Q = order of MA part

The seasonal terms are multiplied by the non-seasonal terms [11]

2.4.5. Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components

Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components (TBATS) is a model that can be used for forecasting with complex seasonal patterns [14]. Simular to the Predictive Analytics Framework (PAF) it considers various models and chooses one of them.

The different model configurations considered are [14]:

- with/without Box-Cox transformation
- with/without Trend
- with/without Trend Damping
- with/without ARMA(p,q)
- non-seasonal model
- various amounts of harmonics used for seasonal effect

Models with different configurations of those options are trained and evaluated via the Akaike information criterion [14]. The model with the smallest criterion is then selected and returned.

2.4.6. Exponential Smoothing

Exponential Smoothing (ES) is a time series forecasting model that is often compared to ARIMA. Similar to ARIMA it uses a weighted sum of past values for the predictions, but those weights are exponentially decreasing for past values for the ES model [15]. This means that the most recent values are weighted higher.

There are three main types of ES, which are Single or Simple Exponential Smoothing, Double Exponential Smoothing and Triple Exponential Smoothing [15]. The difference between them is, that Double Exponential Smoothing adds support for trend and Triple Exponential Smoothing support for seasonality to the model.

2.4.7. Prophet

Another regression model for time series prediction that is often mentioned is the Prophet model from Meta Platforms Ireland Limited, former Facebook Ireland Limited.

This model is a additive regression model with four main components according to the official side from the Meta Platforms Ireland Limited [16]:

- A linear or logistic growth curve trend
- A yearly seasonal component
- A weekly seasonal component
- A user-provided list of holidays

This model detects the seasonality itself and is capable of multivariate predictions [17]

The disadvantage of this model is that it requires two columns 'ds' and 'y' [18]. The 'y' column is the column which should be predicted. The 'ds' column can be a problem for some datasets, because it is a column for dates. If a dataset does not have a date column, it would have to be added artificially, which could result in mistakes from the model.

2.4.8. Recurrent Neural Network

A Recurrent Neural Network (RNN) is a Neural Network (NN) where the previous values are integrated into the prediction [19], it is reused in the following predictions, as indicated in figure 2. The advantage of this is, that the NN output is influenced by multiple previous values. The disadvantage is, that only a limited amount of recent values is considered. For data in a daily interval with a yearly seasonality an at least 365 large layer would be needed to represent this seasonal pattern.

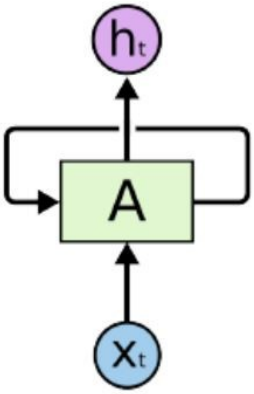

Figure 2: Simplified architecture of a RNN [19]

2.4.9. Long Short-Term Memory

The Long Short-Term Memory (LSTM) is a special implementation of the RNN where the NN cell stores the information about the previous value itself. It contains a cell state and 3 gates, as can be seen in figure 3. These gates are the input, forget and output gates. The input gate defines if the cell is updated, the forget gate resets the cell state if activated and the output gate controls if the current state is shown [20]. This allows the model to bridge long time lags [19]

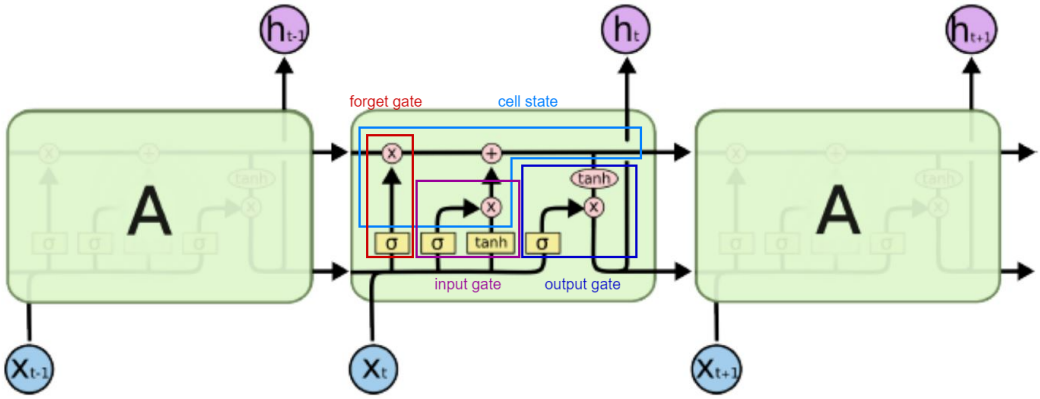

Figure 3: Simplified architecture of a LSTM [19]

2.4.10. Gated Recurrent Unit

A Gated Recurrent Unit (GRU) is a simplified version of the LSTM. It only contains two gates the update and the reset gate [19]. The update gate replaces the input and forget gate and the Reset gate determines how long the information is stored in the cell.

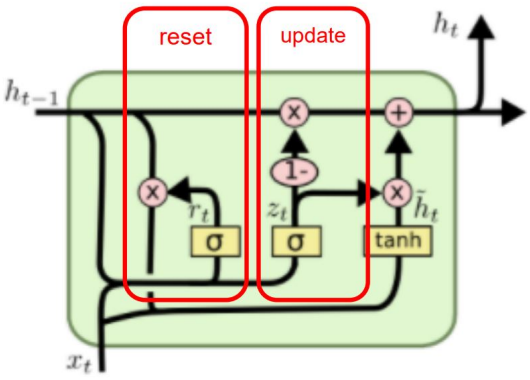

Figure 4: Simplified architecture of a GRU [19]

2.5. Model Performance Evaluation

To compare different time series forecasting models with each other, it is necessary to calculate an evaluation metric that is comparable. Some common evaluation methods are the mean squared error, root mean squared error, akaike information criterion and the bayesian information criterion. The general idea behind those four evaluation methods and some advantages and disadvantages are explained below.

2.5.1. Mean Square Error

The Mean Square Error (MSE) is a generally usable way to get a measurement for the accuracy of a prediction. It compares the predicted values with the set of actual observed values. It is calculated by adding the squared differences between these values up and dividing the result by the number of values [21].

$MSE = \frac{1}{n}\sum_{i=1}^{n}(p_i - o_i)^2$

n = number of values
p = predicted values
o = observed values

Because it uses predicted values, this evaluation metric can be used by every time series forecasting model by splitting the available dataset into a training and testing dataset.

2.5.2. Root Mean Square Error

The Root Mean Square Error (RMSE) is derived from the MSE. It is determined by calculating the square root of the MSE [22]

$RMSE = \sqrt{MSE}$

This has the advantage that the resulting metrics are smaller and therefore easier to read and compare for humans.

2.5.3. Akaike Information Criterion

One commonly used performance measurement used for comparison of time series forecasting models is the Akaike information criterion (AIC). It is used by the automatic lag detection of the autoregression model of the statsmodel package [23] and

the TBATS model [14], among others.

The idea behind this metric is, that not only the difference between the model and the real data affect the score, but also the number of parameters used in the model [24]. This helps to prevent overfitting by penalising models with more parameters.

It takes the maximised value of the log-likelihood function, which is an estimation of how good the model fits the real data, and the number of parameters of the model into account [24].

AIC = 2 k - 2 $\ln(\hat{L})$
k = number of parameters
$\hat{L}$ = maximized likelihood function

The model with the minimum AIC value is assumed to fit the data the best without overfitting.

The disadvantage of this performance measurement is that generally larger models like NN models are at a disadvantage and it is not clear what the correct amount of parameters is, the input values or all parameters (each hidden layer) [25].

Because some models generally have more parameters then others without overfitting, the Akaike information criterion is not well suited to compare different types of models. On the other hand it can be used to compare different models of the same type with each other.

2.5.4. Bayesian Information Criterion

The BIC is related to the Akaike information criterion (AIC) and is also based on the likelihood function [26]. In contrast to the AIC it does not multiply the number of model parameters with 2 but with the log of the number of data points. This results in a heavier weighting of the number of model parameters. In cases where less than 8 data points are available the weight of the number of model parameters would not be increased, but decreased, because the log of 7 and smaller numbers is smaller than 2. Because it is usually not suitable to train a model with so few data points, this problem can be ignored.

BIC = k $\ln(n)$ - 2 $\ln(\hat{L})$
k = number of parameters of the model
n = number of data points

$\hat{L}$ = Maximized likelihood function

3. Software Architecture for Predictive Analytics

For the experience of the using software developer the way he uses the framework and the features it contains is an important part. It has to be defined, how the framework will be implemented in other projects, how it can be accessed and how the interface for this access is build. The framework additionally contains features for the preparation and analysis of the provided data to relieve the developer of this work.

3.1. Framework Architecture

Before an application is developed, it is useful to look at potential software architectures and communication protocols. This knowledge can then be used to potentially decide on the general architecture and the used communication protocols in advance, to prevent the use of an unfitting architecture or communication protocol for the specific problem. Therefor, in this sub-chapter several options for the general application architecture and the communication protocols will be evaluated and fitting ones will be selected. Additionally the interface structure will be explained.

3.1.1. Selected Framework Architecture Pattern for Predictive Analytics Framework

Most new web applications are already implementing the microservice architecture pattern and adding the PAF in form of an additional one represents not a large obstacle. In a survey at eXXcellent solutions GmbH 7 of the developers that participated indicated that they preference the microservice architecture over the monolithic architecture, while only 5 stated the opposite, as it can be seen in figure 5. If the framework would be added in a monolithic pattern into existing programs or business logic services, it would either have to be implemented in the required programming languages or would only be usable for one. By separating the framework in an independent microservice with an web API for the communication between the using application and the framework, the usage of the framework is programming language independent and can be used as long as the chosen web API is supported by the application.

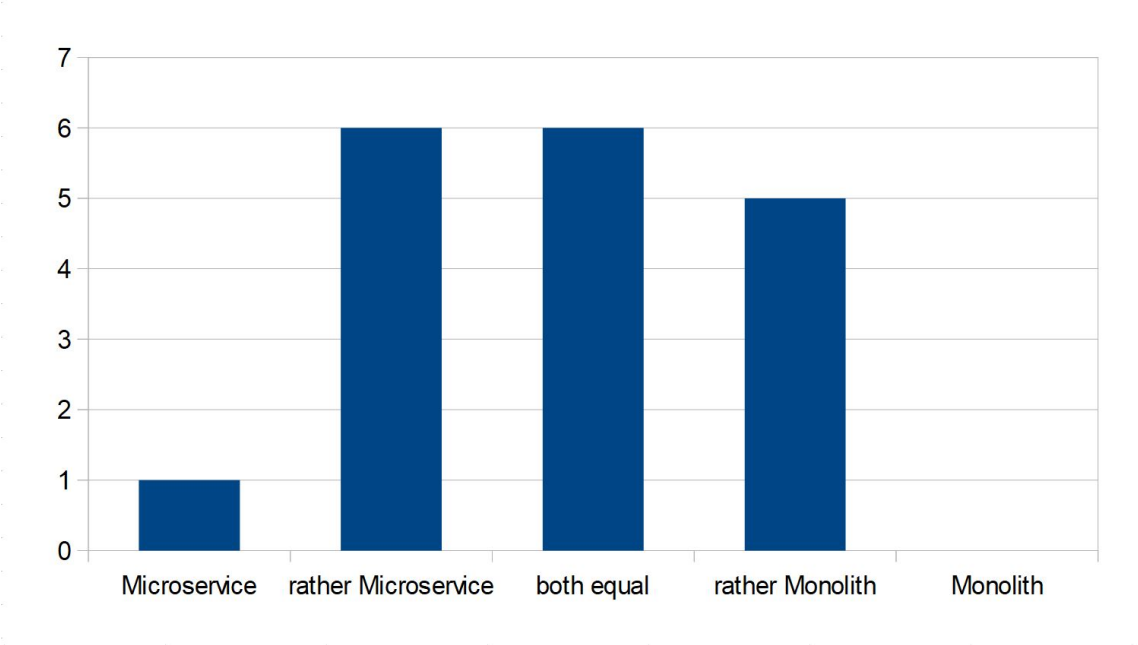

Figure 5: Survey results for the question if rather the microservice or the monolithic architecture is preferred

Because of that, the framework is implemented in form of an microservice with a web API. It can run as a separate server or can be integrated into the microservice architecture of the application, like shown in figure 6.

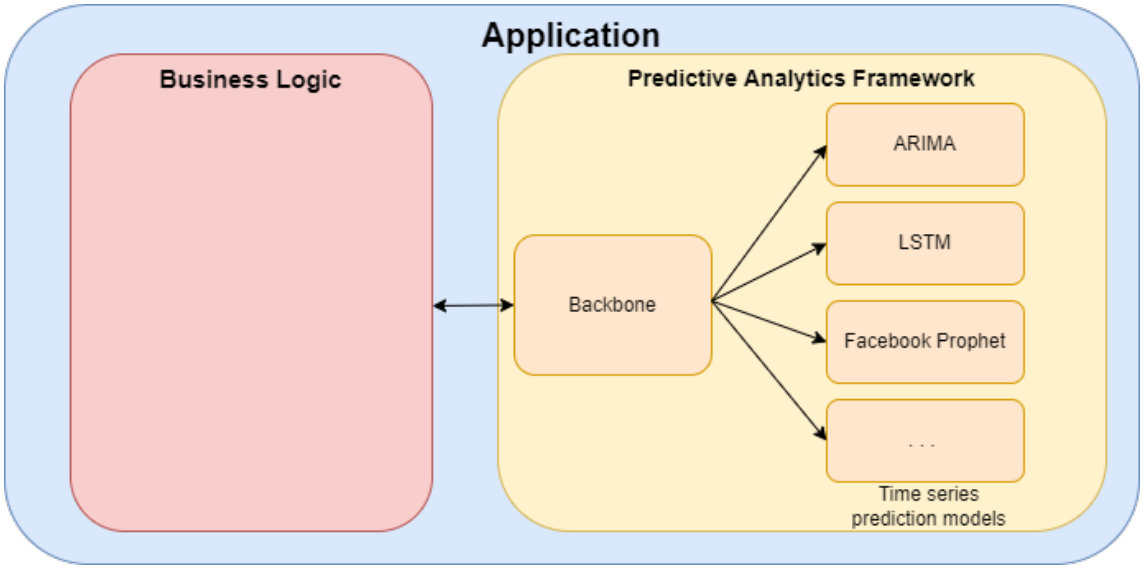

Figure 6: General Predictive Analytics Framework architecture

The application will internally use a monolithic architecture. For smaller applications like the PAF this is well suitable to prevent unneeded complexity and interfaces. This also ensures that the using developer only needs to interact with one interface for data preparation and model training. It also ensures, that all elements of the PAF are available and no important core element is missing.

By selecting this software structure, it is ensured, that all important parts are accessible for the framework. It also simplifies the integration into other projects in comparison to solutions where the different parts of the PAF are separated and have to be integrated individually.

Internal Architecture

Internally the framework is split into three parts, the backbone, the data analysis and preparation tool and the time series prediction models. The structure of the framework can be seen in figure 7.

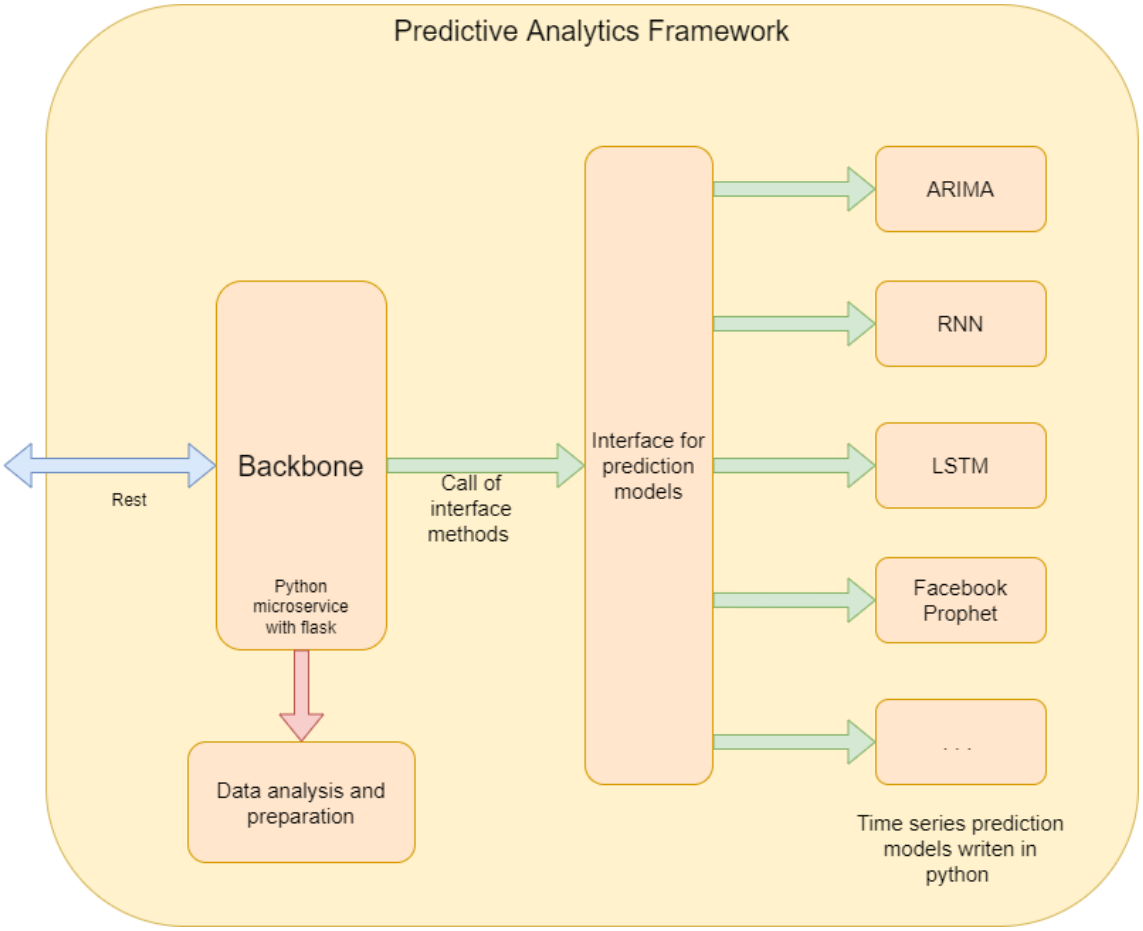

Figure 7: Internal Predictive Analytics Framework architecture

The backbone is responsible for the communication with the using application and has the task to analyse and prepare the data and select a fitting time series model to use for the specific case.

The second part is the data understanding and preparation which is responsible for extracting useful information like minimum, maximum, trend, seasonality and other data from the provided time series. More information about this part of the PAF can be read in section 3.2 of this paper.

The third part are the time series models which are each in a separate module. They implement an abstract class which defines all necessary methods and provides common functionalities like the method for the error calculation. The general architecture can be seen in figure 8. This allows an easier addition of new models. The

time series module is responsible for training the prediction model and optimizing its parameters. Which models where implemented in the release of the framework and why can be read in section 4.1.1.

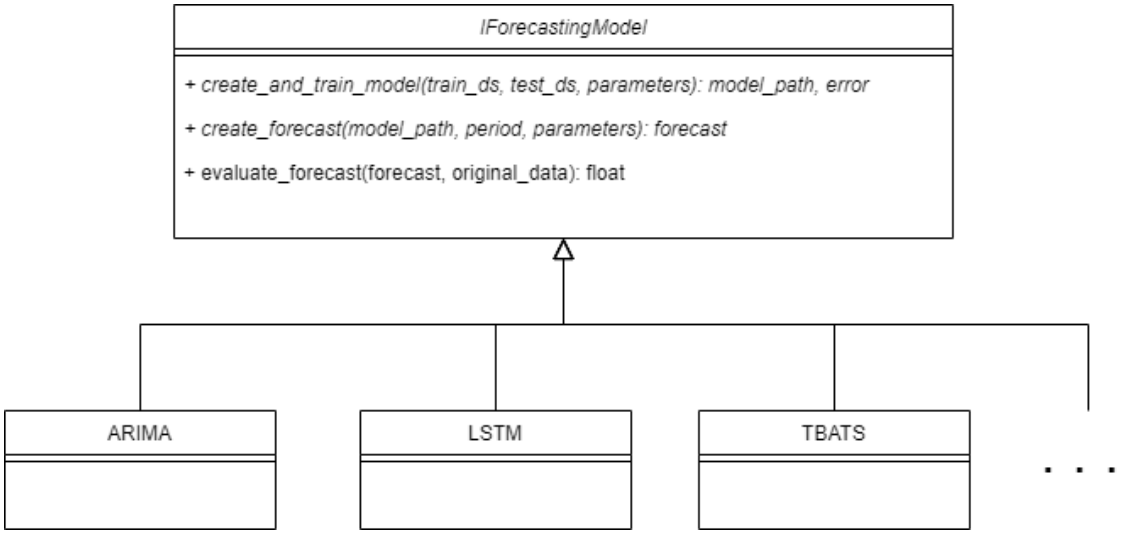

Figure 8: General model module architecture

Programming Language

Because the backbone is separated from the models, different programming languages could be used. The most popular programming language, according to Brian Eastwood from Northeastern University [27], is Python. It is also very popular by developers in the field of machine learning, among other thing because of packages like scikit-learn [28]. For the backbone of many other microservice applications higher level programming languages like java and c# are often used since these are more commonly used and the knowledge base is often larger. In the service done in the eXXcellent solutions GmbH, only 1 of 17 developers selected a Python microservice framework as their preferred one, 13 of those 17 stated that they favor java or c# frameworks. Furthermore the disadvantage of this approach would be that internally the Predictive Analytics Framework would need to maintain two languages. Additionally the communication between the different parts could be more complex to implement. There are rather easy ways to implement a call of a Python script in java but the access of the results is more complex and has to be implemented via the standard output or an extra file.

For some languages there exist libraries which state to help with that but those often have other disadvantages. For java for example there exist the framework Jython [29] which specifies that it provides an easy way to insert Python code into an java application. The disadvantage of this framework is, that it only supports Python version 2.7. Many modern machine learning models like pytorch [30] require more modern versions of Python like version 3.7. This hinders the use of different languages without splitting the framework into different services.

For that reason the programming language used for all parts of the PAF is Python. The For the communication with the using application a REST-API is implemented. More about what framework is used to implement the REST-API can be read in section 3.1.2 and the structure of the API in section 3.1.3.

3.1.2. Communication Protocol

From the in section 2.2 introduced communication protocols, REST is the one that is used in the PAF.

Theoretically, both protocols could be used in the framework. A bidirectional communication is not required for the PAF, because the framework only answers to requests made by a client. The only information the client needs for the use of the framework are the dataset and its information and the ids of the datasets and models in the PAF that should be used.

The reason the REST protocol is used is its widespread availability. Even the address bar of a browser can be partially used to interact with the PAF, for example to download a model or data file. This allows for an easy integration in a wide range of applications.

Using REST as the transition protocol has the additional advantage, that 3rd party REST clients like the Postman REST Client [31] can be used to interact with the PAF. This allows the use of the framework without the integration in another project, if the using developer wants to find out, which model is the best for his dataset or what parameters of a model return good results.

At future versions of the PAF it may be necessary to have statefull and/or bidirectional connections between the clients and the framework for future features. In that case the changeover should not be to difficult due to the separation of the communication and the operating code inside the PAF.

3.1.3. Application Programming Interface

Now that the type of communication protocol used in the PAF is defined, the structure of the interface itself can be designed. There are 5 request paths in the PAF API.

request path: $< host > /api/v1/data/ < action >$

This route is responsible for the data preparation and analysis. It allows the user to analyse the data complete (detect and insert missing entries, interpolate missing values, resample the dataset, find seasonality, decompose the dataset), find missing entries, insert missing entries, interpolate missing values, find seasonal periods, resample the dataset and to decompose the dataset into the trend, seasonal and error components.

request path: $< host > /api/v1/files$

This route allows the user to directly download or upload datasets to the framework

request path: $< host > /api/v1/models$

With this route, the user can train individual models.

request path: $< host > /api/v1/info/ < string : action >$

This route was implemented to give the user a possibility to receive the available univariate and multivariate models.

request path 1: $< host > /api/v1/auto$

This route contains the automatic model selection. The user can either let the framework use all models or give a selection of models to select the best model for a given dataset.

3.2. Understanding and Preparation of Time Series Data for the Predictive Analytics Framework

Before a time series can be used to train a forecasting model it is often necessary to analyse and prepare the data. Some forecasting models require additional infor-

mation like seasonality and sometimes the provided data contains gaps or is in an unfitting resolution, that is sampled too small and detailed for the intended use. To aid the using developers with the preparation of the data and extract needed information independently, the Predictive Analytics Framework contains elements to prepare and analyse the time series provided by the user.

