POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Business Time Series Forecasting through Data Science

Tesi di Laurea Magistrale in
Management Engineering - Ingegneria Gestionale

Author: **Valeria Maranesi**

Student ID: 952904
Advisor: Prof. Carlo Vercellis
Academic Year: 2021-22

Abstract

The advent of Data Age has come. The amount of data produced and managed every day is continuously, exponentially growing in almost every sector. Each organisation or company must deal with more and