3.2.1. Data Preparation

Before the data can be used to train a model for forecasting or to analyse it, the dataset has to be prepared. To fill in missing entries and values.

For this, the PAF provides an interface to prepare the data before the usage for training. This feature first finds and inserts missing time entries, interpolates those missing data and then resamples the data to a desired interval.

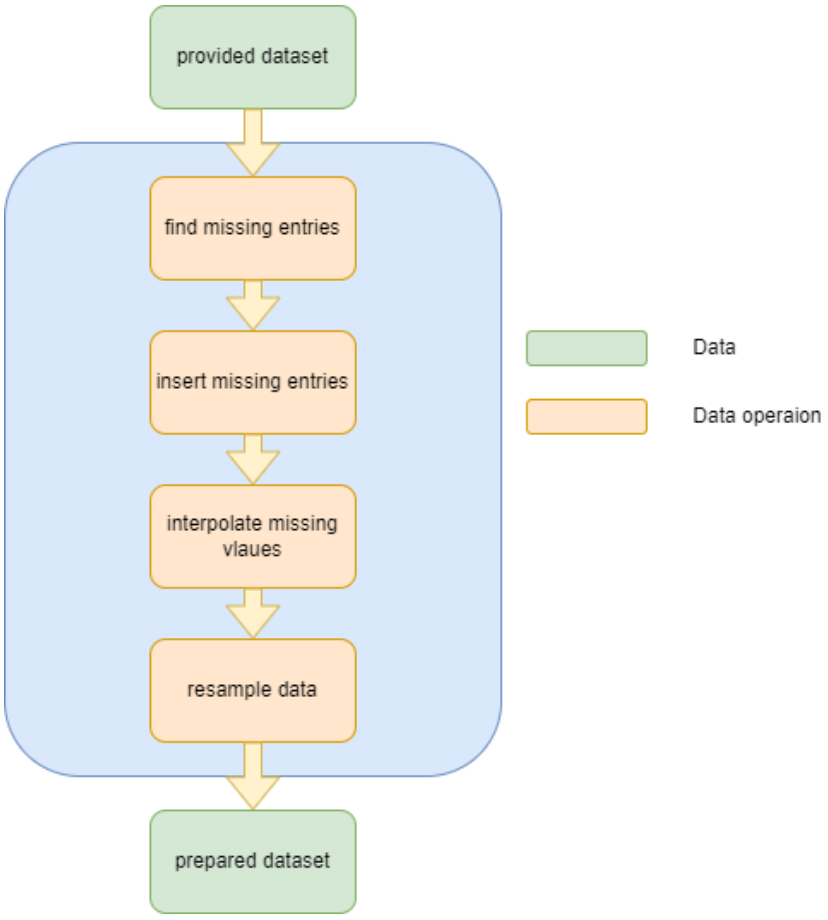

Figure 9: Data preparation steps in the preparation feature

Missing Entries

Most time series forecasting models can not handle missing entries well because they only use the data itself without the time stamps and therefor do not recognize if a entry is missing, like it is the case for the arima model from statsmodels [32] or the TBATS model [33]. In those cases a missing entry can negatively affect the accuracy of the model because the following values are moved and not on their correct position. If the data is recorded monthly with a yearly seasonality, but every year one random month is missing, the model would assume that the data is not yearly seasonal but for every 11 months. Therefor the missing entries have to be detected and inserted into the dataset. To ensure a proper behaviour of the trained forecasting model it is also important to use a fitting interpolation technique to fill in the new entries and existing entries where the value is already missing in the dataset.

Detection

Before missing entries can be detected the standard interval between the time stamps has to be determined. For this, the missing entries detection feature uses the first 10 time points. Time series datasets often have a rather accurate delta between the different data points because many times they are sensor data recorded by micro controllers. For that reason only the first 10 data points where chosen for the determination of the standard interval. Normally the most common interval is used as the standard interval for the dataset, if no single most common interval exists, the mean interval of those first 10 time stamps is used.

A possible way to implement a missing entries detection would be to create a list of the expected time stamps and compare it to the list of existing time stamps in the dataset. The disadvantage of this approach is, that the time stamps need to be accurate. Even a slight deviation will result in a falsely as missing annotated time stamp.

The approach used in the Predictive Analytics Framework is to iterate over the time points and check if the next point is within a deviation of the defined default time interval. The deviation used is +-10% to give the data points some room for errors without detecting them as missing and to not overstep and fail to detect missing entries. The missing entries are at first added without a value and later interpolated with the other time stamps that have no value in the dataset.

The implementation of the missing entries detection looks as follows:

Algorithm 1 Missing entry detection

```
1: min_interval ← default_interval * 0.9
2: max_interval ← default_interval * 1.1
3: missing_entries ← empty list of detected missing entries
4: time_stamp ← time stamps of the dataset
5: i ← 0
6: while i < lenght dataset do
7:     time_dif ← time_stamp[i + 1] − time_stamp[i]
8:     if time_dif between min_interval & max_interval then
9:         expected_entry ← time_stamp[i] + default_interval
10:        misssing_entries adds expected_entry
11:        if (time_stamp[i + 1] − expected_entry) > max_interval then add additional missing entries to missing_entries
12:        end if
13:    end if
14: end while
```

Special cases that must be considered are the two time changeovers per year. This could be done by requesting all changeovers from an online calendar and checking if the last available dates in the dataset are also in the list of time changeovers.

Insertion

After the missing data points have been detected they are added into the dataset with no value. The empty values are then interpolated. The default interpolation method used is linear, but the user can select others like quadratic, cubic or polynomial too.

Data Resampling

After the missing values have been interpolated, the dataset is resampled to reduce the amount of data points to a reasonable amount that is defined by the using developer. It is not useful to train a model with data in the resolution of seconds when multiple years are forecasted. The chosen strategy of resampling is the summation, to get the total amount of the resampled data. That means, that the individual original values are added together to form the new value, for example the values of each day are added to get the value for the complete month.

The summation strategy is chosen, because it is a simple and fast method for time series data resampling.

3.2.2. Data Understanding

It is often useful to analyse the given data before the training of potential models to narrow down the selection of models or their parameters. In some cases information like the seasonality is required to train the model properly. This is for example the case for the TBATS algorithm [33] or the auto_arima algorithm from the pmdarima package [23].

For many programming environments there are already tools available to decompose a time series into its component. For example in python the seasonal_decompose()[34] method provided by the statsmodels[35] framework can be used to decompose a time series into a trend, seasonal and resid component. It works by first estimating the trend with convolution filters, the series is then de-trended and the average of each period is used as the seasonal component [34]. This function is used in the understanding feature of the PAF to give the user the option to get an overview of the composition of the provided dataset.

This helps the user to search for seasonality and if there are uncommon outliers in the dataset, like it is the case for the water temperature dataset, which seasonal decomposed diagram can be seen in appendix A.6 in figure 37, where there is an outlier before the '2021-03' date mark.

Seasonality Extraction

Those information can then be further used to extract information like the seasonality. With the help of the fast fourier transformation the seasonal component of the time series can be decomposed into its individual frequencies [36]. Those frequencies can then be used to determine the periods contained in the dataset with the formula $frequency = \frac{1}{period}$[37]. An example how this can be implemented in python has been published on the website ataspinar.com [38]. Because the seasonal_decompose() feature estimates the trend, there is a possibility for errors, where the trend contains seasonal elements, like it can be seen in section 4.4.4 in the figures 30 and 31, where the trend clearly contains seasonal components, weekly for the small dataset and yearly for the large dataset. For this reason it is better to use the fast fourier transformation on the original dataset to extract the seasonalities. Using the seasonal

component would have had the advantage, that the seasonal frequencies would have been easier to detect, because only a few frequencies would be present.

3.3. Additional Characteristics of the Predictive Analytics Framework

The PAF has some characteristics, that have to be taken into account when using it.

3.3.1. Dataset Format

To ensure, that all features are working and all models can be trained, the PAF requires the dataset to have date time column at the first position. At the moment, only timestamps of the format 'dd.mm.yy HH:MM:SS' are accepted. Some models, like the in section 2.4.7 introduced Prophet model, require complete datasets, that contain a date time column. In addition, Features like the resampling and missing entries detection and insertion also require date times to function. For these reasons this restriction is been applied.

3.3.2. Multivariate Forecasting

The PAF contains 3 models at the start, which are compatible with multivariate predictions. These models are the LSTM, GRU and Prophet models. The drawback of using multivariate models is, that datasets of the other classes are required. To reduce the change of errors for not experienced developers, the default in the PAF is to use only univariate models by only taking the first data column into account. The developer can change this, by setting the used columns. With this, the user can tell the framework which columns should be used and in which order (for example 3,5,1) they should be used. The first data column is the one that will be predicted.

4. Time Series Prediction

The core of the Predictive Analytics Framework is the training and selection of models. Because most models have different parameters it is not only necessary to select between the different model types, but to also select for each model type the best set of parameters. To evaluate the effectiveness of the PAF different datasets representing different use cases where used to evaluate how the framework would perform in these cases.

4.1. Selected Time Series Prediction Models for the Predictive Analytics Framework

Not every available model can be integrated into the PAF at the beginning. Some do not make a lot of sense to be integrated, for some technical or other reasons make an integration not meaningful. Which models where implemented, which not and why these decisions where made will be explained in the following sections.

4.1.1. Selected Models

The models that where selected to be implemented in the initial version of the Predictive Analytics Framework (PAF) are the following:

- Autoregression (AR)
- Seasonal Autoregressive Integrated Moving Average (SARIMA)
- Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components (TBATS)
- Exponential Smoothing (ES)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Other models can be implemented via the in section 3.1.1 explained abstract class.

In the following paragraphs it will be explained, why these models where selected for the use in the PAF

Autoregression

This model is implemented in the framework because of its simplicity. It is a small and fast training model which can be enough for use cases where simple time series have to be forecasted in a small amount of time. It was included into the framework, because small models like it train fast and can perform under certain conditions still good.

Seasonal Autoregressive Integrated Moving Average

The SARIMA is integrated into the PAF since it can handle more complex time series with seasonality and trend better. Due to the more complex model and amount of parameters it takes longer to train than the AR model. This model was added to the PAF because of its better support of more complex seasonality.

Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components

TBATS is a more complex time series model, which internally trains and compares different models with different configurations, similar to the PAF itself. Like in section 2.4.5 explained, it uses internally several different components and optimizes its parameters automatically.

Exponential Smoothing

ES is again a more simple model which trains fast. The selected type of Exponential Smoothing is the Triple Exponential Smoothing because it contains support for trend and seasonality.

Long Short-Term Memory

LSTM is a RNN model that can handle more complex time series and supports multivariate datasets. In contrast to standard RNN the LSTM can bridge longer time lags. This model was added as a representative of the NN models

Gated Recurrent Unit

Equal to LSTM, GRU can handle more complex time series and multivariate datasets. Due to the simpler cell design, it has the advantage of faster training times. This model was added as a faster alternative to the LSTM model.

Prophet

Prophet is integrated into the set of default models of the PAF, because is is a fast training model, which does not require any parameters. In addition, it is possible to use it for multivariate forecasting, like LSTM and GRU.

4.1.2. Not Selected Models

The models that are not integrated into the current version of the Predictive Analytics Framework are Moving Average (MA), Autoregressive Integrated Moving Average (ARIMA), Recurrent Neural Network (RNN) and Prophet.

The following paragraphs explain the decision to not include these models into the PAF.

Moving Average

The MA model is not incorporated into the framework because AR is already integrated as a small, fast training model. Additionally it is also partially implemented through the SARIMA model when only the q parameter is set.

Autoregressive Integrated Moving Average

Similar to the MA model, the ARIMA model is already partially integrated into the framework by the SARIMA model, if the seasonal parameters P, D, Q and M are not set. To minimize the amount of different models only the SARIMA model will be integrated to decrease the amount of models that the PAF has to train.

Recurrent Neural Network

With LSTM and GRU two for time series data optimized special RNN implementations are already included into the PAF. To reduce the amount of models and because of the problem with longer lag gaps explained in section 2.4.8 this model will not be contained by the PAF.

4.2. Model Parameter Optimization

The selected models often have different parameters to tune and optimize the model to the conditions of the dataset. The PAF optimizes these parameters, so that the

using developer receives the optimized model and does not have to try to optimize the parameters themself.

4.2.1. Autoregression

The main parameter of the AR model is the 'lags' parameter. This parameter is an array of integers, which tells the model which lags should be taken into account for the prediction. The PAF uses the seasonality as a reference for the amount of lags to use. When there is no detected seasonality or it is smaller than 15 the maximum lag is set to 15 to provide a sufficient amount of options, if the seasonality is larger, its value is selected as the maximum lag.

The module than iterates from 1 to the maximum lag + 1 and uses the range from 1 to the iteration step + 1 as the value of the 'lags' parameter, as it can be seen in the pseudo code in the algorithm 2. '+ 1' is used because the range(i, j) function counts from i until j, not including the j value.

Algorithm 2 AR lag determination

```
1: function TRAIN_AR_MODEL(()train_ds, lags, seasonality)
2:     if seasonality! = 0 then
3:         seasonal ← True
4:     else
5:         seasonal ← False
6:     end if
7:     ar_model ← AutoRegression(train_ds, lags = lags, seasonal =
   seasonal, period = seasonality)
8:     ar_model.train()
9:     ...
10: end function
11:
12: seasonal_periods ← from data understanding or user
13: train_ds ← dataset drain split
14: if length(seasonal_periods) == 0 then
15:     seasonality ← 0
16: else
17:     seasonality ← minimum(seasonal_periods)
18: end if
19: if seasonality < 15 then
20:     max_lag ← 15
21: else
22:     max_lag ← int(seasonal_periods)
23: end if
24: for lag in range(1, max_lag+1) do
25:     lags ← list(range(1, lags + 1)
26:     train_ar_model(train_ds, lags, seasonality)
27: end for
```

The model additionally has a boolean 'seasonal' and a 'period' paramter. The 'seasonal' parameter is set to true and the 'period' parameter is set to the seasonality value if the data understanding feature from sector 3.2.2 or the user have detected seasonality in the dataset.

The different 'lags' combinations are trained in parallel to decrease training time.

After the training, the model with the smallest AIC value is selected as the best and returned. The AIC metric is used, because it is a good metric to compare different models of the same type and it penalizes larger models, which helps to prevent overfitting, where the model is more complex then necessary.

4.2.2. Seasonal Autoregressive Integrated Moving Average

The SARIMA model contains 7 parameters: 'p', 'd', 'q', 'P', 'D', 'Q' and 'M'. Trying every single combination for these parameters with a grid search algorithm would take a lot of time. To decrease the training time of this model, the auto_arima algorithm [39] from the pmdarima package is use to optimize the parameters of this model. This algorithm uses different tests to identify the optimal values for the 'd' and 'D' parameters and uses a step-wise approach to optimize each parameter to reduce training time [39].

The ranges for the parameters are:

- p: 0 - 5
- d: 0 - 2
- q: 0 - 5
- P: 0 - 2
- D: 0 - 1
- Q: 0 - 2

The parameter 'M' is always set to the found seasonality.

The parameter ranges where selected this way to test a certain amount of parameter combinations without the training times getting out of hands.

4.2.3. Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components

The TBATS algorithm [40] from the tbats package already optimizes the parameters internally. The only parameter that has to be provided is the 'seasonal_periods' parameter, which are the periods the data understanding algorithm from section 3.2.2 detected.

4.2.4. Exponential Smoothing

The ES model of the statsmodel.tsa.holtwinters package [41] can receive 3 parameters:

- trend: type of trend component ('add', 'mul', None)
- seasonal: type of seasonal component ('add', 'mul', None)
- seasonal_periods: the seasonality of the dataset

The parameter value 'add' stands for 'additive' and the value 'mul' for 'multiplicative. They describe how the different parts of the model (trend and seasonality) are combined, either by adding them up or by multiplying.

The different combinations of the parameters 'trend' and 'seasonal' are trained in parallel and the best model is chosen depending on the AIC value of the trained models, to reduce the chance of overfitting.

4.2.5. Long Short-Term Memory / Gated Recurrent Unit

Since both models differ only in one layer, which is either an LSTM or GRU layer inside the neural network, they share most of the internal code and use the same techniques and parameters.

A normal grid search approach is used to train all parameter combinations and find the best model. Because these models do not support metrics like AIC and BIC, the RMSE metric is used to compare the accuracy of their prediction.

The parameters for these models are:

- number of epochs: 10, 25 or 50
- batch size: 20, 30 or 50
- hidden layer size: 30, 40, 50

4.2.6. Prophet

The Prophet model analyses the dataset and detect seasonality itself [16]. It does not require any parameter tuning.

4.3. Automatic Prediction Model Selection

Even tough the information criteria like Akaike and Bayesian are popular evaluation methods for time series forecasting models with the benefit that all data are used to train the model, they are not suitable for the use in the PAF.

One of their drawbacks described in section 2.5 is that they prefer models with less parameters. This gives models which have on default more parameters a disadvantage. Additionally to that it is not always that easy to define the correct number of parameters for a model, as it is stated from the user LiKao on StackExchange [25], because some models have hidden parameters. This is for example the case for RNN and LSTM models. Additionally the user states, that other properties of such models limits the usefulness of information criteria. Because the information criteria use the likelihood function it is necessary to be able to evaluate how likely it is that the data is represented by the model. This is possible for regressive models like AR or ARIMA, but proves to be difficult for complex neuronal networks.

For the evaluation between different models only the MSE or the RMSE are suitable. The information criteria are well suited for the evaluation of the same type of models with different parameters to find out the best set of parameters for a model and can be used in the separate models for this task. For the selection of the best model between different types the MSE or RMSE can be used. Which of these two is used does not make a difference for the result because the RMSE is derived from the MSE. Because the RMSE is smaller and easier for humans to compare, it was selected as the metric that is used to compare the best models of each type with each other and to select the best one.

4.4. Test Datasets

Different types of datasets with different properties where selected to test the Predictive Analytics Framework prediction models on a large variety of use cases. The selected datasets are:

- 8 datasets from visit counter representing visitor data from public places of events
- 4 datasets from the Singapore Covid-19 development representing data of infectious diseases
- 2 datasets of the power consumption in Germany to represent strongly seasonal data (daily, weekly, yearly)
- 1 dataset of the water temperature of a well, representing typical sensor data

- 5 datasets of the Tesla stock market value, representing strongly random and hard to predict data
- 8 datasets of an ekg signal representing, representing uniform, easy predicable data

4.4.1. Visit Counter Ulm

These datasets represent a typical use case for governments and event organisers. They are available on the datahub of the city Ulm [42]. Datasets like these could be used to estimate how many visitors are expected for events like Christmas market or concerts or to evaluate the value of sales buildings depending on the amount of potential visitors.

Most of these have a gap in the data record from the 03.08.2021 to the 09.08.2021, as can be seen in appendix A.1 in the figures 10, 11, 12, 13, 15, 16 and 17. The only dataset without this gap is the Platzgasse Kohlgasse dataset shown in figure 14. The cause of this could be either a maintenance on most of the detection points or an error in the recording system that only received or recorded the information from the Platzgasse Kohlgasse detection unit.

Other general information about these datasets can be seen in the tables 3, 4, 5, 6, 8, 9 and 10

Because of this the dataset will not be used in its original resolution in the evaluation of the models. The dataset will be resampled in hourly and daily resolution to fill the missing values with 0. This way the dataset represents badly maintained datasets with missing values. The gaps are not filled in in the original resolution because of irregularities in the intervals between the individual data points timestamps.

Gloecklerstrasse Busparkplatz

min	21
max	4367
mean	197.61
size	39577
default interval	3 min

Table 3: Visit counter Gloecklerstrasse Busparkplatz dataset information

Gloecklerstrasse Hirschstrasse

min	21
max	2634
mean	335.655
size	39733
default interval	3 min

Table 4: Visit counter Gloecklerstrasse Hirschstrasse dataset information

Sedelhof Passage

min	2
max	207
mean	21.79
size	39406
default interval	3 min

Table 5: Visit counter Sedelhof Passage dataset information

Platzgasse Herrenkellergasse

min	52
max	728
mean	180.14
size	39192
default interval	3 min

Table 6: Visit counter Platzgasse Herrenkellergasse dataset information

Platzgasse Kohlgasse

min	33
max	442
mean	104.12
size	31005
default interval	3 min

Table 7: Visit counter Platzgasse Kohlgasse dataset information

Hirschstrasse Muensterplatz

min	20
max	1346
mean	275.69
size	36750
default interval	3 min

Table 8: Visit counter Hirschstrasse Muensterplatz dataset information

Hafenbad Hafengasse

min	47
max	997
mean	272.28
size	39990
default interval	3 min

Table 9: Visit counter Hafenbad Hafengasse dataset information

Hans und Sophie Scholl Platz

min	1
max	750
mean	64.93
size	39470
default interval	3 min

Table 10: Visit counter Hans und Sophie Scholl Platz dataset information

4.4.2. Covid-19 Numbers Singapore

These datasets represent the development of infectious deceases. Additionally to that, there are also two cumulative datasets, which means that the values of those datasets only increase. These cumulative datasets are largely influenced by the trend, as shown in appendix A.2 in the figures 19 and 21.

The other two other datasets shown in appendix A.2 in the figures 18 and 20 show rather flat courses except for two hills representing two Covid-19 waves Singapore experienced.

The datasets are available on the data.world data hub [43].

The following tables 11, 12, 20 and 21 show the minimum, maximum and mean values of the datasets.

Covid-19 Confirmed Cases

min	0
max	26032
mean	1374.96
size	802
default interval	1 day

Table 11: Covid-19 confirmed cases dataset information

Covid-19 Confirmed Cases cumulative

min	1
max	1109744
mean	133041.45
size	802
default interval	1 day

Table 12: Covid-19 confirmed cases cumulative dataset information

Covid-19 Deaths

min	0
max	18
mean	1.59
size	802
default interval	1 day

Table 13: Covid-19 deaths dataset information

Covid-19 Deaths cumulative

min	0
max	1276
mean	197.08
size	802
default interval	1 day

Table 14: Covid-19 deaths cumulative dataset information

4.4.3. 8-Channel EKG-Signal

This dataset was recorded and published on the kaggle data hub [44]. It contains an 8-channel ekg-signal. It was chosen to represent regular, easy to predict datasets. Due to irregularities in the heart rhythm in the later course of the dataset only the first 30000 data points are taken into the evaluation.

The following tables 15, 16, 17, 18, 19, 20, 21 and 22 contain the minimum, maximum, mean values and the size of the datasets.

The decomposition of these datasets can be seen in appendix A.3 in the figures 22, 23, 24, 25, 26, 27, 28 and 29.

min	-0.220154
max	0.596069
mean	0.009779
size	30000

Table 15: Ekg channel 1 dataset information

min	-0.230835
max	1.1153
mean	0.05253
size	30000

Table 16: Ekg channel 2 dataset information

min	-0.18396
max	0.7099
mean	0.046173
size	30000

Table 17: Ekg channel 3 dataset information

min	0.000183
max	0.001343
mean	0.000722
size	30000

Table 18: Ekg channel 4 dataset information

min	-0.48291
max	0.140015
mean	-0.002967
size	30000

Table 19: Ekg channel 5 dataset information

min	-0.005127
max	-0.003784
mean	-0.004557
size	30000

Table 20: Ekg channel 6 dataset information

min	-0.004822
max	-0.002686
mean	-0.00343
size	30000

Table 21: Ekg channel 7 dataset information

min	0.004089
max	0.007385
mean	0.005057
size	30000

Table 22: Ekg channel 8 dataset information

4.4.4. Power Consumption Germany

These datasets are available on the SMARD dataplatform of the German Federal Network Agency [45].

They contain several personalities caused by human behaviour:

- Daily caused by the sleep cycles
- Weekly caused by the work cycles
- Yearly caused by larger energy need in the winter for heating

Due to the large amount of data point, only the yearly seasonality is visible in the graphs shown in appendix A.4 in the figures 30 and 31.

The general information about these datasets can be seen in the tables 24 and 23.

The large dataset is used to compare the models capability to train with larger amounts of data. The smaller was selected to see, how the models can handle multiple seasonalities.

Power Consumption 04/2021 - 04/2022

min	0
max	20342
mean	14283.73
size	37728
default interval	15 min

Table 23: Power consumption Germany 04/21 - 04/22 dataset information

Power Consumption 01/2015 - 07/2022

Due to the size of this dataset only weekly and monthly resolutions are taken into the evaluation.

min	7752
max	20342
mean	14272.92
size	265820
default interval	15 min

Table 24: Power consumption Germany 01/15 - 07/22 dataset information

4.4.5. Tesla Inc Stock Market Value

These datasets are available at github and are provided by the user plotly [46]

These datasets represent rather random and hard to predict data. As it can be seen in appendix A.5 in the figures 32, 33, 34, 35 and 36

The tables 25, 26, 27, 28 and 29 show the general information like the minimum, maximum and mean values of the datasets.

Tesla Stock Open Value

min	142.32
max	386.69
mean	272.77
size	756
default interval	1 day

Table 25: Tesla stock open dataset information

Tesla Stock Close Value

min	143.67
max	385.0
mean	272.65
size	756
default interval	1 day

Table 26: Tesla stock close dataset information

Tesla Stock Volume Value

min	710277
max	33597290
mean	6148865
size	756
default interval	1 day

Table 27: Tesla stock volume dataset information

Tesla Stock Low Value

min	141.05
max	379.35
mean	268.03
size	756
default interval	1 day

Table 28: Tesla stock low dataset information

Tesla Stock High Value

min	154.97
max	389.61
mean	277.13
size	756
default interval	1 day

Table 29: Tesla stock high dataset information

4.4.6. Water Temperature Well Ulm

In the dataset obtained from the City Ulm data hub [47] the water temperature of a well is documented. Predicting the future water temperature can be very important to minimize the risk of infections with dangerous pathogens like the legionella, which spread faster in warmer water and can multiply to dangerous levels if the drinking water temperature lays between 20 and 55 °C [48].

The table 30 shows the general information about this dataset and the figure 37 in appendix A.6 show the components of this dataset.

min	-4.1
max	37.5
mean	11.87
size	75260
default interval	10 min

Table 30: Water temperature dataset information

4.5. Evaluation Results

The implemented models of the PAF were evaluated with all the datasets introduced in section 4.4. Only univariate datasets where used to be able to evaluate all models.

These datasets where additionally used to evaluate one additional model, which is not part of the final PAF version. This model is a manual grid search ARIMA algorithm to compare and evaluate the benefits of the chosen auto_arima algorithm form the pmdarima package [39]. Similar to the auto_arima algorithm, which trains the individual models stepwise, the grid search approach also does not use parallel computing to decrease training times. This is done, to showcase how smart algorithms can achieve similar results with less computational effort.

The datasets where evaluated in different resolutions:

Dataset	resolutions
visitors (section 4.4.1)	hourly, daily
covid (section 4.4.2)	daily (original), weekly, monthly
ekg (section 4.4.3)	original (no time information)
power consumption (section 4.4.4)	daily, weekly, monthly
telsa stock (section 4.4.5)	daily (original), weekly, monthly
water temperature (section 4.4.6)	daily, weekly, monthly

Table 31: Resolutions used for the different datasets

To reduce the size of the tables, following abbreviations are used:

Dataset	abbreviation
Visit counter Gloecklerstrasse Busparkplatz	vcGB
Visit counter Gloecklerstrasse Hirschstrasse	vcGH
Visit counter Sedelhof Passage	vcSP
Visit counter Platzgasse Herrenkeller	vcPH
Visit counter Platzgasse Kohlgasse	vcPK
Visit counter Hirschstrasse Muensterplatz	vcHM
Visit counter Hafenbad Hafengasse	vcHH
Visit counter Hans und Sophie Scholl Platz	vcHS
Covid-19 cases	covC
Covid-19 cases cumulative	covCC
Covid-19 deaths	covD
Covid-19 deaths cumulative	covDC
Ekg channel 1	egk1
Ekg channel 2	egk2
Ekg channel 3	egk3
Ekg channel 4	egk4
Ekg channel 5	egk5
Ekg channel 6	egk6
Ekg channel 7	egk7
Ekg channel 8	egk8
power consumption 01/15 - 07/22	powerL
power consumption 04/21 - 04/22	powerS
Tesla stock close	teslaC
Tesla stock high	teslaH
Tesla stock low	teslaL
Tesla stock open	teslaO
Tesla stock volume	teslaV
water temperature well	water

Table 32: Dataset abbreviations used for compact representation

The letter after the dataset name in the 'Res.'(Resolution) column refers to the sampling rate that was used:

- H: Hourly
- D: Daily
- W: Weekly
- M: Monthly

The ekg datasets where not resampled because they were recorded with a 250 Hz sampling rate. In addition, they were not used with the Prophet model because of the lack of time data.

The parameters that where used by the models are shown in appendix C.2.

4.5.1. Problems During Evaluation and Testing

During the evaluation process some problems arise.

ValueError at auto_arima Prediction

Due to a known bug int the pmdarima framework it was not possible to create an evaluation for the large power consumption dataset from section 31 with the auto_arima algorithms in daily resolution. This error causes the prediction to create an ValueError when the algorithm presumably is not able to find a good solution, accordingly to the github user Shuvo-saha on a thread regarding this problem on a thread on github [49].

Unknown Error during GRU and LSTM Training

During the training of the GRU and LSTM models unknown internal errors occur during the model training for the Tesla volume dataset introduced in section 4.4.5 in monthly resolution. This error occurs on all parameter combinations for these models. Because of this error, no data could be gathered for this dataset in the monthly resolution for the GRU and LSTM models.

Excessive Training Times

The self implemented grid search algorithm for SARIMA was not evaluated with the visit counter datasets from section 4.4.1 with the hourly sampling size because of excessive training times. The individual datasets would have taken over 12 hours to train, resulting in at least over 4 days of nonstop training only for those 8 datasets. This is in times of rising energy prices where electricity prices in Germany rise by 35 to 40% [50] not sustainable. Because of this, the datasets where only evaluated with the daily sampling size.

In the current implementation the individual models of the grid search approach are trained after each other. Conducting the training in parallel could shorten the training time significantly. This is not done, because a comparison between a simple

single threaded grid search approach and a smart optimization algorithm, which also only trains one model at a time is the target of this trial.

4.5.2. Evaluation Conclusion

The following tables 33, 34, 35, 36 and 37 show, which model was the best considering the evaluation metrics RMSE/MSE, AIC and BIC. All three types of metrics are used to compare the result of each and to show differences between their outcome.

Dataset/ Model	Res.	MSE / RMSE	AIC	BIC
vcGB	H	GRU	ES	ES
	D	GRU	AR	AR
vcGH	H	GRU	ES	ES
	D	GRU	AR	AR
vcHH	H	GRU	ES	ES
	D	GRU	SARIMA grid	SARIMA grid
vcHS	H	GRU	ES	ES
	D	AR	AR	AR
vcPH	H	GRU	ES	ES
	D	GRU	SARIMA grid	SARIMA grid
vcHM	H	GRU	ES	ES
	D	GRU	AR	AR
vcPK	H	GRU	ES	ES
	D	ES	SARIMA grid	SARIMA grid
vcSP	H	GRU	ES	ES
	D	GRU	SARIMA grid	SARIMA grid

Table 33: Best models for the visit counter datasets 4.4.1

Dataset/ Model	Res.	MSE / RMSE	AIC	BIC
covC	D	GRU	SARIMA grid	SARIMA grid
	W	LSTM	ES	ES
	M	LSTM	SARIMA grid	SARIMA grid
covCC	D	LSTM	SARIMA grid	SARIMA grid
	W	LSTM	SARIMA grid	SARIMA grid
	M	GRU	AR	ES
covD	D	GRU	ES	ES
	W	GRU	ES	ES
	M	GRU	SARIMA grid	SARIMA grid
covCD	D	LSTM	ES	ES
	W	LSTM	AR	ES
	M	GRU	SARIMA grid	SARIMA grid

Table 34: Best models for the Covid datasets 4.4.2

Dataset/ Model	MSE / RMSE	AIC	BIC
ekg1	GRU / LSTM	SARIMA grid	SARIMA grid
ekg2	GRU / LSTM	ES	ES
ekg3	GRU / LSTM	ES	ES
ekg4	GRU	ES	ES
ekg5	GRU	ES	ES
ekg6	GRU	ES	ES
ekg7	GRU	ES	ES
ekg8	LSTM	ES	ES

Table 35: Best models for the ekg datasets 4.4.3

Dataset/ Model	Res.	MSE / RMSE	AIC	BIC
powerL	D	GRU	ES	ES
	W	TBATS	ES	ES
	M	TBATS	AR	AR
powerS	D	AR	SARIMA grid	SARIMA grid
	W	GRU	AR	AR
	M	GRU	AR	AR
water	D	GRU	SARIMA grid	SARIMA grid
	W	GRU	SARIMA grid	SARIMA grid
	M	GRU	AR	AR

Table 36: Best models for the power consumption 4.4.4 and water temperature 4.4.6

Dataset/ Model	Res.	MSE / RMSE	AIC	BIC
teslaC	D	SARIMA	SARIMA grid	SARIMA grid
	W	TBATS	SARIMA grid	SARIMA grid
	M	GRU	AR	AR
teslaH	D	AR	SARIMA grid	SARIMA grid
	W	TBATS	ES	AR
	M	GRU	AR	AR
teslaL	D	AR	SARIMA grid	SARIMA grid
	W	LSTM	SARIMA grid	SARIMA grid
	M	GRU	SARIMA grid	SARIMA grid
teslaO	D	AR	SARIMA grid	SARIMA grid
	W	GRU	SARIMA grid	SARIMA grid
	M	GRU	AR	AR
teslaV	D	LSTM	SARIMA grid	SARIMA grid
	W	LSTM	AR	AR
	M	AR	SARIMA grid	SARIMA grid

Table 37: Best models for the Tesla stock datasets 4.4.5

The most often as best model selected model type when using the MSE or RMSE metric is the GRU model which was 38 time the best, followed by the LSTM model with 13 occurrences as the optimal model. Due to a lack of support for the Information Criterion metrics these models and the Prophet model are not available for the selection under the AIC and BIC metrics because of the in section 4.3 mentioned

problem. For these metrics the SARIMA grid search, AR and ES where the best models.

Some datasets types had a tendency to work on some models better with one of the metrics. This is for example the case for the GRU model and the visitor datasets when using the MSE or RMSE metric, as shown in table 34. Similarly table 35 shows, that the ES model is the best model for the ekg datasets using the AIC or BIC metric with one exception for the ekg1 dataset, where the SARIMA grid search model stood out.

The LSTM and GRU models do not differ by much. The GRU model tends to train a little bit faster and to be a bit more precise. If only one of those models could be implemented, the GRU model should be it. But because of the usually low cost of these models, with the exception of the ekg datasets training times shown in table 60, there is no great disadvantage and it is probably more beneficial to have both.

The Prophet model did not stand out much, it is never the best and is most of the time in the middle of the models. It trains fast, as it can be seen in the tables 58, 59, 60, 61 and 62 in section C.1.5, but not as fast as the AR and ES models. Additionally, its RMSE metric often lays in the middle between the other models.

The models AR and ES are the fastest training models. They even are sometimes the best model, like it is the case for the small power dataset in daily resolution in table 36. Because of the low cost of these models and the nonetheless good performance, these models belong in the repertoire of the PAF.

Following additional observations have been made:

Akaike information criterion vs Bayesian information criterion

Even thought the values of the two metric differ, they brought the same end result for almost all of the datasets. The only three different results are for the Tesla high dataset in weekly resolution in table 37 and in table 34 the datasets Covid-19 cumulative cases in monthly and the cumulative deaths in weekly resolution. Each time they differed in the selection between AR and ES, which is could either be caused by the different weighting of the number of parameters explained in section 2.5.4 or different amounts of parameters.

Seasonal Autoregressive Integrated Moving Average from auto_arima vs. Seasonal Autoregressive Integrated Moving Average grid search

Using the AIC and BIC metrics, the grid search approach of SARIMA usually outperforms the model from the auto_arima [39] algorithm, as it can be seen in the tables 39, 40, 41, 42, 38, 44, 45, 46, 47 and 43. When using the MSE or RMSE metric, the roles are swapped and the auto_arima algorithm outperforms the grid search approach on most occasions. Even though the documentation of the auto_arima algorithm [39] states, that the AIC metric is used to optimize the model, these results could indicate that other metrics are also used for the optimization. In addition to the worse RMSE values, the grid search approach also took significantly longer, as visible in the tables 58, 59, 60, 61 and 62 in section C.1.5. While the auto_arima algorithm most of the time only took minutes, the grid search approach took hours several times for the training. This is an additional benefit of the auto_arima algorithm from pmdarima [39] besides the better RMSE values.

5. Conclusion

Not every developer has experience in the field of time series forecasting. To help these developers, a framework has been developed which trains different forecasting models and selects the one with the lowest Root Mean Square Error value. They just need to provide a dataset to train the model and how many steps they want to forecast.

The framework can also help developer with more experience in the time series forecasting area, by comparing different models and finding optimal parameters for these models. This saves time, because they do not have to test each model themself and try to find fitting parameters for these models. In addition it also contains methods to prepare time series data, like resampling, detection of missing entries and seasonality.

The framework is implemented as a server, which can be run separately or integrated into another project as a microservice. The interface to interact with the framework follows the REST design principles, which allows the use of third party software like the Postman REST client [31] to interact with the PAF.

To take some work from the using developer, the framework contains features to prepare datasets. This includes the detection and insertion of missing dataset entries, interpolation of missing values and resampling of the dataset to a reasonable timestamp resolution. Additionally, features to decompose a time series into the trend, seasonal and error component and to calculate the seasonal periods of a time series is included.

The different models where evaluated against each other with different univariate time series dataset to get an overview how the models perform compared to each other. In addition, a manual single threaded grid search SARIMA algorithm was also included into the set of evaluated models, to get a comparison between the training time and accuracy of this approach and an automatically parameter optimizing algorithm in form of the auto_arima algorithm from the pmdarima package [39].

During the evaluation of the models it was observed, that complex Neural Network (NN) models are often the best model when considering the RMSE metric. On the other hand, for some datasets simple and fast training models like AR and ES beat those larger and longer training NN models like LSTM and GRU. In addition it was seen, that automatic parameter optimizing algorithms like auto_arima [39] can

not only train the model faster, but can also achieve better results considering the RMSE evaluation metric.

6. References

Books and Journals

[5] Konrad Gos and Wojciech Zabierowski. "The Comparison of Microservice and Monolithic Architecture". In: *2020 IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)*. IEEE, 2020, pp. 150–153. ISBN: 978-1-7281-7179-1. DOI: 10.1109/MEMSTECH49584.2020.9109514.

[8] Timo Bayer and Philipp Ruf. "Technologien und Muster zur Entwicklung von Nanoservice-Architekturen". In: *Informatik Journal 2017/18*. Vol. 7. 2017, pp. 11–23. ISBN: 978-3-00-058391-9. URL: https://www.researchgate.net/profile/Timo-Bayer/publication/323839703_Technologien_und_Muster_zur_Entwicklung_von_Nanoservice-Architekturen/links/5aaed1b20f7e9b4897c037e9/Technologien-und-Muster-zur-Entwicklung-von-Nanoservice-Architekturen.pdf.

[21] Allan H. Murphy. "Skill Scores Based on the Mean Square Error and Their Relationships to the Correlation Coefficient". In: *Monthly Weather Review* 116.12 (1988), pp. 2417–2424. ISSN: 0027-0644. DOI: 10.1175/1520-0493(1988)116{\textless}2417:SSBOTM{\textgreater}2.0.CO;2.

[22] T. Chai and R. R. Draxler. *Root mean square error (RMSE) or mean absolute error (MAE)?* 2014. DOI: 10.5194/gmdd-7-1525-2014.

[24] Bozdogan. "Akaike's Information Criterion and Recent Developments in Information Complexity". In: *Journal of mathematical psychology* 44.1 (2000), pp. 62–91. ISSN: 0022-2496. DOI: 10.1006/jmps.1999.1277.

[28] Jiangang Hao and Tin Kam Ho. "Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language". In: *Journal of Educational and Behavioral Statistics* 44.3 (2019), pp. 348–361. ISSN: 1076-9986. DOI: 10.3102/1076998619832248.

[36] James W. Cooley, Peter A. W. Lewis, and Peter D. Welch. "The Fast Fourier Transform and Its Applications". In: *IEEE Transactions on Education* 12.1 (1969), pp. 27–34. ISSN: 0018-9359. DOI: 10.1109/TE.1969.4320436.

Internetsources

[1] Benjamin Krause. "Was ist REST? Einsatz in der Industrie". In: *OPC-Router.de* (4.11.2021). URL: https://www.opc-router.de/was-ist-rest/.

[2] *TCP - Grundlagen*. 5.09.2022. URL: https://www.kunbus.de/tcp-grundlagen.

[3] "Application Programming Interface (API)". In: (19.08.2020). URL: https://www.ibm.com/cloud/learn/api.

[4] CompTIA. *Using AI in Business: Examples of Artificial Intelligence Application in Business*. 25.08.2022. URL: https://connect.comptia.org/blog/using-ai-in-business.

[6] Sudip Sengupta. *Microservices vs Nanoservices: Weighing Framework Options*. Ed. by BMC Blogs. 2021. URL: https://www.bmc.com/blogs/microservice-vs-nanoservice/.

[7] Joey Clover. *Microservices are hard — an invaluable guide to microservices. — HackerNoon*. 15.03.2022. URL: https://hackernoon.com/microservices-are-hard-an-invaluable-guide-to-microservices-2d06bd7bcf5d.

[9] *Websocket API vs REST API - What's the difference?* 5.09.2022. URL: https://www.wallarm.com/what/websocket-vs-rest-api.

[10] Marco Peixeiro. "The Complete Guide to Time Series Analysis and Forecasting". In: *Towards Data Science* (7.08.2019). URL: https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775.

[11] Prof. Dr. Volker Herbort. *Advanced Machine Learning - 7. Time Series Analytics*.

[12] *8.3 Autoregressive models — Forecasting: Principles and Practice (2nd ed)*. 26.04.2022. URL: https://otexts.com/fpp2/AR.html.

[13] *8.4 Moving average models — Forecasting: Principles and Practice (2nd ed)*. 26.04.2022. URL: https://otexts.com/fpp2/MA.html.

[14] Nadeem. "Time Series Forecasting using TBATS Model - Analytics Vidhya - Medium". In: *Analytics Vidhya* (21.11.2021). URL: https://medium.com/analytics-vidhya/time-series-forecasting-using-tbats-model-ce8c429442a9.

[15] Jason Brownlee. "A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python". In: *Machine Learning Mastery* (19.08.2018). URL: https://machinelearningmastery.com/exponential-smoothing-for-time-series-forecasting-in-python/.

[16] Meta Research. *Prophet: forecasting at scale - Meta Research — Meta Research*. 3.09.2022. URL: https://research.facebook.com/blog/2017/02/prophet-forecasting-at-scale/.

[17] Soubhik Khankary. "Multivariate Time Series Forecasting using FBProphet". In: *MLearning.ai* (30.01.2022). URL: https://medium.com/mlearning-ai/multivariate-time-series-forecasting-using-fbprophet-66147f049e66.

[18] Prophet. *Quick Start*. 2022. URL: https://facebook.github.io/prophet/docs/quick_start.html.

[19] Prof. Dr. Volker Herbort. *Advanced Machine Learning - 8. Time Series Analytics*.

[20] *LSTM Networks - EXPLAINED! - YouTube*. 3.09.2022. URL: https://www.youtube.com/watch?v=QciIcRxJvsM&ab_channel=CodeEmporium.

[23] *Autoregressions — statsmodels*. 29.06.2022. URL: https://www.statsmodels.org/devel/examples/notebooks/generated/autoregressions.html.

[25] https://stats.stackexchange.com/users/7194/likao. *Why isn't Akaike information criterion used more in machine learning?* 2019. URL: https://stats.stackexchange.com/q/325250.

[26] Analyttica Datalab. *What is Bayesian Information Criterion (BIC)?* Ed. by Medium. 2019. URL: https://medium.com/@analyttica/what-is-bayesian-information-criterion-bic-b3396a894be6.

[27] Brian Eastwood. *The 10 Most Popular Programming Languages to Learn in 2021*. 2020. URL: https://www.northeastern.edu/graduate/blog/most-popular-programming-languages/.

[29] Python Software Foundation. “Jython”. In: URL: https://www.jython.org/index.

[30] *PyTorch*. 12.04.2022. URL: https://pytorch.org/get-started/locally/.

[31] *REST Client — Postman API Platform [Free Download]*. 7.09.2022. URL: https://www.postman.com/product/rest-client/.

[32] *statsmodels.tsa.arima.model.ARIMA — statsmodels*. 22.07.2022. URL: https://www.statsmodels.org/devel/generated/statsmodels.tsa.arima.model.ARIMA.html.

[33] PyPI. *tbats*. 27.07.2022. URL: https://pypi.org/project/tbats/.

[34] *statsmodels.tsa.seasonal.seasonal_decompose — statsmodels*. 12.06.2022. URL: https://www.statsmodels.org/dev/generated/statsmodels.tsa.seasonal.seasonal_decompose.html.

[35] *Introduction — statsmodels*. 12.06.2022. URL: https://www.statsmodels.org/dev/index.html.

[37] *Frequency Formula - What is Frequency Formula? Examples*. 31.08.2022. URL: https://www.cuemath.com/frequency-formula/.

[38] ML Fundamentals. *Time-Series forecasting with Stochastic Signal Analysis techniques*. 2020. URL: https://ataspinar.com/2020/12/22/time-series-forecasting-with-stochastic-signal-analysis-techniques/.

[39] *pmdarima.arima.auto_arima — pmdarima 1.8.5 documentation*. 15.06.2022. URL: https://alkaline-ml.com/pmdarima/modules/generated/pmdarima.arima.auto_arima.html.

[40] *TBATS — sktime documentation*. 24.08.2022. URL: https://www.sktime.org/en/stable/api_reference/auto_generated/sktime.forecasting.tbats.TBATS.html.

[41] *statsmodels.tsa.holtwinters.ExponentialSmoothing — statsmodels*. 27.08.2022. URL: https://www.statsmodels.org/dev/generated/statsmodels.tsa.holtwinters.ExponentialSmoothing.html.

[42] *Besuchertrend Ulmer Innenstadt - CKAN*. 26.08.2022. URL: https://datenhub.ulm.de/ckan/dataset/besuchertrend-ulmer-innenstadt.

[43] Hui Xiang Chua. *COVID-19 Singapore - dataset by hxchua*. 2020. URL: https://data.world/hxchua/covid-19-singapore.

[44] Nmack41. *EKG / ECG 8-Channel*. 31.01.2018. URL: https://www.kaggle.com/datasets/nmack41/ekg-ecg-8channel.

[45] Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen. *SMARD — Marktdaten*. 2022. URL: https://www.smard.de/home/downloadcenter/download-marktdaten#!?downloadAttributes=%7B%22selectedCategory%22:false,%22selectedSubCategory%22:false,%22selectedRegion%22:false,%22from%22:1647644400000,%22to%22:1648591199999,%22selectedFileType%22:false%7D.

[46] GitHub. *datasets/tesla-stock-price.csv at master · plotly/datasets*. 26.08.2022.

[47] *Wassertemperatur - CKAN*. 26.08.2022. URL: https://datenhub.ulm.de/ckan/dataset/lorapark_wassertemperatur.

[48] Umweltbundesamt. "Stellungnahme des UBA - Energiesparen bei der Warmwasserbereitung - Vereinbarkeit von Energieeinsparung und Hygieneanforderungen an Trinkwasser". In: (). URL: https://www.umweltbundesamt.de/sites/default/files/medien/419/dokumente/warmwasserbereitung_energiesparen_stellungnahme_uba.pdf.

[49] GitHub. *Error: Input contains NaN, infinity or a value too large for dtype('float64'): pmdarima.predict() · Issue #404 · alkaline-ml/pmdarima*. 1.09.2022. URL: https://github.com/alkaline-ml/pmdarima/issues/404.

[50] tagesschau. "Kosten für Gas, Strom und Benzin: So hoch wird die Energie-Rechnung". In: *tagesschau.de* (5.04.2022). URL: https://www.tagesschau.de/wirtschaft/verbraucher/steigende-energiekosten-101.html.

A. Dataset Decomposition

These are the diagrams of the decomposed datasets.

The diagrams of the decomposed datasets are structured as follows:

- Original dataset: The original data of the dataset
- Trend component: The trend the dataset is following
- Seasonal component: The filtered out seasonality of the dataset
- Resid component: The random error after trend and seasonality is removed

A.1. Visit Counter Ulm

Figure 10: Seasonal decompose of the visit counter for the Gloecklerstrasse Buspark-platz

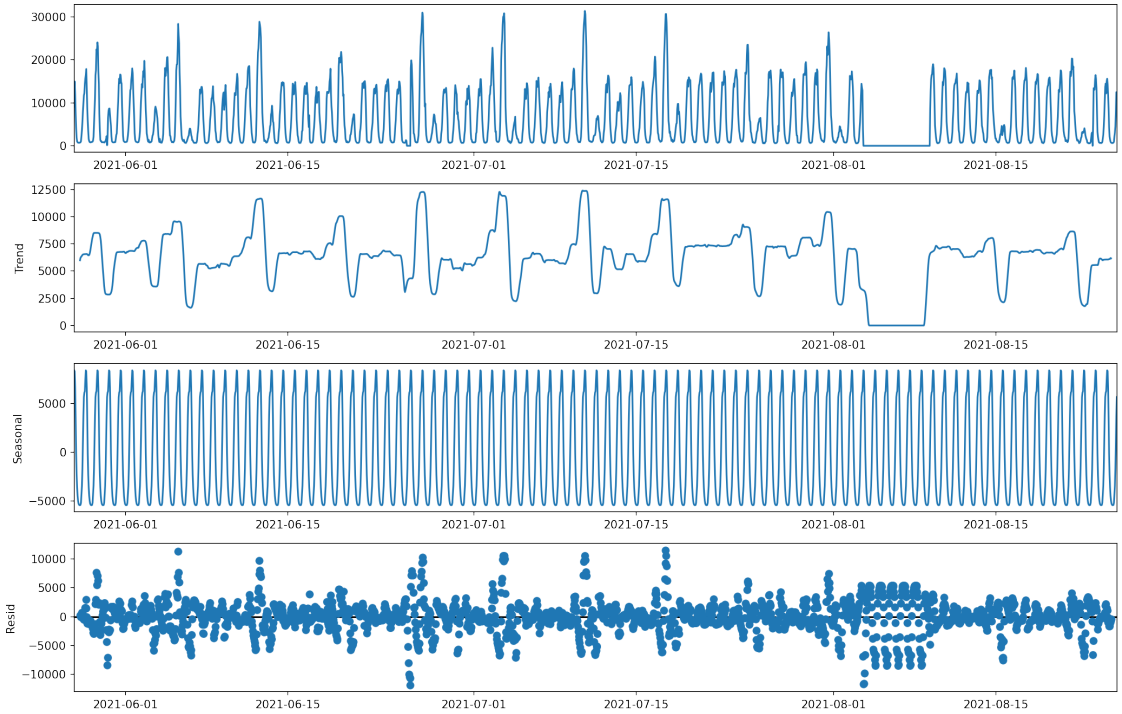

Figure 11: Seasonal decompose of the visit counter for the Gloecklerstrasse Hirschstrasse

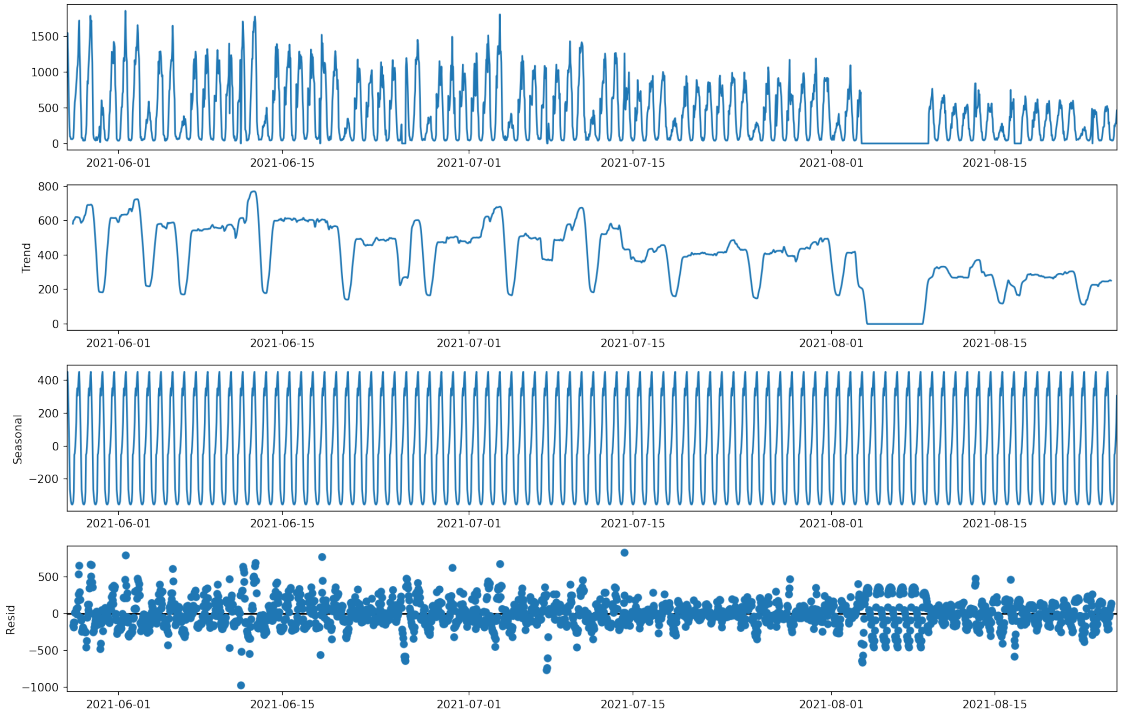

Figure 12: Seasonal decompose of the visit counter for the Sedelhof Passage

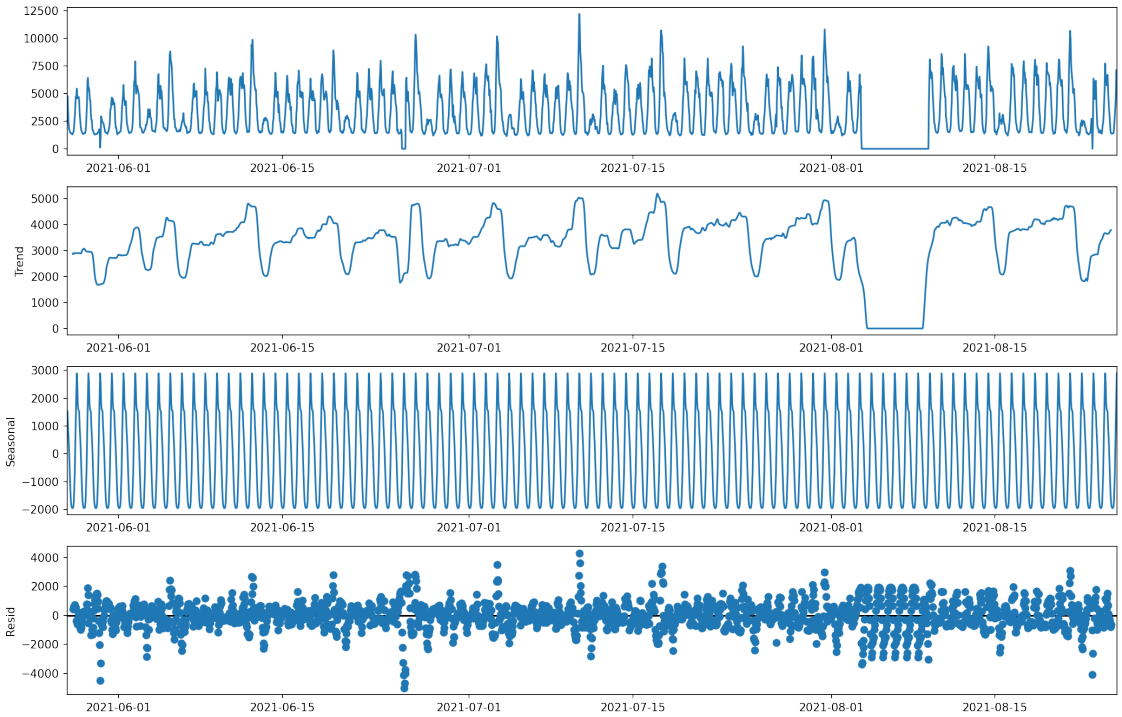

Figure 13: Seasonal decompose of the visit counter for the Platzgasse Herrenkellergasse

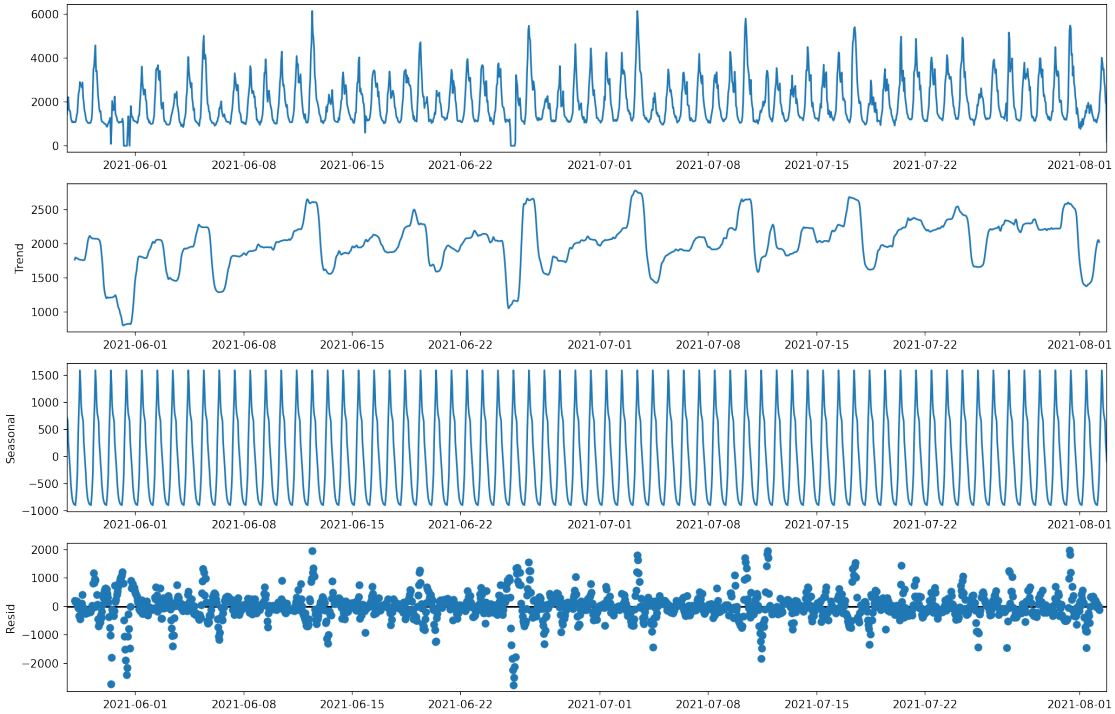

Figure 14: Seasonal decompose of the visit counter for the Platzgasse Kohlgasse

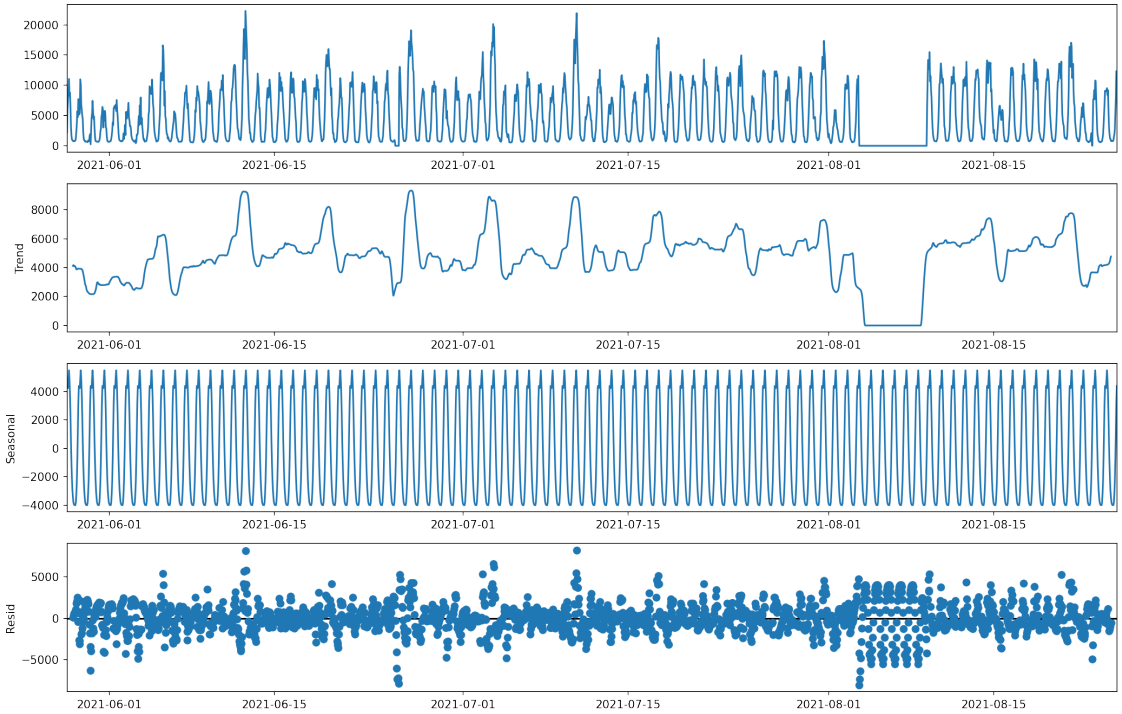

Figure 15: Seasonal decompose of the visit counter for the Hirschstrasse Muensterplatz

Figure 16: Seasonal decompose of the visit counter for the Hafenbad Hafengasse

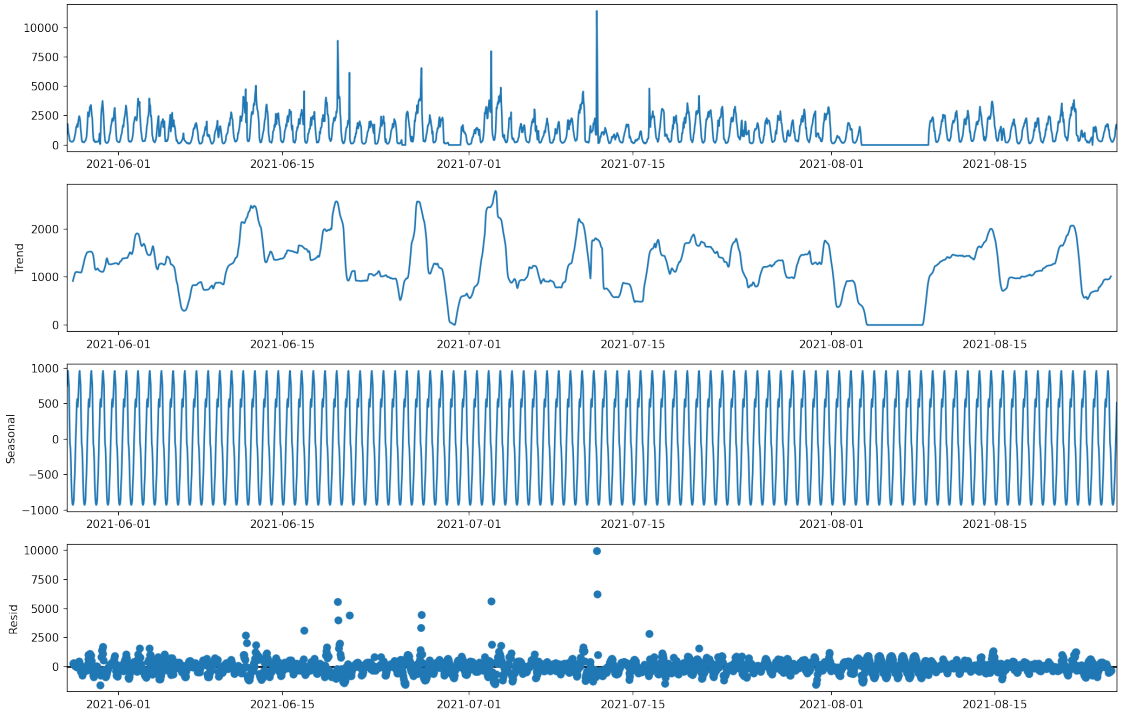

Figure 17: Seasonal decompose of the visit counter for the Hanz und Sophie Scholl Platz

A.2. Covid-19 Numbers Singapore

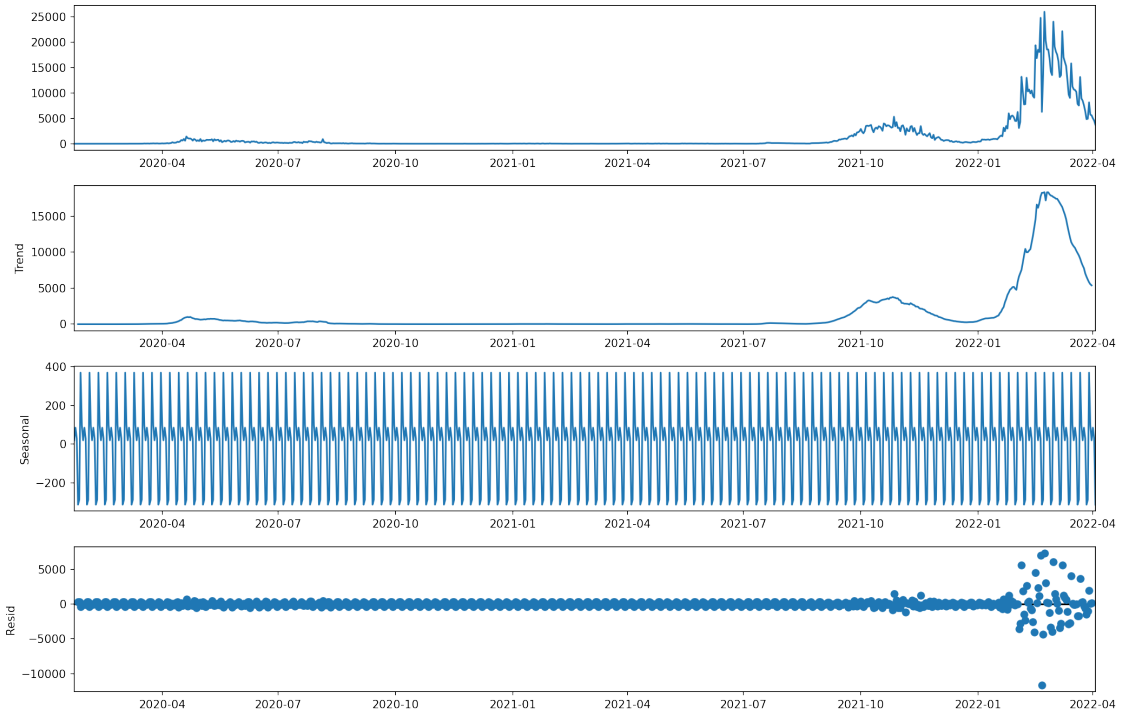

Figure 18: Seasonal decompose of the number of Covid-19 infection in Singapore

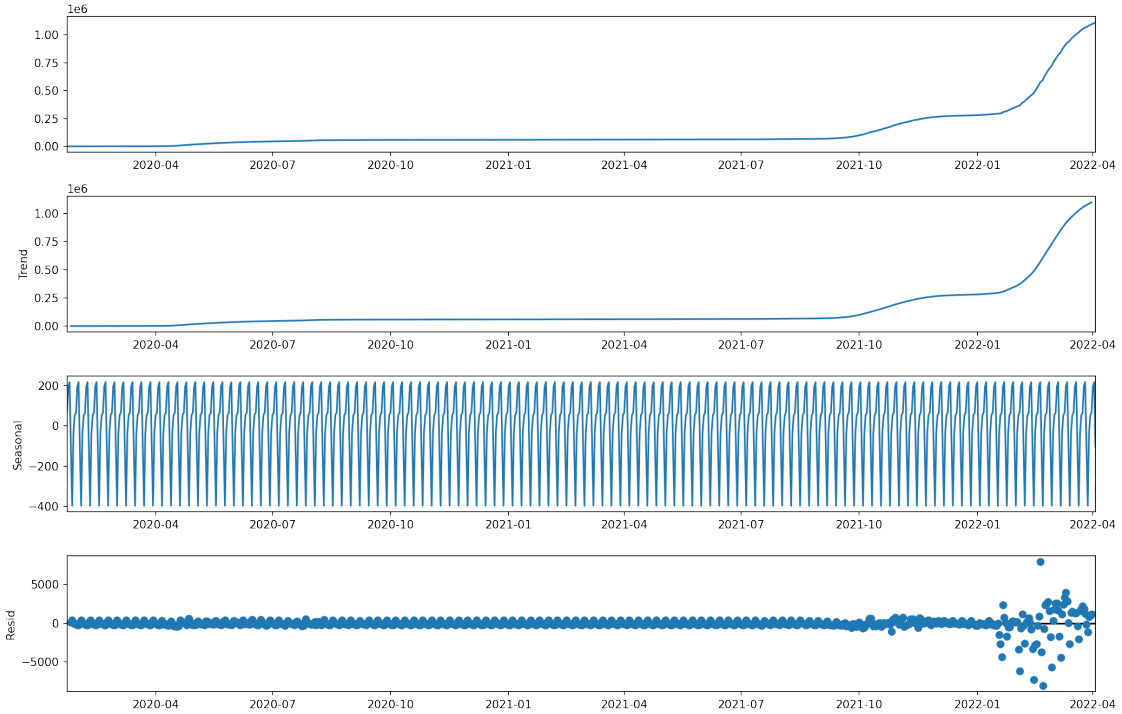

Figure 19: Seasonal decompose of the cumulative number of Covid-19 infection in Singapore

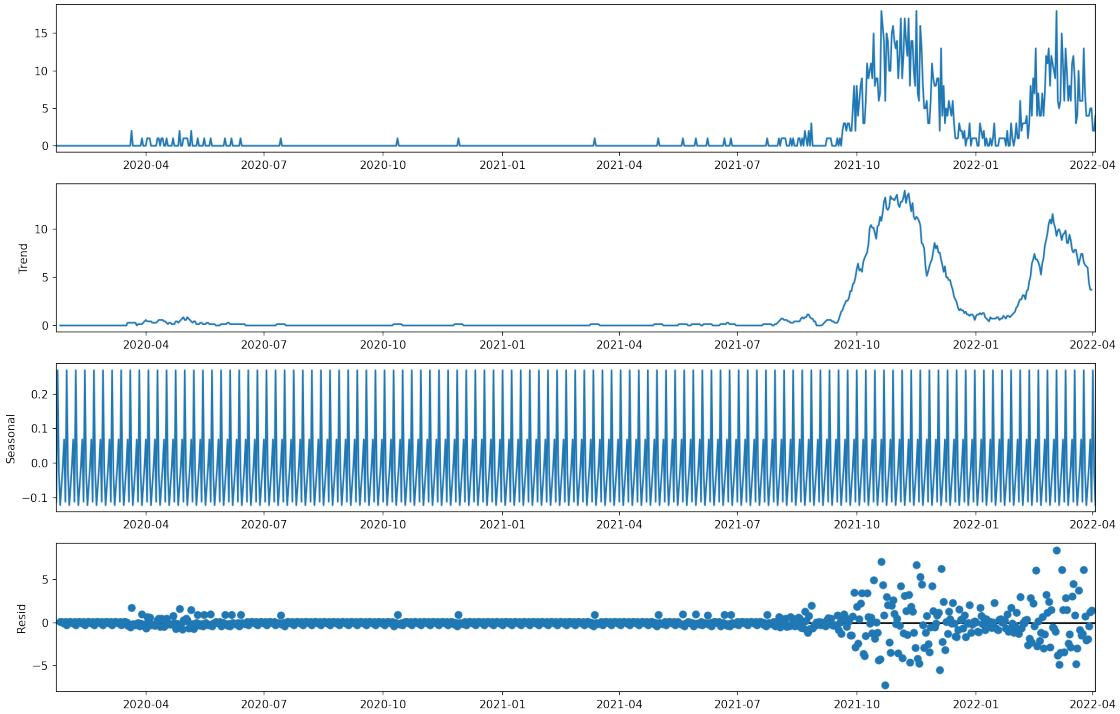

Figure 20: Seasonal decompose of the number of Covid-19 deaths in Singapore

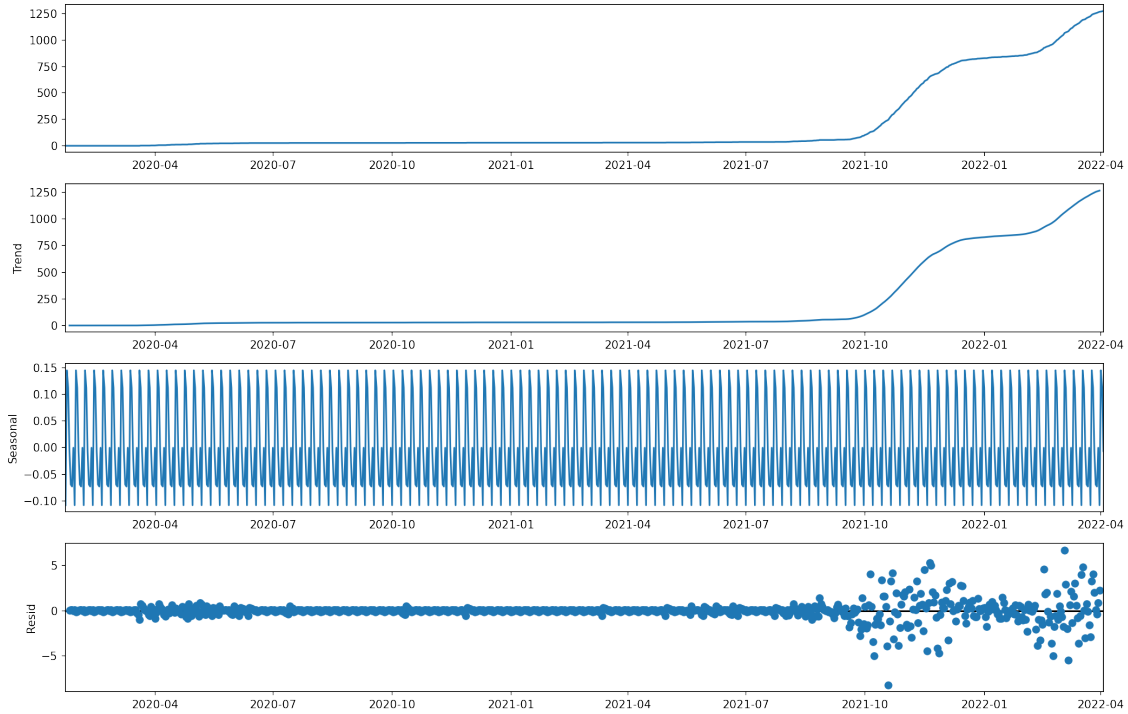

Figure 21: Seasonal decompose of the cumulative number of Covid-19 deaths in Singapore

A.3. 8-Channel EKG-Signal

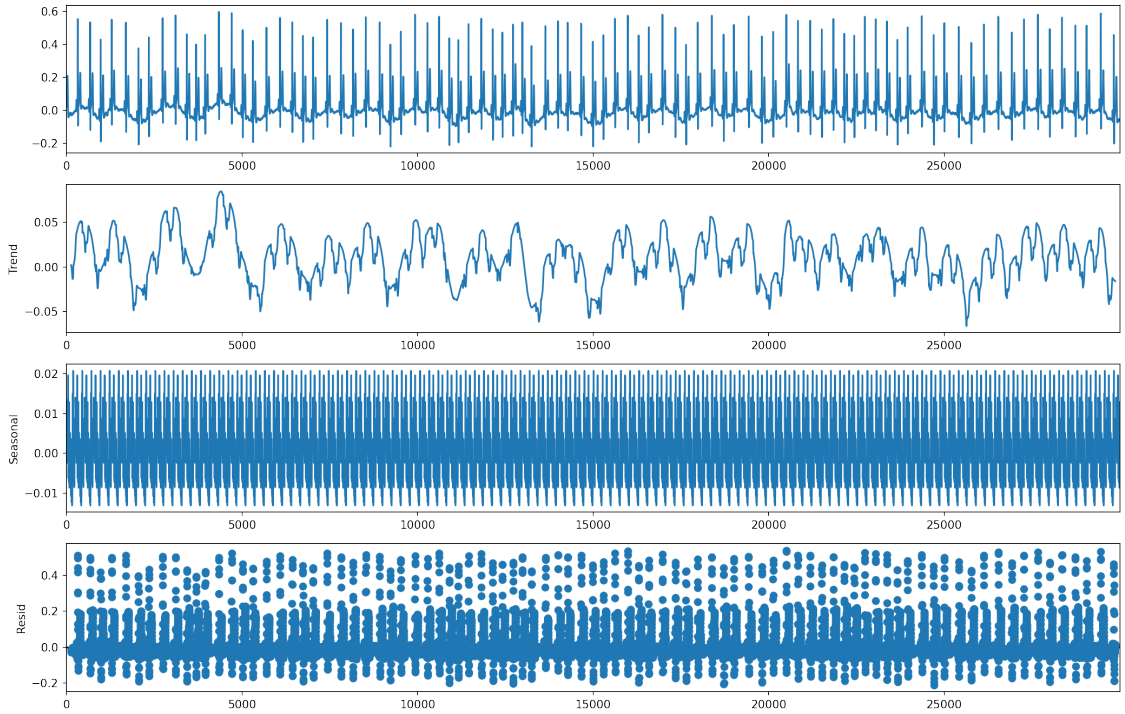

Figure 22: Seasonal decompose of Channel 1 of the ekg-signal

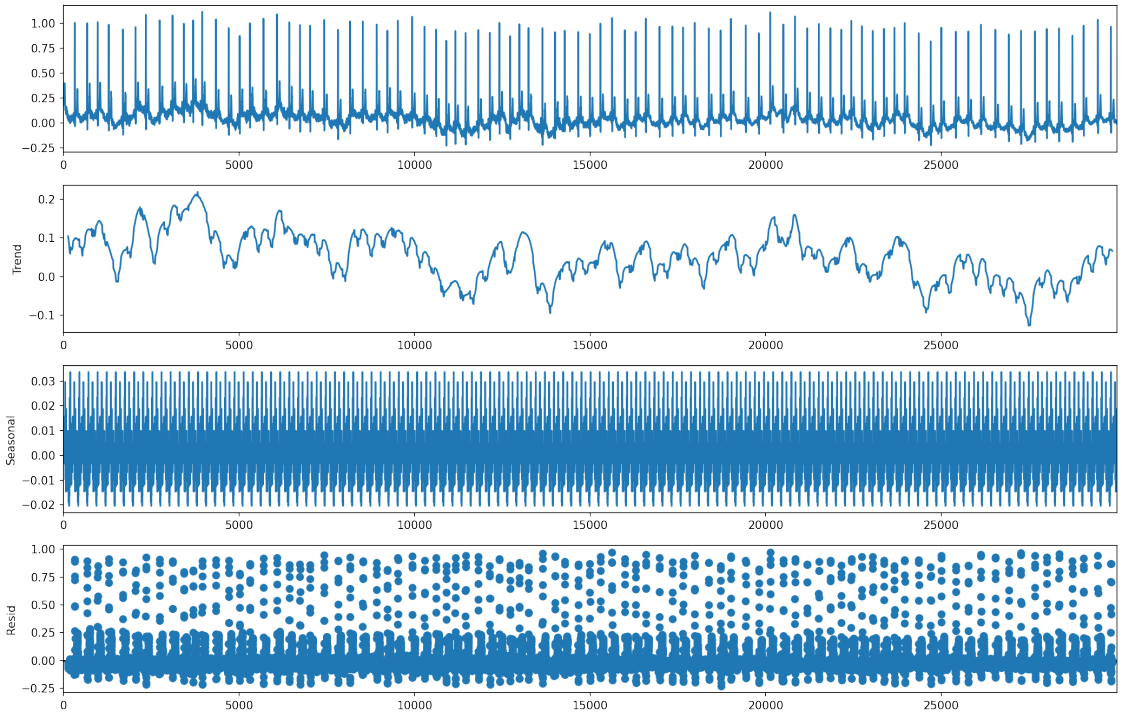

Figure 23: Seasonal decompose of Channel 2 of the ekg-signal

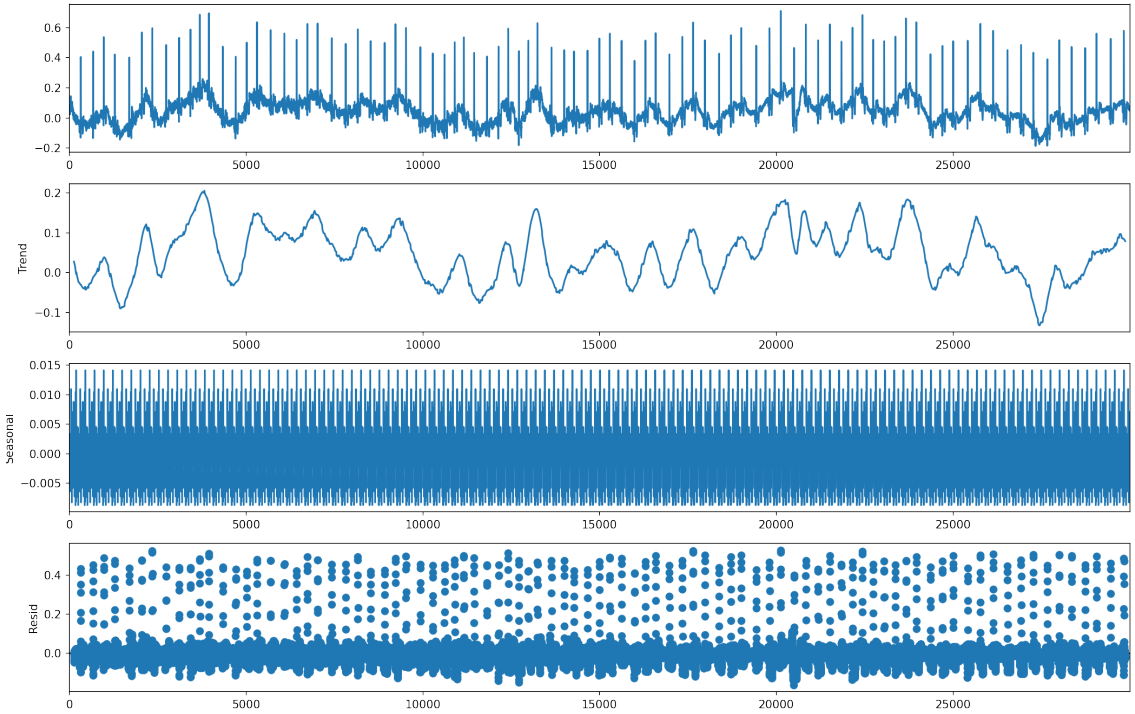

Figure 24: Seasonal decompose of Channel 3 of the ekg-signal

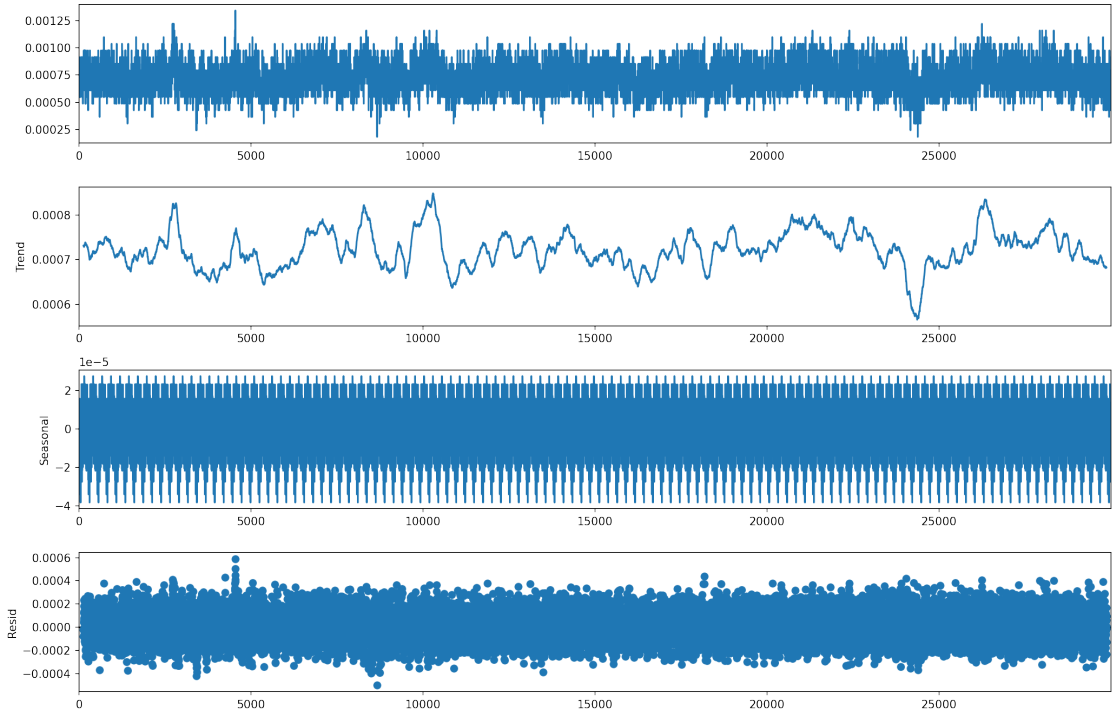

Figure 25: Seasonal decompose of Channel 4 of the ekg-signal

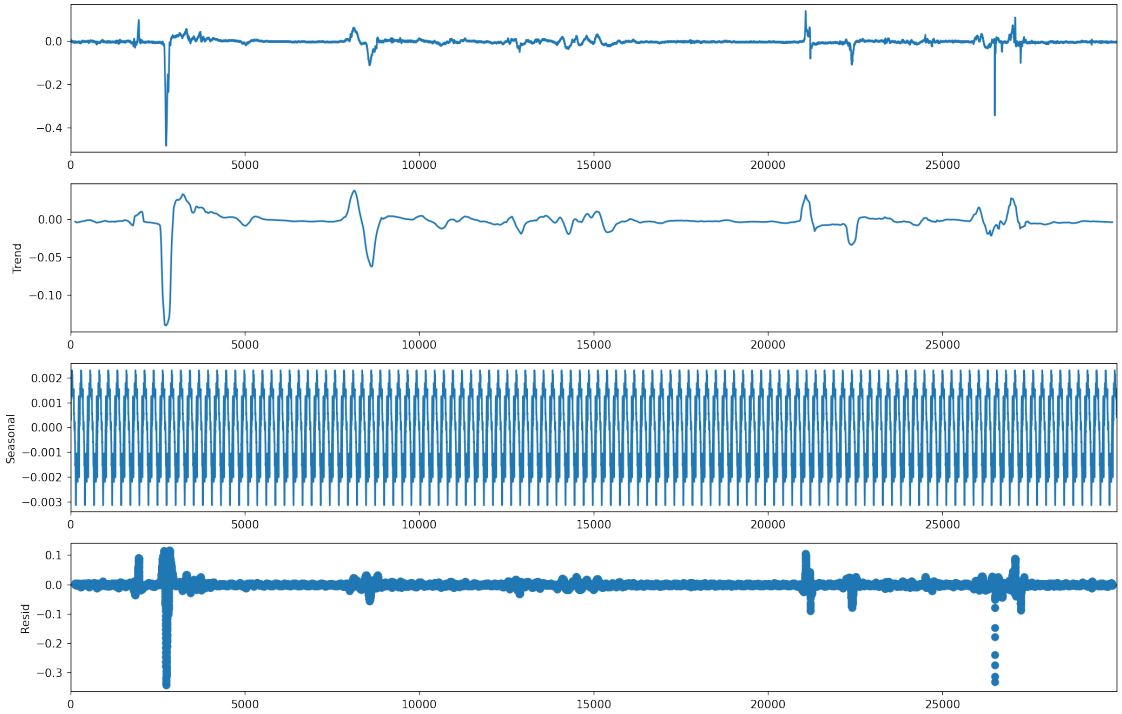

Figure 26: Seasonal decompose of Channel 5 of the ekg-signal

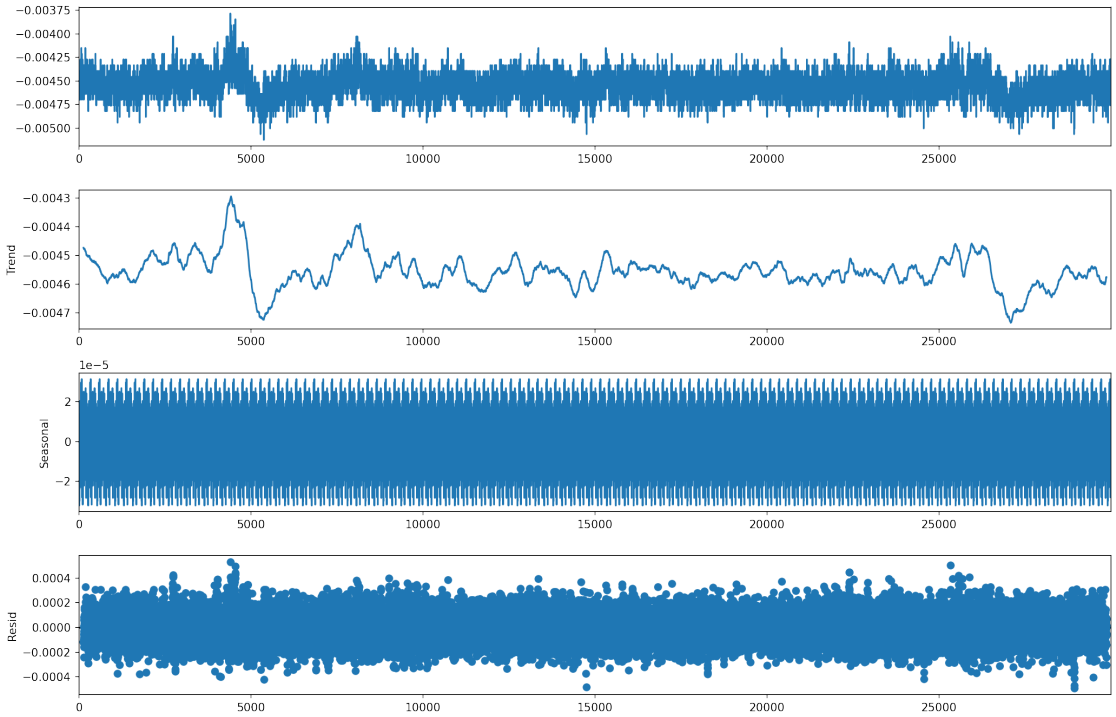

Figure 27: Seasonal decompose of Channel 6 of the ekg-signal

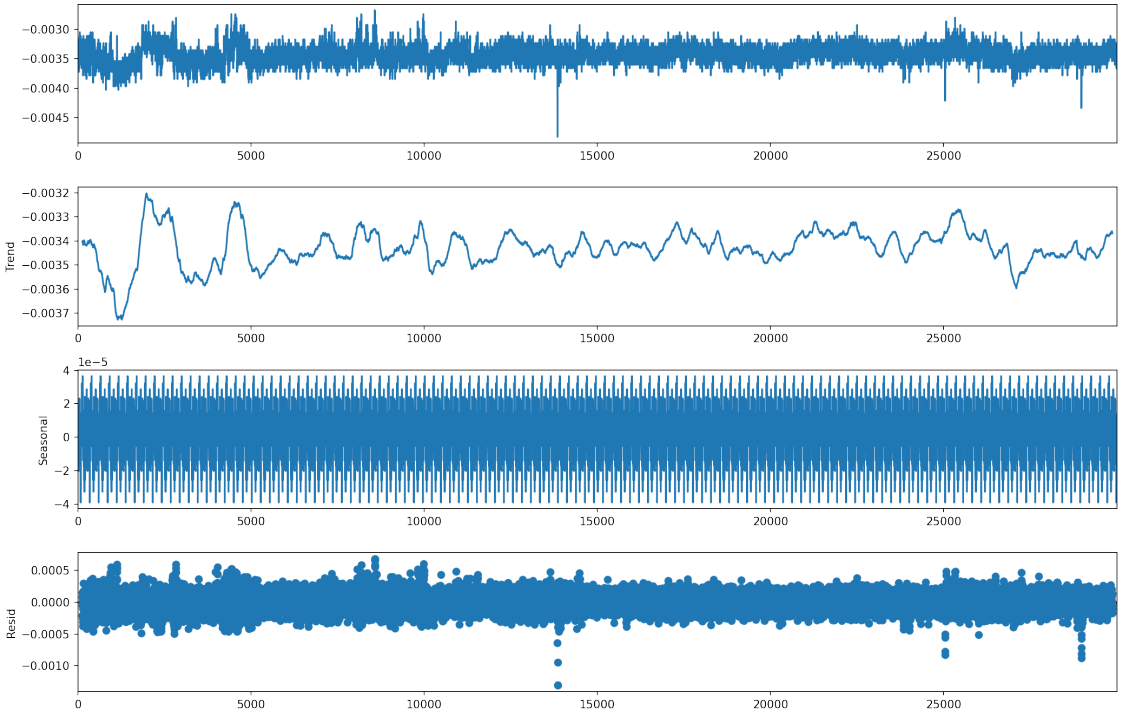

Figure 28: Seasonal decompose of Channel 7 of the ekg-signal

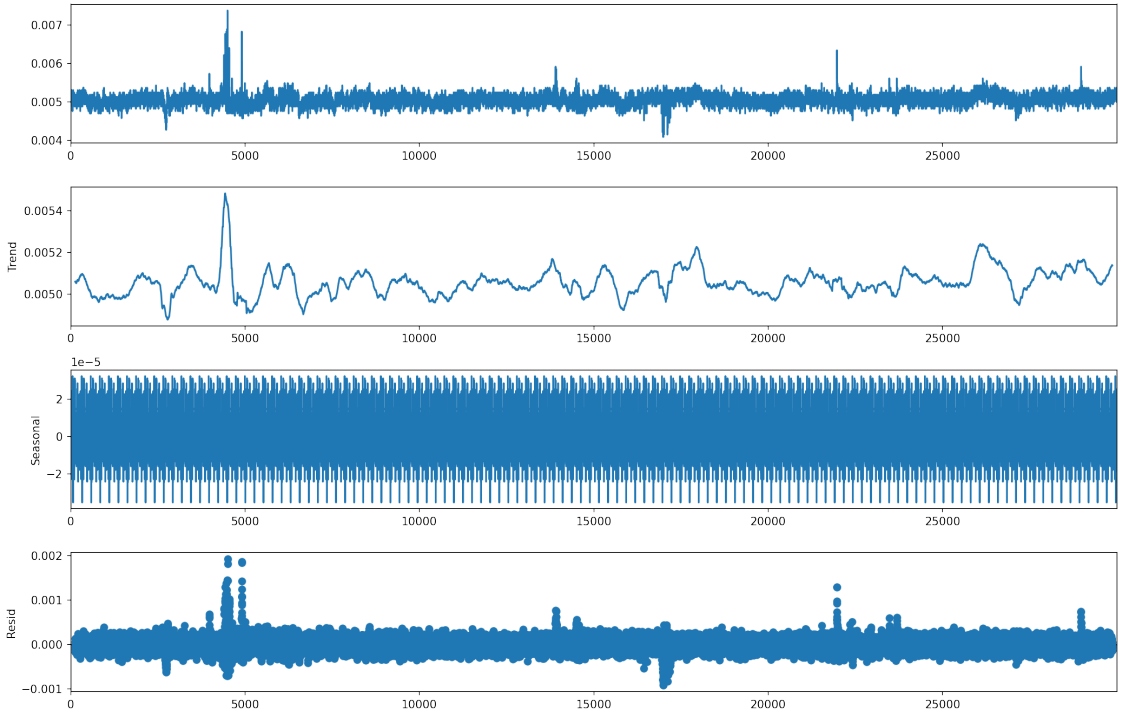

Figure 29: Seasonal decompose of Channel 8 of the ekg-signal

A.4. Power Consumption Germany

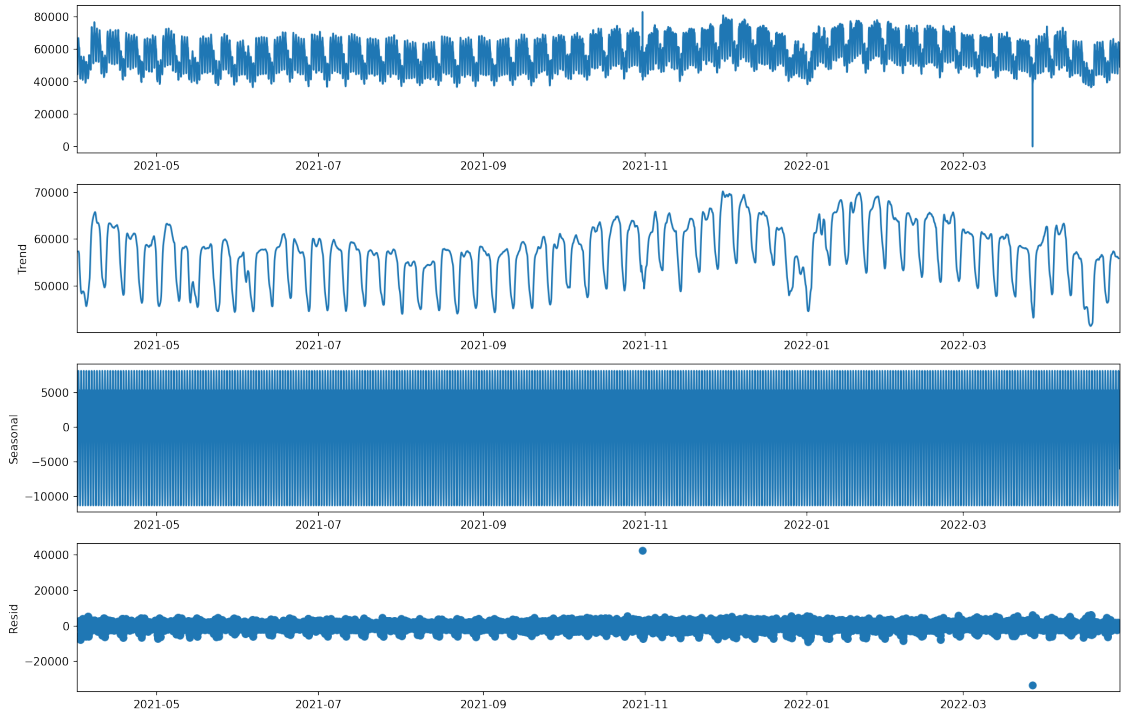

Figure 30: Seasonal decompose of the German power consumption from 04/2021 to 04/2022

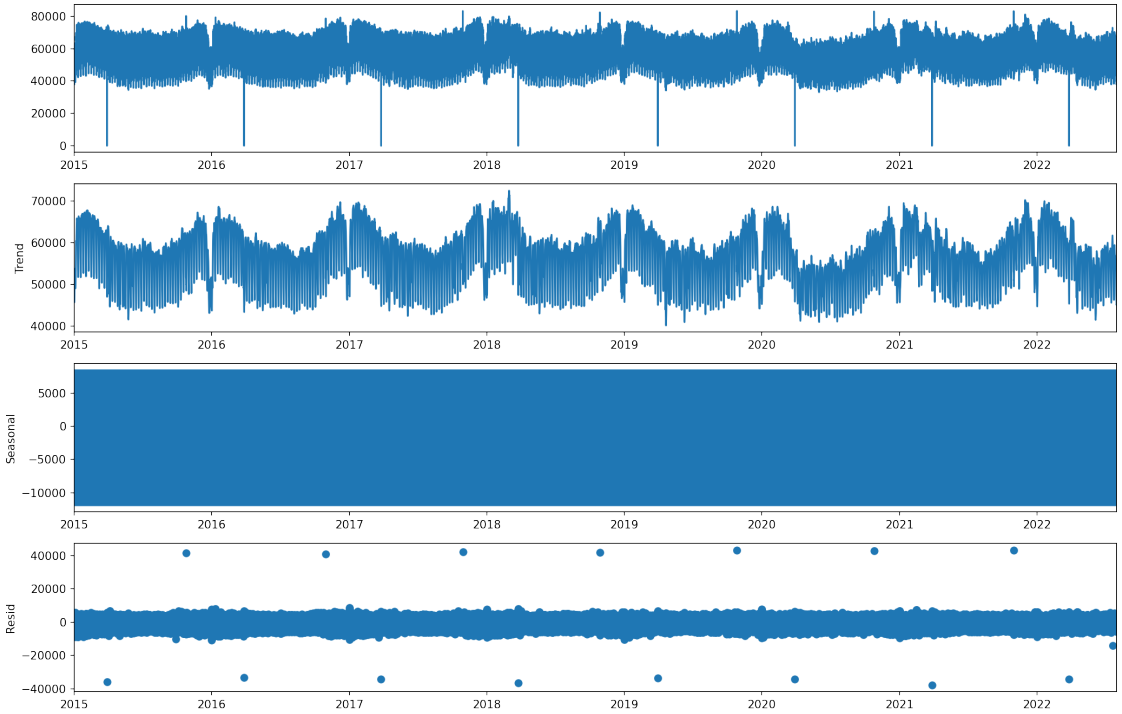

Figure 31: Seasonal decompose of the German power consumption from 01/2015 to 07/2022

A.5. Tesla Inc Stock Market Value

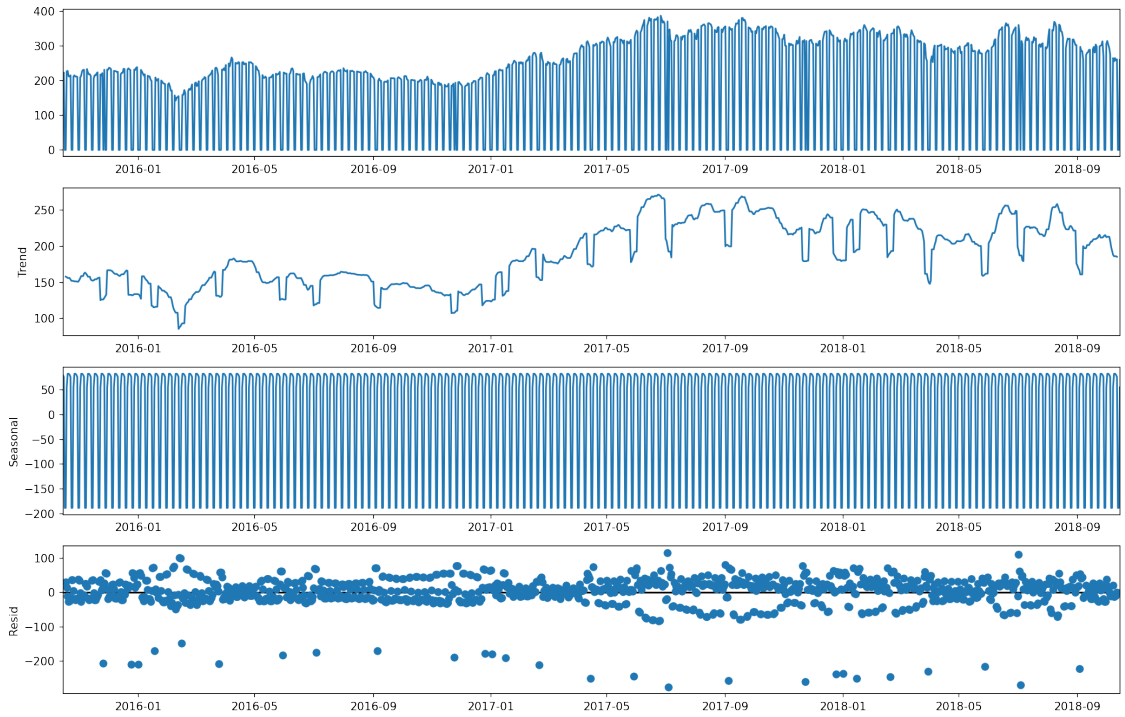

Figure 32: Seasonal decompose of the Tesla Inc Stock Market Open Value

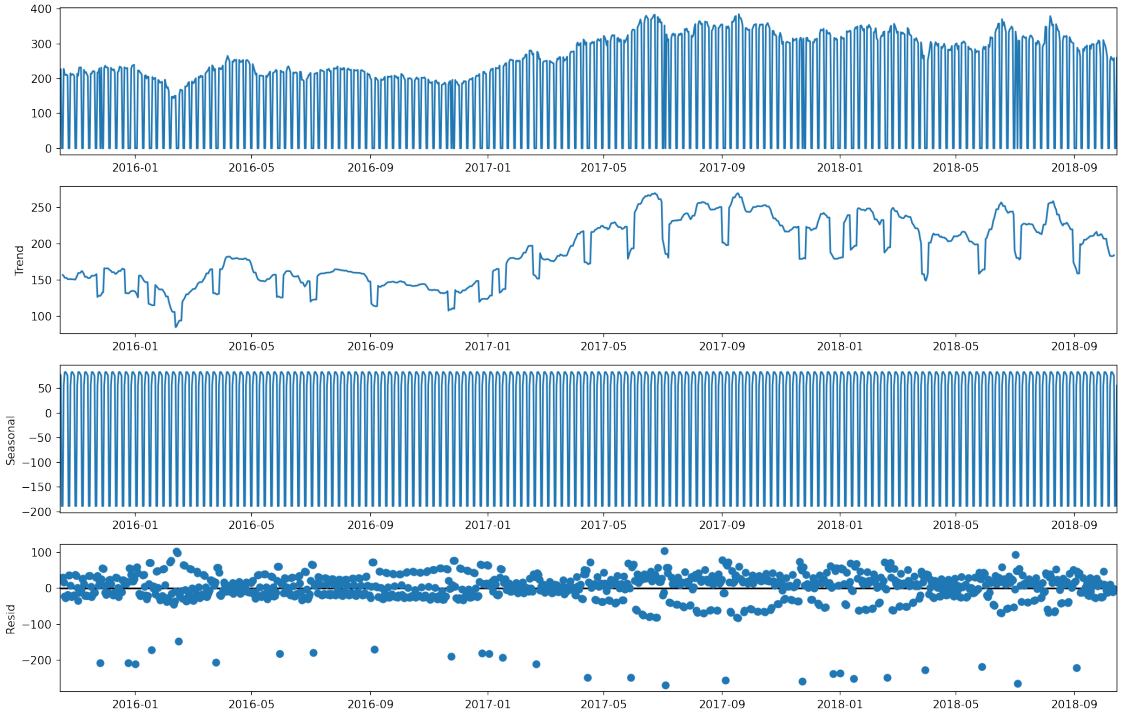

Figure 33: Seasonal decompose of the Tesla Inc Stock Market Close Value

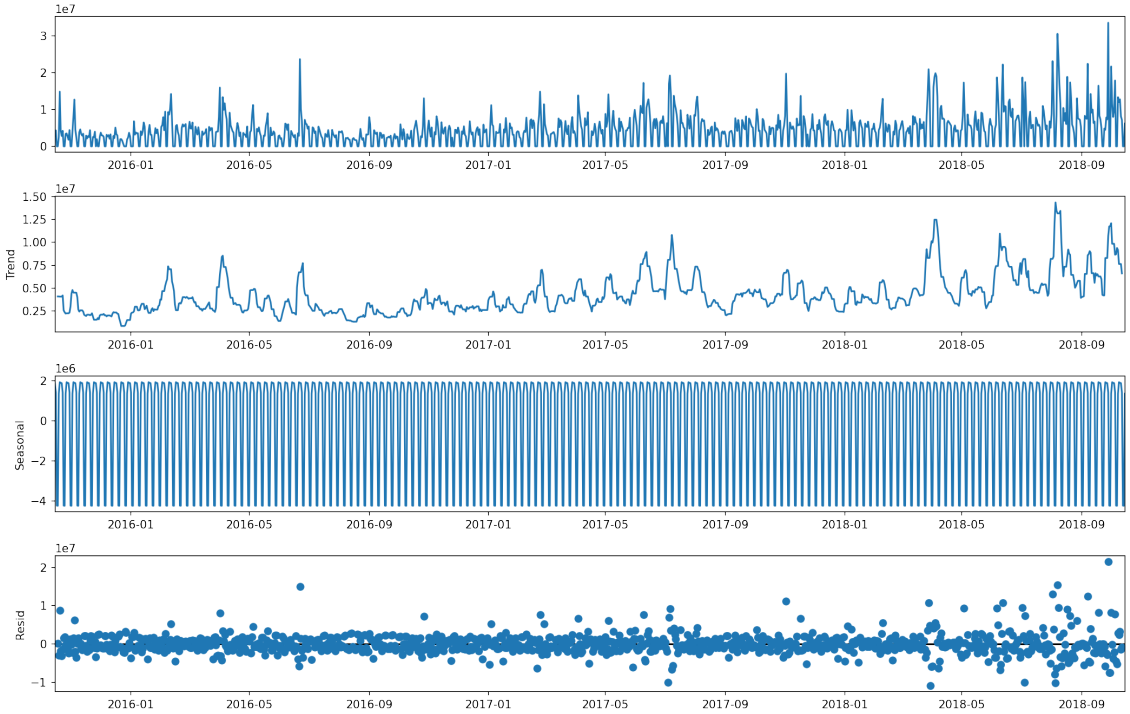

Figure 34: Seasonal decompose of the Tesla Inc Stock Market Volume Value

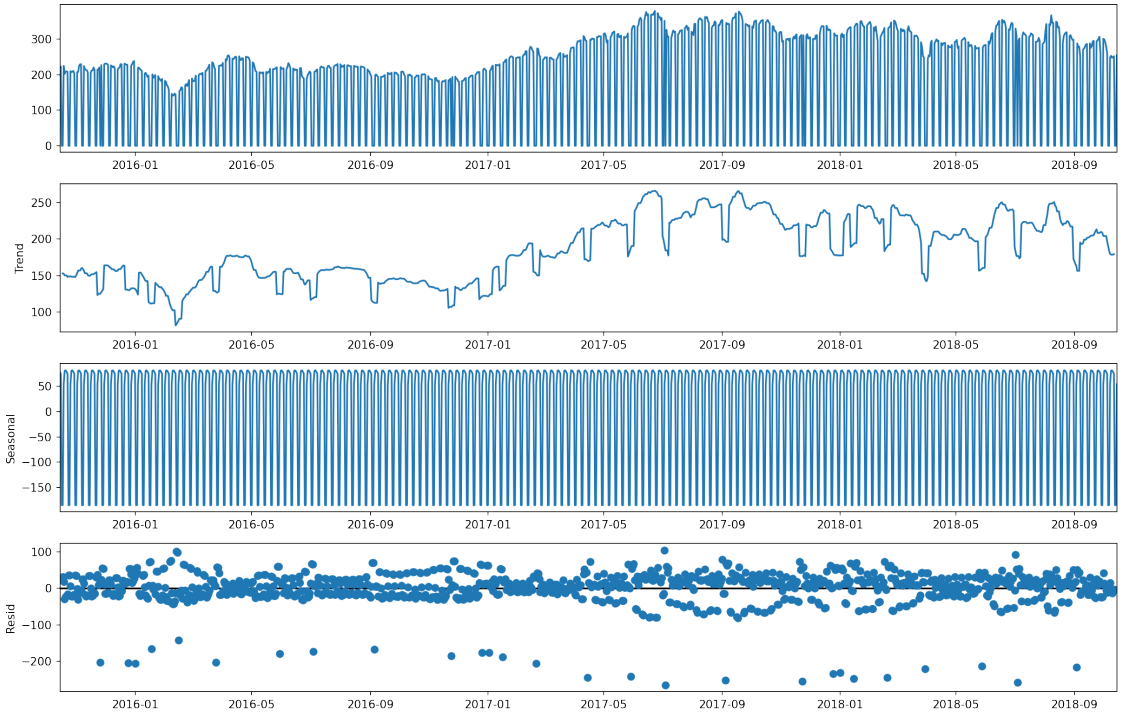

Figure 35: Seasonal decompose of the Tesla Inc Stock Market Low Value

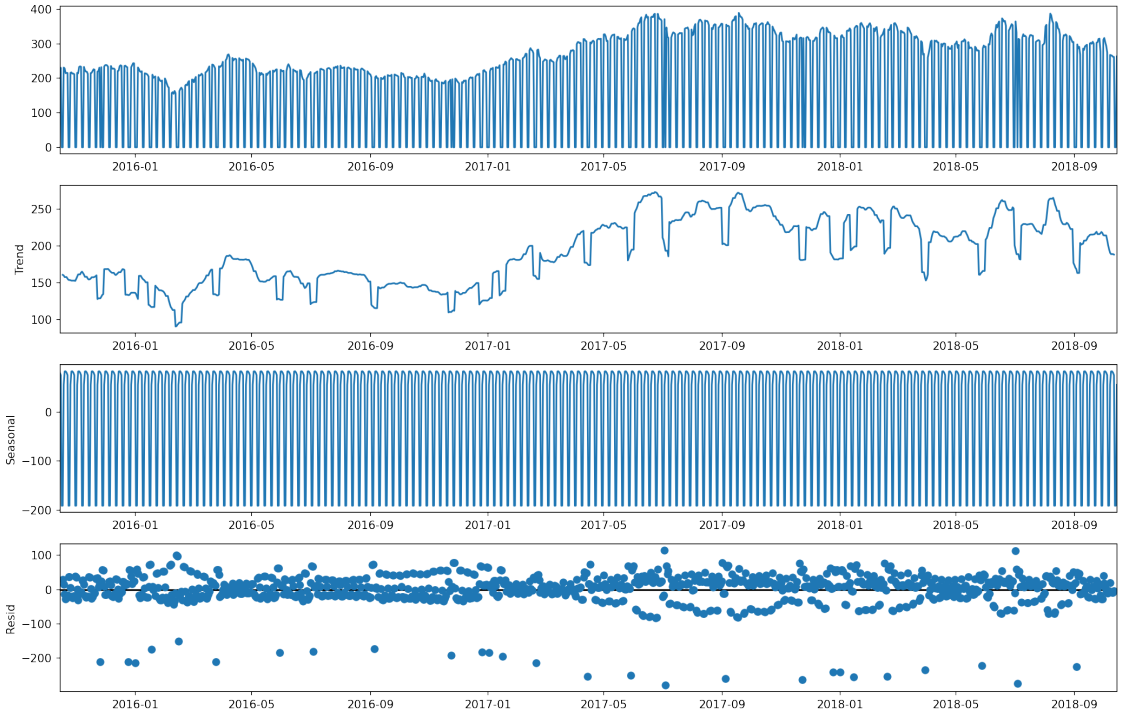

Figure 36: Seasonal decompose of the Tesla Inc Stock Market High Value

B

A.6. Water Temperature Well Ulm

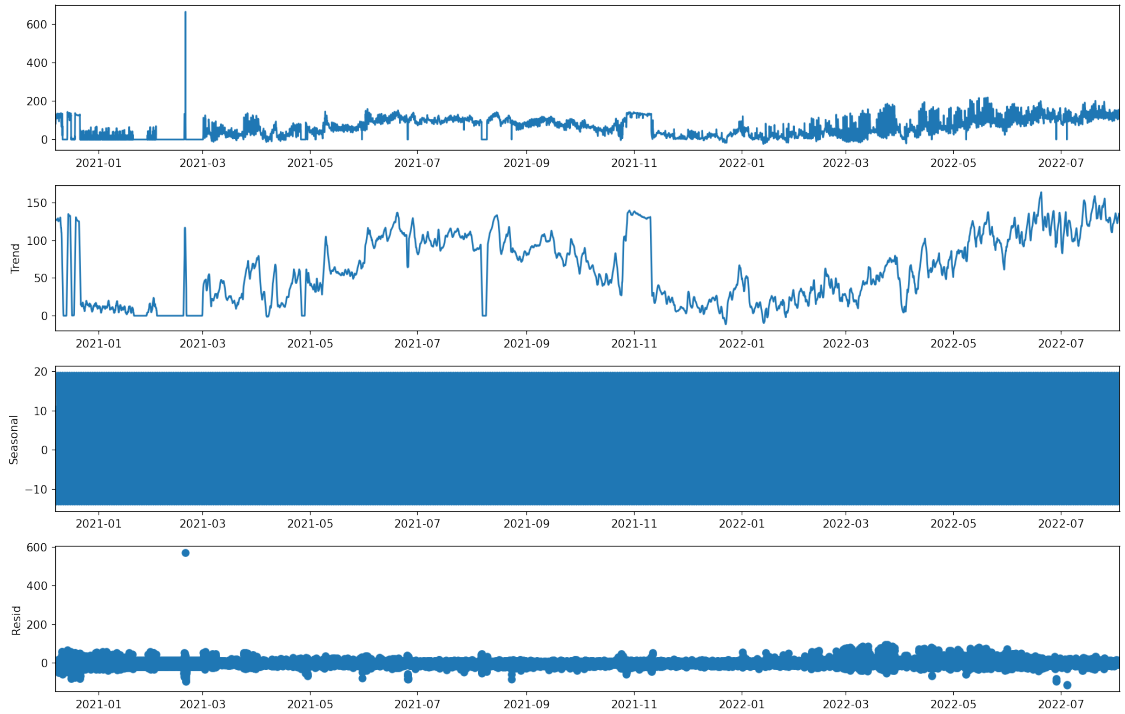

Figure 37: Seasonal decompose of the water temperatur at the well at the Weinhof in Ulm

B. Application Programming Interface

These are the request paths and the methods that are provided by the PAF.

B.1. Data Application Programming Interface

This part of the interface is responsible for the data preparation and analysis.

request path 1: $< host > /api/v1/data/ < action >$

request path 1: $< host > /api/v1/data/ < action > / < fileid >$

Request Method: POST

Possible actions:

- analyse: Analyse and prepare the dataset for the training of a model. Returns the id of the file. Additionally returns found seasonalities and trend, seasonal and error components if selected.
- missing_entries: Detects missing entries in a dataset. Returns a list of the found missing entries.
- insert_entries: Inserts new entries into the dataset. Returns the filled dataset
- interpolate: Interpolates missing values in the dataset. Returns the interpolated dataset
- seasonality: Detects seasonality in the dataset. Returns a list of found intervals of the dataset.
- resample: Resamples the interval to a given interval. Returns the resampled dataset
- decompose: separates the dataset into 3 components: Trend, seasonality and random error. Returns the separate components of the dataset

Required parameters:

- data_file (file): If no file id is given in the request path, a file for the dataset has to be given.

Optional parameters:

- interval (string): (required for the resampling and decompose action) target interval for the dataset.
- missing_dates (list of datetimes): (required for 'insert_entries' action) list of datetimes in the format 'dd.mm.yy HH:MM:SS'.
- interpolation_method (string): Interpolation method to be used ('linear', 'nearest', 'zero', 'slinear', 'quadratic', 'cubic', 'spline', 'barycentric', 'polynomial')

Following boolean options are available for 'analyse' action:

- find_missing: find and fill in missing values. The default value is 'True'
- find_seasonality: find seasonality in the dataset. The default value is 'True'. Disabling this can have negative affects on models which use this information for optimization.
- decompose: decomposes the dataset into the trend, seasonal and error component. The default value is 'False'. The decomposed information is not used by the framework and only for the user.

B.2. File Application Programming Interface

This interface allows the user to upload and download data files directly.

request path 1: $< host > /api/v1/files$

Request Method: POST
Uploads a data file to the framework
Required Parameters:

- data_file (file): the file to upload

Optional Parameters:

- separator: The separator used in the csv file. The default is ';'

request path 1: $< host > /api/v1/files/ < int : fileid >$

Request Method: GET
Downloads a data file from the framework.

B.3. Models Application Programming Interface

This interface is responsible for the training of single models, uploading or downloading model files and creating predictions for existing models

request path 1: $< host > /api/v1/models$

Request Method: POST

Creates and trains a new model
Required parameters:

- model_type (string): The model type to train
- data_file (string) or data_id (integer): Either a file or a file id for the dataset have to be given.

Optional parameters:

- separator (char): the separator that is used in the data file if no file id is given. The default is ';'
- columns (list of integer): The columns to use if a multivariate version is wanted. The default is univariate with the use of the first data column.
- multivariate (boolean): If set to True, all columns of the dataset will be used for the training of a multivariate model
- prediction_interval (either integer or date time in format 'dd.mm.yy HH:MM:SS'): the interval until which the finished model should predict to. If none is given, the model will just be trained and no prediction will be made.
- parameters (json of model parameters): Parameters of the model. If none are given, the model will optimize the parameters itself

Request Method: PUT

Uploads a model to the PAF.
Required parameters:

- model_file (file): file of the model
- model_type (string): model type

Optional parameters:

- num_classes (int): number of classes the model supports. The default value is 1 for univariate models

- resolution (String): Resolution of the training dataset (e.g. d - daily, w - weekly). This parameter is needed when for the model prediction a time stamp is given instead of a number of prediction steps.
- rmse (float): RMSE value of the model

request path 2: $< host > /api/v1/models/ < int : modelid >$
Downloads or deletes a model

Request Method: GET
Downloads the model with the corresponding id.
No parameters are available for this method.

Request Method: DELETE
Deletes the model with the corresponding id.
No parameters are available for this method.

request path 3: $< host > /api/v1/models/ < int : modelid > /predict$

Request Method: GET
Uses the model with the given id for a prediction.
Required parameters:

- prediction_interval (integer): The amount of steps that should be predicted.

B.4. Info Application Programming Interface

This part of the API is responsible for delivering general information like the available models.

request path: $< host > /api/v1/info/ < string : action >$

Request Method: GET
Possible actions:

- models: returns every supported model of the PAF
- multivariate_models: returns every supported multivariate model of the PAF

B.5. Auto Application Programming Interface

This is the access point for the automatic model selection feature of the PAF

request path 1: $< host > /api/v1/auto$

request path 2: $< host > /api/v1/auto/ < int : fileid >$

If the file id is given, the corresponding file already uploaded to the PAF will be used. If no id is given, the file is expected in the request body.

Request Method: POST

Required parameters:

- interval (string): the interval in which the dataset should be resampled to. (S: Second, min: Minute, H: Hour, D: Day, W: Week, M: Month, Q: Quarter, A: Year)
- data_file (csv file): If the file id is not given in the request path, a file has to be uploaded in the request body.

Optional parameters:

- separator (char): the separator that is used in the data file if no file id is given. The default is ';'
- columns (list of integer): The columns to use if a multivariate version is wanted. The default is univariate with the use of the first data column.
- multivariate (boolean): If set to True, all columns of the dataset will be used for the training of a multivariate model
- models (list of strings): The models that should be used. In default all available models for the prediction type (uni- or multivariate) will be used
- prediction_interval (either integer or date time in format 'dd.mm.yy HH:MM:SS): the interval until which the finished model should predict to. If none is given, the model will just be trained and no prediction will be made.

C. Evaluation

C.1. Evaluation Results

C.1.1. Akaike Information Criterion Results

For some models the AIC value is not available due to the in section 4.3 explained problem with the maximum likelihood function that can not be defined easily for every model type.

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
vcGB	H	27102	27663	-	35524	22890
	D	1296	1637	1416	1724	1427
vcGH	H	29036	29795	-	37502	25107
	D	1365	1784	1516	1824	1519
vcHH	H	27856	28473	-	36303	23907
	D	1338	1689	14	1776	1481
vcHS	H	26252	26682	-	34574	22065
	D	1249	1564	1375	1658	1354
vcPH	H	26212	26823	-	34552	22165
	D	1276	1638	20	1705	1406
vcHM	H	28369	29170	-	36671	24326
	D	1347	1717	1483	1781	1481
vcPK	H	18136	18633	-	24060	15037
	D	799	1139	18	1155	962
vcSP	H	21538	22007	-	29883	17134
	D	1061	1359	10	1452	1148

Table 38: AIC evaluation results visit counter datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
covC	D	7737	7935	12	10363	6162
	W	1320	1569	1530	1351	1304
	M	174	501	10	423	436
covCC	D	7772	8729	18	10122	6244
	W	1545	1883	10	2012	1665
	M	305	662	597	635	595
covD	D	1361	1519	22	3865	-242
	W	362	445	424	594	198
	M	132	251	10	255	187
covCD	D	1355	1666	12	3967	-174
	W	584	1285	717	994	600
	M	161	383	12	386	318

Table 39: AIC evaluation results Covid datasets

Dataset/ Model	AR	SARIMA	SARIMA grid	TBATS	ES
ekg1	-238832	-234105	-239375	-38901	-221631
ekg2	-183140	-182045	-183158	2632	-187444
ekg3	-193338	-191432	-193509	-10637	-207291
ekg4	-376155	-374992	-376418	-202912	-442069
ekg5	-289920	-245013	-263207	-102176	-309638
ekg6	-377679	-376878	-377948	-204264	-443305
ekg7	-375935	-373087	-375204	-202516	-440836
ekg8	-371610	-371227	-371791	-198026	-437410

Table 40: AIC evaluation results ekg datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
powerL	D	54168	ValueError (4.5.1)	54699	65407	48449
	W	8615	9040	8971	9818	8342
	M	1889	2412	1980	2510	2226
powerS	D	7258	7844	20	8601	6817
	W	879	1370	1160	1403	1231
	M	236	375	324	370	346
water	D	6684	7014	22	8658	5766
	W	1051	1332	18	1432	1139
	M	181	352	315	361	316

Table 41: AIC evaluation results power consumption and water temperature datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
teslaC	D	9149	9278	8	12880	6743
	W	1439	1625	315	1886	1275
	M	165	498	467	511	410
teslaH	D	9171	9302	10	12902	6766
	W	1441	1628	1578	1889	1278
	M	167	499	468	512	413
teslaL	D	9121	9250	12	12855	6716
	W	1436	1621	12	1881	1270
	M	178	497	16	511	411
teslaO	D	9146	9275	10	12879	6741
	W	1438	1625	14	1885	1274
	M	166	498	467	512	412
teslaV	D	27628	28250	22	31586	25718
	W	3892	4394	4224	4625	4038
	M	779	1086	14	1103	1003

Table 42: AIC evaluation results Tesla stock datasets

C.1.2. Bayesian Information Criterion Results

Equally to the AIC value, it is not possible to get the BIC value for every model. Due to missing support from the TBATS model, no BIC value could be specified

for that model.

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
vcGB	H	27369	27696	-	-	23032
	D	1343	1651	1424	-	1448
vcGH	H	29302	29844	-	-	25248
	D	1412	1796	1527	-	1540
vcHH	H	28122	28511	-	-	24049
	D	1385	1700	29	-	1501
vcHS	H	26519	26720	-	-	22075
	D	1296	1575	1386	-	1375
vcPH	H	26479	26855	-	-	22307
	D	1323	1649	42	-	1426
vcHM	H	28635	29197	-	-	24468
	D	1394	1728	1494	-	1506
vcPK	H	18388	18669	-	-	15172
	D	837	1149	34	-	980
vcSP	H	21804	22061	-	-	17276
	D	1108	1371	21	-	1168

Table 43: BIC evaluation results visit counter datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
covC	D	7839	7975	38	-	6211
	W	1360	1584	1535	-	1314
	M	180	505	15	-	440
covCC	D	7847	8756	58	-	6262
	W	1585	1898	22	-	1675
	M	311	664	600	-	597
covD	D	1459	1559	70	-	-224
	W	402	460	436	-	208
	M	140	252	14	-	189
covCD	D	1430	1688	38	-	-156
	W	624	1290	732	-	611
	M	167	386	17	-	320

Table 44: BIC evaluation results Covid datasets

Dataset/ Model	AR	SARIMA	SARIMA grid	TBATS	ES
ekg1	-238695	-234048	-239278	-	-221599
ekg2	-183035	-181988	-183061	-	-187411
ekg3	-193201	-191375	-193412	-	-207259
ekg4	-376010	-374936	-376345	-	-442053
ekg5	-289775	-244956	-263135	-	-309590
ekg6	-377534	-376829	-377859	-	-443289
ekg7	-375781	-373031	-375115	-	-440820
ekg8	-371497	-371170	-371710	-	-437394

Table 45: BIC evaluation results ekg datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
powerL	D	54299	ValueError (4.5.1)	54767	-	48500
	W	8689	9077	8997	-	8350
	M	1946	2421	2001	-	2230
powerS	D	7343	7878	57	-	6850
	W	907	1374	1175	-	1234
	M	236	376	324	-	347
water	D	6780	7039	67	-	5774
	W	1091	1338	37	-	1143
	M	183	356	316	-	316

Table 46: BIC evaluation results power consumption and water temperature datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES
teslaC	D	9259	9306	27	-	6786
	W	1494	1640	335	-	1280
	M	177	503	471	-	416
teslaH	D	9281	9330	33	-	6809
	W	1495	1642	1601	-	1283
	M	179	504	471	-	415
teslaL	D	9230	9274	40	-	6759
	W	1490	1635	28	-	1276
	M	189	503	26	-	414
teslaO	D	9255	9304	33	-	6784
	W	1492	1639	33	-	1280
	M	178	504	471	-	415
teslaV	D	27737	28298	74	-	25761
	W	3946	4405	4243	-	4043
	M	801	1090	18	-	1006

Table 47: BIC evaluation results Tesla stock datasets

C.1.3. Mean Square Error Results

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
vcGB	H	1489560	5974368	-	9305421
	D	32658689729	4827302268	44374616892	6698186533
vcGH	H	10598789	35220841	-	28791044
	D	238976098767	7714489012	251477119300	19038811161
vcHH	H	3023124	17630627	-	23914888
	D	29354099478	11589010511	13139269439	54045854322
vcHS	H	368153	872570	-	1377989
	D	140476687	159106298	3319094845	212248914
vcPH	H	2141673	8490740	-	11051986
	D	17484870925	7195734048	1165124972	56360866940
vcHM	H	5494493	20516153	-	25377395
	D	1.1544695e+16	26370854755	315454041645	194099246031
vcPK	H	346822	623684	-	326270
	D	25456419	59565818	2.6235681e+18	26775499
vcSP	H	69981	36161	-	84543
	D	247811212	22604563	34577835	31080920

Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
vcGB	H	21960054	675934	612412	9860206
	D	6750916432	778570430	636228740	2298111725
vcGH	H	145502214	2827865	2246058	36248117
	D	18522224915	4665982000	4559233500	7180041863
vcHH	H	95739139	1298109	1030175	20128865
	D	13623741122	1484127600	1301663200	4720082074
vcHS	H	2110237	372003	346692	1505077
	D	890591664	208247980	186133700	279929233
vcPH	H	49939241	357492	383756	8603979
	D	7474354587	821948860	703831360	2123557513
vcHM	H	87551287	1702604	1597746	21093352
	D	209387636541	2207416300	1963872500	4511662592
vcPK	H	519218	145480	141923	981733
	D	18526536	63862636	68508630	396472034
vcSP	H	194594	25815074	28955725	120167
	D	39158058	15483223	17673162	28195818

Table 48: MSE evaluation results visit counter datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
covC	D	28090398	57453252	69218212	40510246
	W	4566249388	3608218080	1298362429	1344277128
	M	4.2265642e+18	30580439383	45273107542	42292944100
covCC	D	464696522716	161340956079	249042454922	24033829933
	W	9.6279226e+12	2.8984593e+15	1.2197888e+13	1.0973144e+12
	M	1.3994367e+16	1.0426083e+14	5.9927554e+13	7.5222154e+13
covD	D	50392	54	59	136
	W	9072767473	355	65307	57455
	M	1875602017122	30872	30872	66146
covCD	D	839353509	260416	712260	322698
	W	716216983102	34978017	566655	30331075
	M	2.6195066e+15	330347101	561210233	259139978
Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
covC	D	28933467	51090	33432203	45959430
	W	1294767398	27209510	58944016	623909059
	M	101732336164	614636700	671987800	6061742892
covCC	D	28063016151	3159579400	6846330400	234775870127
	W	1435486570075	42989560000	60965850000	3895030701057
	M	1.0400966e+14	1.8124354e+12	1.7258021e+12	4.3869830e+16
covD	D	1496	1.1	0.9	27
	W	61704	211	167	814
	M	43813	7385309	6681	28969
covCD	D	322661	12348335	67342	525018
	W	8432996	113189945	193957	11172001
	M	280910218	11176150	10008509	164068320

Table 49: MSE evaluation results Covid datasets

Dataset/ Model	AR	SARIMA	SARIMA grid	TBATS
ekg1	0.0068	0.0068	0.0068	0.0090
ekg2	0.0193	0.0193	0.0193	457
ekg3	0.0076	0.0076	0.0078	0.1087
ekg4	1.5206e-08	1.5252e-08	1.5127e-08	1.5046e-08
ekg5	0.0002	4.8013	0.0024	0.0002
ekg6	1.7287e-08	1.7292e-08	1.7305e-08	1.7730e-08
ekg7	1.7870e-08	1.7939e-08	2.3318e-08	1.9849e-08
ekg8	1.8870e-08	1.8881e-08	1.8872e-08	1.9394e-08

Dataset/ Model	ES	LSTM	GRU	Prophet
ekg1	40.4598	0.0004	0.0004	-
ekg2	14.9864	0.0013	0.0013	-
ekg3	0.0267	0.0004	0.0004	-
ekg4	2.7118e-08	9.6713e-09	9.5733e-09	-
ekg5	1.4213	2.4489e-05	1.4677e-05	-
ekg6	1.7029e-08	9.1425e-09	9.1059e-09	-
ekg7	2.2535e-08	1.0342e-08	1.0242e-08	-
ekg8	1.8492e-08	1.2099e-08	1.2200e-08	-

Table 50: MSE evaluation results ekg datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
powerL	D	9991855016	ValueError (4.5.1)	14941772165	17452880450
	W	388938034843	173810483980	576886375883	161349176398
	M	3.6259783e+13	2.1974787e+13	3.4765093e+16	2.0908727e+16
powerS	D	8974290455	16082692176	1885927389111	26627924652
	W	2424654713464	1706486504983	702247279260	2.8525920e+15
	M	6.3321475e+13	4.5064292e+14	4.7186193e+14	4.3927131e+14
water	D	2412500	2419327	6602209	4553283
	W	45975282	96851352	1185960461236	119535409
	M	835220008	853520387	2279074338	1546293137
Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
powerL	D	26169889980	8107991000	6715552300	419865341839
	W	1.1658124e+12	276003320000	226224570000	1.5878940e+16
	M	2.2411578e+13	2.4921600e+13	2.4012298e+13	2.7684784e+14
powerS	D	28577280435	15564264000	14325124000	320585753191
	W	3.1312076e+12	348978100000	289783600000	2.1702654e+13
	M	7.7000597e+14	5.0220030e+14	3.6000704e+14	2.7203021e+14
water	D	4181235	344562	290813	1781208
	W	109723211	19074292	16606347	94870776
	M	2177921628	408674560	337701820	2309858055

Table 51: MSE evaluation results power consumption and water temperature datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
teslaC	D	2595	2578	65160	2975
	W	103207	117478	2.4857049e+14	102490
	M	1955378308	1908906	3116129	1943429
teslaH	D	2728	2738	67930	3088
	W	104873	119494	204161	103579
	M	3443730977	1991419	3164510	1964789
teslaL	D	2480	2491	62416	2892
	W	102767	115212	2094802	102379
	M	874822082	1798395	376330935	1873114
teslaO	D	2608	2644	65268	2979
	W	103124	117097	2189217	101683
	M	2622214405	1892915	3027985	1891302
teslaV	D	3.3759713e+13	4.0624165e+13	8.2545734e+13	3.3641099e+16
	W	6.6761721e+14	7.3379150e+14	1.3686276e+15	7.1334131e+14
	M	1.8076539e+15	8.0405384e+15	3.4997032e+18	7.4943183e+15

Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
teslaC	D	2939	4136	4918	31271
	W	103359	26117943	32940395	971870
	M	3982386	5039440	1444002	22195258
teslaH	D	3020	4216	5426349	31784
	W	104376	27451932	31415156	1017744
	M	1940269	4138348	1528963	22611937
teslaL	D	2901	3919	4546	30195
	W	103462	26124863	33133	1011970
	M	1832507	5368543	1386020	21693856
teslaO	D	2962	4136	5231	31243
	W	102653	28672846	26776166	995691
	M	1854452	3978090	1373203	22149016
teslaV	D	3.6545266e+13	8861707000000	9787153000000	3.2157805e+13
	W	6.9322149e+14	1.1803707e+14	1.2540169e+14	3.5737393e+14
	M	7.7591981e+15	Error (4.5.1)	Error (4.5.1)	5.6035518e+15

Table 52: MSE evaluation results Tesla stock datasets

C.1.4. Root Mean Square Error Results

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
vcGB	H	1220	2444	-	3050
	D	180717	69478	210652	81842
vcGH	H	3255	5934	-	5365
	D	488851	87832	501474	137981
vcHH	H	1738	4198	-	4890
	D	171330	107652	114626	232477
vcHS	H	606	934	-	1173
	D	11852	12613	57611	14568
vcPH	H	1463	2913	-	3324
	D	132230	84827	34133	237404
vcHM	H	2344	4529	-	5037
	D	3397748	162391	561652	440566
vcPK	H	588	789	-	571
	D	5045	7717	1619743225	5174
vcSP	H	264	190	-	290
	D	15742	4754	5880	5575

Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
vcGB	H	4686	675934	782	3140
	D	82163	778570430	25223	47938
vcGH	H	12062	2827865	1498	6020
	D	136096	4665982000	67522	84735
vcHH	H	9784	1298109	1014	4486
	D	116720	1484127600	36078	68702
vcHS	H	1452	372003	588	1226
	D	29842	208247980	13643	16731
vcPH	H	7066	357492	619	2933
	D	86454	821948860	26529	46082
vcHM	H	9356	1702604	1264	4592
	D	457588	2207416300	44315	67168
vcPK	H	720	145480	376	990
	D	4304	63862636	8276	19911
vcSP	H	441	25815074	170	346
	D	6257	15483223	4203	5309

Table 53: RMSE evaluation results visit counter datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
covC	D	5300	7579	8319	6364
	W	67574	60068	36032	36664
	M	2055860955	174872	212774	205652
covCC	D	681686	401672	499041	155028
	W	3102889	1702486	3492547	1047527
	M	118297792	10210819	7741288	8673070
covD	D	224	7	7	11
	W	95251	18	255	239
	M	1369526	175	175	257
covCD	D	28971	510	843	568
	W	846296	5914	752	5507
	M	1618489	18175	23689	16097

Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
covC	D	5378	226	182	6779
	W	35982	5216	7677	24978
	M	318955	24791	25922	77857
covCC	D	167520	56210	82742	484536
	W	1198117	207339	246912	1973583
	M	10198512	1346267	1313697	6623430
covD	D	38	1	0.9	5
	W	248	14	12	28
	M	209	85	81	170
covCD	D	568	111	259	724
	W	2903	336	440	3342
	M	16760	3343	3163	12808

Table 54: RMSE evaluation results Covid datasets

Dataset/ Model	AR	SARIMA	SARIMA grid	TBATS
ekg1	0.0830	0.0830	0.0830	0.0953
ekg2	0.1389	0.1389	0.1390	21.3852
ekg3	0.0876	0.0877	0.0887	0.3297
ekg4	0.0001	0.0001	0.0001	0.0001
ekg5	0.0151	2.1911	0.0491	0.0151
ekg6	0.0001	0.0001	0.0001	0.0001
ekg7	0.0001	0.0001	0.0001	0.0001
ekg8	0.0001	0.0001	0.0001	0.0001

Dataset/ Model	ES	LSTM	GRU	Prophet
ekg1	6.3608	0.0205	0.0205	-
ekg2	3.8712	0.0367	0.0367	-
ekg3	0.1636	0.0217	0.0219	-
ekg4	0.0001	9.8343e-05	9.7843e-05	-
ekg5	1.1921	0.0049	0.0038	-
ekg6	0.0001	9.5616e-05	9.5425e-05	-
ekg7	0.0001	0.0001	0.0001	-
ekg8	0.0001	0.0001	0.0001	-

Table 55: RMSE evaluation results ekg datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
powerL	D	99959	ValueError (4.5.1)	122236	132109
	W	623648	416905	759530	401682
	M	6021609	4687727	5896193	4572606
powerS	D	94732	126817	1373290	163180
	W	1557130	1306325	838001	1688961
	M	25163758	21228351	21722383	20958800
water	D	1553	1555	2569	2133
	W	6780	9841	1089018	10933
	M	28900	29215	47739	39322

Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
powerL	D	161771	90044	81948	647970
	W	1079727	525360	475630	3984838
	M	4734086	4992153	4900234	16638745
powerS	D	169048	124756	119687	566202
	W	1769521	590743	538315	4658610
	M	27748981	22409825	18973851	16493338
water	D	2044	586	539	1334
	W	10474	4367	4075	9740
	M	46668	20215	18376	48060

Table 56: RMSE evaluation results power consumption and water temperature datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS
teslaC	D	50	50	255	54
	W	321	342	15766118	320
	M	44219	1381	1765	1394
teslaH	D	52	52	260	55
	W	323	345	451	321
	M	58683	1411	1778	1401
teslaL	D	49	49	249	53
	W	320	339	1447	319
	M	29577	1341	19399	1368
teslaO	D	51	51	255	54
	W	321	342	1479	318
	M	51207	1375	1740	1375
teslaV	D	5810310	6373708	9085468	5800094
	W	25838289	27088586	36994967	26708450
	M	42516514	89669049	1870749390	86569731

Dataset/ Model	Res.	ES	LSTM	GRU	Prophet
teslaC	D	54	64	70	176
	W	321	161	181	985
	M	1995	2244	1201	4711
teslaH	D	54	64	73	178
	W	323	165	177	1008
	M	1392	2034	1236	4755
teslaL	D	53	62	67	173
	W	321	161	182	1005
	M	1353	2317	1177	4657
teslaO	D	54	64	72	176
	W	320	169	163	997
	M	1361	1994	1171	4706
teslaV	D	6045268	2976861	3128442	5670785
	W	26329099	10864486	11198289	18904336
	M	88086310	Error (4.5.1)	Error (4.5.1)	74856875

Table 57: RMSE evaluation results Tesla stock datasets

C.1.5. Training Duration Results

The data in the following tables 58, 59, 60, 61 and 62 is given in seconds.

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES	LSTM	GRU	Prophet
vcGB	H	0.63	205	-	91	0.8	198	184	26
	D	0.04	4	686	11	0.1	75	74	2
vcGH	H	0.11	482	-	176	0.3	201	184	20
	D	0.04	0.5	659	16	0.1	75	74	2
vcHH	H	0.11	438	-	63	0.3	202	184	16
	D	0.04	6	652	11	0.1	76	74	2
vcHS	H	0.11	329	-	76	0.3	198	184	19
	D	0.04	0.6	648	14	0.1	77	74	2
vcPH	H	0.11	202	-	79	0.3	203	186	26
	D	0.06	2	661	11	0.1	76	74	3
vcHM	H	0.11	117	-	54	0.3	200	184	14
	D	0.04	4	629	15	0.1	76	74	2
vcPK	H	0.10	333	-	53	0.2	163	150	21
	D	0.04	0.7	580	17	0.3	76	73	2
vcSP	H	0.11	499	-	72	0.3	201	187	15
	D	0.04	8	834	10	0.1	77	73	2

Table 58: Training duration visit counter datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES	LSTM	GRU	Prophet
covC	D	0.55	74	4051	40	0.64	121	118	3
	W	0.03	0.8	12	10	0.04	82	75	2
	M	0.03	0.9	9	11	0.03	78	73	3
covCC	D	0.04	14	49	65	0.07	113	108	4
	W	0.03	2	13	14	0.03	83	76	2
	M	0.03	0.1	6	9	0.03	78	74	3
covD	D	0.05	56	3868	30	0.23	122	114	3
	W	0.03	1	11	9	0.04	80	79	3
	M	0.04	0.2	126	7	0.11	78	77	3
covCD	D	0.04	3	44	24	0.04	112	113	3
	W	0.03	0.1	14	14	0.02	81	81	2
	M	0.03	0.3	9	6	0.02	78	78	3

Table 59: Training duration Covid datasets

Dataset/ Model	AR	SARIMA	SARIMA grid	TBATS	ES	LSTM	GRU	Prophet
ekg1	0.654	74	997	1371	1.27	1343	1240	-
ekg2	0.122	70	1010	1652	0.74	1343	1227	-
ekg3	0.119	80	1059	1050	0.76	1377	1224	-
ekg4	0.284	725	16123	5825	19.64	1520	1300	-
ekg5	0.172	202	5685	1569	4.41	1463	1274	-
ekg6	0.149	516	15544	761	4.28	1517	1292	-
ekg7	0.161	495	16074	523	4.19	1523	1294	-
ekg8	0.121	54	1056	983	1.38	1384	1218	-

Table 60: Training duration ekg datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES	LSTM	GRU	Prophet
powerL	D	0.53	ValueError (4.5.1)	14098	163	2.09	213	208	6
	W	0.05	807	450	70	0.52	93	90	3
	M	0.04	1.09	1199	24	0.30	75	76	4
powerS	D	0.05	59	2182	35	0.43	90	92	3
	W	0.04	0.15	155	8	0.33	74	75	4
	M	0.02	0.08	6	10	0.03	74	74	2
water	D	0.53	52	3485	50	0.62	108	102	3
	W	0.04	0.27	177	8	0.08	80	78	2
	M	0.03	0.36	6	13	0.03	77	76	2

Table 61: Training duration power consumption and water temperature datasets

Dataset/ Model	Res.	AR	SARIMA	SARIMA grid	TBATS	ES	LSTM	GRU	Prophet
teslaC	D	0.53	21	7072	42	0.72	135	127	4
	W	0.04	89	287	25	0.44	81	80	3
	M	0.06	0.3	10	8	0.03	75	74	3
teslaH	D	0.05	41	6893	49	0.23	136	128	4
	W	0.08	92	295	27	0.43	82	80	2
	M	0.07	0.2	10	7	0.04	79	75	4
teslaL	D	0.08	15	6850	46	0.25	137	127	3
	W	0.05	92	303	26	0.44	84	78	2
	M	0.03	0.2	11	7	0.04	77	72	3
teslaO	D	0.05	26	6726	47	0.23	137	120	4
	W	0.06	120	287	23	0.46	83	78	2
	M	0.03	0.3	10	7	0.03	78	71	3
teslaV	D	0.06	165	5406	42	0.21	137	122	3
	W	0.04	0.6	204	16	0.40	85	79	2
	M	0.04	0.1	482	16	0.35	Error (4.5.1)	Error (4.5.1)	4

Table 62: Training duration Tesla stock datasets

C.2. Parameter of the Best Model

These are the parameters used by the best version of each model types for the results in section 4.5.

C.2.1. Autoregression

Dataset/ Model	Res.	lags
vcGB	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcGH	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcHH	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcHS	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcPH	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcHM	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcPK	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
vcSP	H	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24]
	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]

Table 63: Parameters of best AR models of visitor datasets

Dataset/ Model	Res.	lags
covC	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
covCC	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
covD	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8]
covCD	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Table 64: Parameters of best AR models of Covid datasets

Dataset/ Model	lags
ekg1	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
ekg2	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
ekg3	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
ekg4	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
ekg5	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
ekg6	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
ekg7	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
ekg8	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

Table 65: Parameters of best AR models of ekg datasets

Dataset/ Model	Res.	lags
powerL	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
powerS	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4]
water	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7]

Table 66: Parameters of best AR models of power consumtion and water temperature datasets

Dataset/ Model	Res.	lags
teslaC	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
teslaH	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
teslaL	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
teslaO	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
teslaV	D	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	W	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
	M	[1, 2, 3, 4, 5, 6, 7, 8]

Table 67: Parameters of best AR models of tesla stock datasets

C.2.2. Seasonal Autoregressive Integrated Moving Average

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
vcGB	H	(1,0,0)(1,0,2,24)
	D	(3,0,0)(1,0,1,7)
vcGH	H	(4,0,1)(2,0,0,24)
	D	(0,0,1)(0,0,2,7)
vcHH	H	(1,0,1)(2,0,1,24)
	D	(1,0,1)(1,0,1,7)
vcHS	H	(1,0,1)(1,0,2,24)
	D	(1,0,0)(0,0,2,7)
vcPH	H	(2,0,0)(1,0,1,24)
	D	(1,0,1)(1,0,1,7)
vcHM	H	(1,0,0)(2,0,0,24)
	D	(1,0,1)(2,0,0,7)
vcPK	H	(1,0,1)(1,0,2,24)
	D	(1,0,0)(0,0,2,7)
vcSP	H	(5,0,0)(2,0,1,24)
	D	(2,0,0)(1,0,1,7)

Table 68: Parameters of best SARIMA models of visitor datasets

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
covC	D	(5,0,0)(2,0,1,7)
	W	(4,0,0)(0,0,0,0)
	M	(2,0,1)(0,0,0,0)
covCC	D	(1,0,4)(0,0,0,0)
	W	(2,0,3)(0,0,0,0)
	M	(1,0,0)(0,0,0,0)
covD	D	(5,0,0)(2,0,1,7)
	W	(5,0,0)(0,0,0,0)
	M	(0,0,0)(0,0,0,4)
covCD	D	(2,0,1)(0,0,0,0)
	W	(0,0,0)(0,0,0,0)
	M	(1,0,0)(0,0,0,0)

Table 69: Parameters of best SARIMA models of Covid datasets

Dataset/ Model	(p,d,q)(P,D,Q,M)
ekg1	(5,0,0)(0,0,0,0)
ekg2	(5,0,0)(0,0,0,0)
ekg3	(5,0,0)(0,0,0,0)
ekg4	(1,0,2)(0,0,2,4.1666)
ekg5	(1,0,0)(2,0,2,2.0831)
ekg6	(4,0,0)(0,0,0,4.1672)
ekg7	(2,0,1)(0,0,2,4.1672)
ekg8	(5,0,0)(0,0,0,0)

Table 70: Parameters of best SARIMA models of ekg datasets

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
powerL	D	ValueError (4.5.1)
	W	(3,0,2)(1,0,2,52.18)
	M	(0,0,1)(0,0,1,12)
powerS	D	(4,0,2)(1,0,1,7)
	W	(0,0,0)(0,0,0,4.35)
	M	(0,0,0)(0,0,0,0)
water	D	(4,0,0)(0,0,0,30.5)
	W	(1,0,0)(0,0,0,4.25)
	M	(4,0,0)(0,0,0,0)

Table 71: Parameters of best SARIMA models of power consumption and water temperature datasets

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
teslaC	D	(1,0,0)(1,0,2,7)
	W	(1,0,1)(1,0,0,52.18)
	M	(2,0,0)(0,0,0,0)
teslaH	D	(0,0,2)(1,0,1,7)
	W	(1,0,1)(1,0,0,52.18)
	M	(2,0,0)(0,0,0,0)
teslaL	D	(1,0,0)(1,0,1,7)
	W	(1,0,1)(1,0,0,52.18)
	M	(2,0,0)(0,0,0,0)
teslaO	D	(2,0,0)(1,0,1,7)
	W	(1,0,1)(1,0,0,52.18)
	M	(2,0,0)(0,0,0,0)
teslaV	D	(2,0,3)(2,0,1,7)
	W	(1,0,0)(0,0,1,4.35)
	M	(0,0,1)(0,0,0,12)

Table 72: Parameters of best SARIMA models of Tesla stock datasets

C.2.3. SARIMA grid search

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
vcGB	H	excessive training times (4.5.1)
	D	(1,2,1)(0,1,1,7.0)
vcGH	H	excessive training times (4.5.1)
	D	(1,2,2)(0,1,1,7.0)
vcHH	H	excessive training times (4.5.1)
	D	(5,1,1)(0,0,0,7.0)
vcHS	H	excessive training times (4.5.1)
	D	(0,2,2)(1,1,1,7.0)
vcPH	H	excessive training times (4.5.1)
	D	(5,0,2)(0,0,1,7.0)
vcHM	H	excessive training times (4.5.1)
	D	(1,2,2)(0,1,1,7.0)
vcPK	H	excessive training times (4.5.1)
	D	(2,2,5)(0,1,1,7.0)
vcSP	H	excessive training times (4.5.1)
	D	(1,2,0)(2,0,1,7.0)

Table 73: Parameters of best SARIMA grid search models of visitor datasets

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
covC	D	(0,0,1)(2,0,1,7.0)
	W	(0,2,1)(0,0,0,0)
	M	(2,1,2)(0,0,0,0)
covCC	D	(2,0,5)(0,0,0,0)
	W	(3,0,0)(0,0,0,0)
	M	(2,2,0)(0,0,0,0)
covD	D	(1,1,5)(2,1,2,7.0)
	W	(2,2,2)(0,0,0,0)
	M	(2,0,2)(0,1,0,4.0)
covCD	D	(4,0,0)(0,0,0,0)
	W	(4,2,1)(0,0,0,0)
	M	(3,2,2)(0,0,0,0)

Table 74: Parameters of best SARIMA grid search models of Covid datasets

Dataset/ Model	(p,d,q)(P,D,Q,M)
ekg1	(5,0,5)(0,0,0,0)
ekg2	(5,0,5)(0,0,0,0)
ekg3	(5,0,5)(0,0,0,0)
ekg4	(2,0,5)(0,0,0,4.1666)
ekg5	(1,1,5)(2,1,0,2.0831)
ekg6	(5,0,4)(0,0,0,4.1672)
ekg7	(3,1,5)(2,0,0,4.1672)
ekg8	(3,0,5)(0,0,0,0)

Table 75: Parameters of best SARIMA grid search models of ekg datasets

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
powerL	D	(5,0,5)(0,1,1,7.0)
	W	(2,0,4)(0,1,0,4.35)
	M	(3,2,4)(1,1,1,12.0)
powerS	D	(4,2,5)(0,0,0,7.0)
	W	(5,2,1)(0,1,2,4.35)
	M	(0,2,1)(0,0,0,0)
water	D	(2,2,4)(2,0,2,7.0)
	W	(3,2,3)(2,1,0,4.35)
	M	(0,2,1)(0,0,0,0)

Table 76: Parameters of best SARIMA grid search models of power consumption and water temperature datasets

Dataset/ Model	Res.	(p,d,q)(P,D,Q,M)
teslaC	D	(0,2,1)(2,0,0,7.0)
	W	(3,2,3)(0,1,0,4.35)
	M	(1,2,1)(0,0,0,0)
teslaH	D	(1,1,0)(2,0,1,7.0)
	W	(3,1,2)(0,1,2,4.35)
	M	(1,2,1)(0,0,0,0)
teslaL	D	(1,1,1)(2,0,1,7.0)
	W	(0,2,4)(1,1,0,4.35)
	M	(3,2,4)(0,0,0,0)
teslaO	D	(1,1,1)(1,0,1,7.0)
	W	(2,1,1)(1,1,2,4.35)
	M	(1,2,1)(0,0,0,0)
teslaV	D	(3,2,4)(1,0,2,7.0)
	W	(3,2,2)(0,1,1,4.35)
	M	(3,2,3)(0,1,0,12.0)

Table 77: Parameters of best SARIMA grid search models of Tesla stock datasets

C.2.4. Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend and Seasonal components

Dataset/ Model	Res.	Parameters
vcGB	H	alpha: 0.9972, beta: -0.1331, use_box_cox: False
	D	alpha: 0.7371, beta: None, use_box_cox: False
vcGH	H	alpha: 1.2798, beta: -0.0943, use_box_cox: False
	D	alpha: 0.5719, beta: None, use_box_cox: False
vcHH	H	alpha: 1.1971, beta: -0.2095, use_box_cox: False
	D	alpha: 0.6182, beta: 0.0626, use_box_cox: False
vcHS	H	alpha: 0.9698, beta: -0.2210, use_box_cox: False
	D	alpha: 0.8310, beta: -0.2159, use_box_cox: False
vcPH	H	alpha: 1.4058, beta: -0.3261, use_box_cox: False
	D	alpha: 0.5450, beta: 0.1195, use_box_cox: False
vcHM	H	alpha: 1.0424, beta: -0.1790, use_box_cox: False
	D	alpha: 0.4215, beta: 0.1326, use_box_cox: False
vcPK	H	alpha: 1.1422, beta: -0.2347, use_box_cox: False
	D	alpha: -0.1432, beta: 0.0174, use_box_cox: False
vcSP	H	alpha: 0.8860, beta: -0.1206, use_box_cox: False
	D	alpha: 0.4861, beta: None, use_box_cox: False

Table 78: Parameters of best TBATS models of visitor datasets

Dataset/ Model	Res.	Parameters
covC	D	alpha: 0.3794, beta: 0.0347, use_box_cox: False
	W	alpha: 1.2997, beta: 0.0545, use_box_cox: True
	M	alpha: 0.6751, beta: -0.1780, use_box_cox: True
covCC	D	alpha: 0.8947, beta: 0.5921, use_box_cox: True
	W	alpha: 0.9392, beta: 1.6768, use_box_cox: True
	M	alpha: 1.0602, beta: -0.0105, use_box_cox: True
covD	D	alpha: -0.0537, beta: 0.0608, use_box_cox: False
	W	alpha: 0.2402, beta: 1.7972, use_box_cox: False
	M	alpha: 1.1740, beta: None, use_box_cox: False
covCD	D	alpha: 1.0413, beta: 0.2876, use_box_cox: False
	W	alpha: 0.5109, beta: 1.2467, use_box_cox: False
	M	alpha: 1.1233, beta: None, use_box_cox: False

Table 79: Parameters of best TBATS models of Covid datasets

Dataset/ Model	Parameters
ekg1	alpha: 1.6489, beta: 0.0006, use_box_cox: False
ekg2	alpha: 0.9245, beta: 1.7783, use_box_cox: False
ekg3	alpha: 0.6707, beta: 2.1781, use_box_cox: False
ekg4	alpha: 0.5036, beta: -0.1138, use_box_cox: True
ekg5	alpha: 1.8063, beta: None, use_box_cox: False
ekg6	alpha: 0.5719, beta: -0.1334, use_box_cox: False
ekg7	alpha: 0.7052, beta: -0.1629, use_box_cox: False
ekg8	alpha: 0.7042, beta: -0.1613, use_box_cox: True

Table 80: Parameters of best TBATS models of ekg datasets

Dataset/ Model	Res.	Parameters
powerL	D	alpha: 1.2554, beta: -0.2242, use_box_cox: False
	W	alpha: 0.2223, beta: -0.0276, use_box_cox: False
	M	alpha: -0.0100, beta: None, use_box_cox: True
powerS	D	alpha: 0.7938, beta: -0.1386, use_box_cox: False
	W	alpha: 0.1907, beta: None, use_box_cox: False
	M	alpha: -0.0099, beta: None, use_box_cox: False
water	D	alpha: 0.6615, beta: None, use_box_cox: False
	W	alpha: 0.3281, beta: None, use_box_cox: False
	M	alpha: 1.3003, beta: 0.0152, use_box_cox: False

Table 81: Parameters of best TBATS models of power consumption and water temperature datasets

Dataset/ Model	Res.	Parameters
teslaC	D	alpha: 0.0510, beta: None, use_box_cox: False
	W	alpha: 0.3686, beta: None, use_box_cox: False
	M	alpha: 0.3667, beta: None, use_box_cox: False
teslaH	D	alpha: 0.0514, beta: None, use_box_cox: False
	W	alpha: 0.3683, beta: None, use_box_cox: False
	M	alpha: 0.3675, beta: None, use_box_cox: False
teslaL	D	alpha: 0.0516, beta: None, use_box_cox: False
	W	alpha: 0.3805, beta: None, use_box_cox: False
	M	alpha: 0.3653, beta: None, use_box_cox: False
teslaO	D	alpha: 0.0504, beta: None, use_box_cox: False
	W	alpha: 0.3755, beta: None, use_box_cox: False
	M	alpha: 0.3662, beta: None, use_box_cox: False
teslaV	D	alpha: 0.0436, beta: None, use_box_cox: False
	W	alpha: 0.2784, beta: None, use_box_cox: True
	M	alpha: 0.4329, beta: None, use_box_cox: False

Table 82: Parameters of best TBATS models of Tesla stock datasets

C.2.5. Exponential Smoothing

Dataset/ Model	Res.	Parameters
vcGB	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0
vcGH	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0
vcHH	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0
vcHS	H	trend: None, seasonal: None, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0
vcPH	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0
vcHM	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: add, seasonal: add, seasonality: 7.0
vcPK	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0
vcSP	H	trend: None, seasonal: add, seasonality: 24.0
	D	trend: None, seasonal: add, seasonality: 7.0

Table 83: Parameters of best ES models of visitor datasets

Dataset/ Model	Res.	Parameters
covC	D	trend: add, seasonal: add, seasonality: 7.0
	W	trend: add, seasonal: None, seasonality: 0
	M	trend: add, seasonal: None, seasonality: 0
covCC	D	trend: add, seasonal: None, seasonality: 0
	W	trend: add, seasonal: None, seasonality: 0
	M	trend: None, seasonal: None, seasonality: 0
covD	D	trend: add, seasonal: None, seasonality: 7.0
	W	trend: add, seasonal: None, seasonality: 0
	M	trend: None, seasonal: None, seasonality: 4.0
covCD	D	trend: add, seasonal: None, seasonality: 0
	W	trend: add, seasonal: None, seasonality: 0
	M	trend: None, seasonal: None, seasonality: 0

Table 84: Parameters of best ES models of Covid datasets

Dataset/ Model	Parameters
ekg1	trend: add, seasonal: None, seasonality: 0
ekg2	trend: add, seasonal: None, seasonality: 0
ekg3	trend: add, seasonal: None, seasonality: 0
ekg4	trend: None, seasonal: None, seasonality: 4.1666
ekg5	trend: add, seasonal: add, seasonality: 2.0831
ekg6	trend: None, seasonal: None, seasonality: 4.1672
ekg7	trend: None, seasonal: None, seasonality: 4.1672
ekg8	trend: None, seasonal: None, seasonality: 0

Table 85: Parameters of best ES models of ekg datasets

Dataset/ Model	Res.	Parameters
powerL	D	trend: None, seasonal: mul, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: None, seasonal: None, seasonality: 12.0
powerS	D	trend: None, seasonal: mul, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: add, seasonal: None, seasonality: 0
water	D	trend: None, seasonal: None, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: None, seasonal: None, seasonality: 0

Table 86: Parameters of best ES models of power consumption and water temperature datasets

Dataset/ Model	Res.	Parameters
teslaC	D	trend: None, seasonal: add, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: add, seasonal: None, seasonality: 0
teslaH	D	trend: None, seasonal: add, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: None, seasonal: None, seasonality: 0
teslaL	D	trend: None, seasonal: add, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: None, seasonal: None, seasonality: 0
teslaO	D	trend: None, seasonal: add, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: None, seasonal: None, seasonality: 0
teslaV	D	trend: None, seasonal: add, seasonality: 7.0
	W	trend: None, seasonal: None, seasonality: 4.35
	M	trend: None, seasonal: None, seasonality: 12.0

Table 87: Parameters of best ES models of Tesla stock datasets

C.2.6. Long Short-Term Memory

Dataset/ Model	Res.	Parameters
vcGB	H	lags:24 epochs:50 batch_size:20 hidden_l:40
	D	lags:7 epochs:25 batch_size:50 hidden_l:30
vcGH	H	lags:24 epochs:50 batch_size:20 hidden_l:30
	D	lags:7 epochs:25 batch_size:30 hidden_l:40
vcHH	H	lags:24 epochs:50 batch_size:20 hidden_l:50
	D	lags:7 epochs:25 batch_size:20 hidden_l:30
vcHS	H	lags:24 epochs:50 batch_size:50 hidden_l:50
	D	lags:7 epochs:25 batch_size:30 hidden_l:30
vcPH	H	lags:24 epochs:50 batch_size:20 hidden_l:30
	D	lags:7 epochs:50 batch_size:30 hidden_l:30
vcHM	H	lags:24 epochs:50 batch_size:20 hidden_l:30
	D	lags:7 epochs:25 batch_size:30 hidden_l:40
vcPK	H	lags:24 epochs:50 batch_size:20 hidden_l:40
	D	lags:7 epochs:25 batch_size:20 hidden_l:40
vcSP	H	lags:24 epochs:50 batch_size:20 hidden_l:50
	D	lags:7 epochs:25 batch_size:30 hidden_l:30

Table 88: Parameters of best LSTM models of visitor datasets

Dataset/ Model	Res.	Parameters
covC	D	lags:7 epochs:25 batch_size:50 hidden_l:50
	W	lags:1 epochs:10 batch_size:30 hidden_l:30
	M	lags:1 epochs:10 batch_size:50 hidden_l:30
covCC	D	lags:1 epochs:10 batch_size:50 hidden_l:30
	W	lags:1 epochs:10 batch_size:20 hidden_l:30
	M	lags:1 epochs:25 batch_size:20 hidden_l:30
covD	D	lags:7 epochs:50 batch_size:30 hidden_l:30
	W	lags:1 epochs:10 batch_size:20 hidden_l:50
	M	lags:4 epochs:10 batch_size:50 hidden_l:50
covCD	D	lags:1 epochs:10 batch_size:50 hidden_l:40
	W	lags:1 epochs:10 batch_size:30 hidden_l:50
	M	lags:1 epochs:25 batch_size:20 hidden_l:30

Table 89: Parameters of best LSTM models of Covid datasets

Dataset/ Model	Parameters
ekg1	lags:1 epochs:50 batch_size:30 hidden_l:50
ekg2	lags:1 epochs:50 batch_size:50 hidden_l:30
ekg3	lags:1 epochs:50 batch_size:50 hidden_l:30
ekg4	lags:5 epochs:25 batch_size:50 hidden_l:30
ekg5	lags:3 epochs:50 batch_size:50 hidden_l:50
ekg6	lags:5 epochs:25 batch_size:50 hidden_l:40
ekg7	lags:5 epochs:25 batch_size:20 hidden_l:50
ekg8	lags:1 epochs:50 batch_size:50 hidden_l:30

Table 90: Parameters of best LSTM models of ekg datasets

Dataset/ Model	Res.	Parameters
powerL	D	lags:7 epochs:50 batch_size:30 hidden_l:40
	W	lags:5 epochs:50 batch_size:30 hidden_l:30
	M	lags:12 epochs:50 batch_size:50 hidden_l:40
powerS	D	lags:7 epochs:50 batch_size:20 hidden_l:40
	W	lags:5 epochs:25 batch_size:50 hidden_l:50
	M	lags:1 epochs:50 batch_size:50 hidden_l:50
water	D	lags:7 epochs:25 batch_size:50 hidden_l:50
	W	lags:5 epochs:25 batch_size:50 hidden_l:40
	M	lags:1 epochs:50 batch_size:50 hidden_l:50

Table 91: Parameters of best LSTM models of power consumption and water temperature datasets

Dataset/ Model	Res.	Parameters
teslaC	D	lags:7 epochs:50 batch_size:30 hidden_l:50
	W	lags:5 epochs:10 batch_size:20 hidden_l:50
	M	lags:1 epochs:50 batch_size:50 hidden_l:50
teslaH	D	lags:7 epochs:50 batch_size:30 hidden_l:50
	W	lags:5 epochs:10 batch_size:20 hidden_l:50
	M	lags:1 epochs:50 batch_size:50 hidden_l:40
teslaL	D	lags:7 epochs:50 batch_size:30 hidden_l:50
	W	lags:5 epochs:25 batch_size:30 hidden_l:30
	M	lags:1 epochs:50 batch_size:20 hidden_l:50
teslaO	D	lags:7 epochs:50 batch_size:30 hidden_l:50
	W	lags:5 epochs:50 batch_size:30 hidden_l:40
	M	lags:1 epochs:50 batch_size:50 hidden_l:50
teslaV	D	lags:7 epochs:50 batch_size:20 hidden_l:50
	W	lags:5 epochs:10 batch_size:20 hidden_l:30
	M	Error (4.5.1)

Table 92: Parameters of best LSTM models of Tesla stock datasets

C.2.7. Gated Recurrent Unit

Dataset/ Model	Res.	Parameters
vcGB	H	lags:24 epochs:50 batch_size:20 hidden_l:50
	D	lags:7 epochs:25 batch_size:20 hidden_l:50
vcGH	H	lags:24 epochs:50 batch_size:20 hidden_l:50
	D	lags:7 epochs:25 batch_size:20 hidden_l:40
vcHH	H	lags:24 epochs:50 batch_size:30 hidden_l:50
	D	lags:7 epochs:25 batch_size:30 hidden_l:30
vcHS	H	lags:24 epochs:50 batch_size:20 hidden_l:40
	D	lags:7 epochs:25 batch_size:50 hidden_l:50
vcPH	H	lags:24 epochs:50 batch_size:20 hidden_l:50
	D	lags:7 epochs:50 batch_size:20 hidden_l:30
vcHM	H	lags:24 epochs:50 batch_size:20 hidden_l:50
	D	lags:7 epochs:25 batch_size:20 hidden_l:30
vcPK	H	lags:24 epochs:50 batch_size:20 hidden_l:40
	D	lags:7 epochs:25 batch_size:20 hidden_l:30
vcSP	H	lags:24 epochs:50 batch_size:20 hidden_l:40
	D	lags:7 epochs:25 batch_size:30 hidden_l:30

Table 93: Parameters of best GRU models of visitor datasets

Dataset/ Model	Res.	Parameters
covC	D	lags:7 epochs:50 batch_size:50 hidden_l:50
	W	lags:1 epochs:10 batch_size:30 hidden_l:30
	M	lags:1 epochs:10 batch_size:50 hidden_l:30
covCC	D	lags:1 epochs:50 batch_size:20 hidden_l:40
	W	lags:1 epochs:10 batch_size:30 hidden_l:30
	M	lags:1 epochs:10 batch_size:20 hidden_l:40
covD	D	lags:7 epochs:50 batch_size:30 hidden_l:30
	W	lags:1 epochs:10 batch_size:30 hidden_l:30
	M	lags:4 epochs:10 batch_size:30 hidden_l:40
covCD	D	lags:1 epochs:10 batch_size:50 hidden_l:40
	W	lags:1 epochs:10 batch_size:30 hidden_l:50
	M	lags:1 epochs:10 batch_size:30 hidden_l:30

Table 94: Parameters of best GRU models of Covid datasets

Dataset/ Model	Parameters
ekg1	lags:1 epochs:50 batch_size:20 hidden_l:30
ekg2	lags:1 epochs:50 batch_size:50 hidden_l:30
ekg3	lags:1 epochs:25 batch_size:50 hidden_l:30
ekg4	lags:5 epochs:25 batch_size:50 hidden_l:30
ekg5	lags:3 epochs:50 batch_size:30 hidden_l:40
ekg6	lags:5 epochs:50 batch_size:50 hidden_l:30
ekg7	lags:5 epochs:25 batch_size:20 hidden_l:40
ekg8	lags:1 epochs:50 batch_size:50 hidden_l:30

Table 95: Parameters of best GRU models of ekg datasets

Dataset/ Model	Res.	Parameters
powerL	D	lags:7 epochs:50 batch_size:20 hidden_l:50
	W	lags:5 epochs:50 batch_size:50 hidden_l:30
	M	lags:12 epochs:50 batch_size:20 hidden_l:50
powerS	D	lags:7 epochs:50 batch_size:20 hidden_l:40
	W	lags:5 epochs:25 batch_size:20 hidden_l:50
	M	lags:1 epochs:25 batch_size:20 hidden_l:30
water	D	lags:7 epochs:25 batch_size:20 hidden_l:40
	W	lags:5 epochs:10 batch_size:30 hidden_l:50
	M	lags:1 epochs:50 batch_size:50 hidden_l:30

Table 96: Parameters of best GRU models of power consumption and water temperature datasets

Dataset/ Model	Res.	Parameters
teslaC	D	lags:7 epochs:50 batch_size:30 hidden_l:50
	W	lags:5 epochs:50 batch_size:30 hidden_l:50
	M	lags:1 epochs:50 batch_size:20 hidden_l:40
teslaH	D	lags:7 epochs:50 batch_size:20 hidden_l:40
	W	lags:5 epochs:50 batch_size:50 hidden_l:40
	M	lags:1 epochs:50 batch_size:20 hidden_l:50
teslaL	D	lags:7 epochs:50 batch_size:30 hidden_l:40
	W	lags:5 epochs:50 batch_size:50 hidden_l:40
	M	lags:1 epochs:50 batch_size:20 hidden_l:50
teslaO	D	lags:7 epochs:50 batch_size:30 hidden_l:40
	W	lags:5 epochs:50 batch_size:50 hidden_l:50
	M	lags:1 epochs:50 batch_size:50 hidden_l:40
teslaV	D	lags:7 epochs:50 batch_size:20 hidden_l:50
	W	lags:5 epochs:10 batch_size:50 hidden_l:40
	M	Error (4.5.1)

Table 97: Parameters of best GRU models of Tesla stock datasets

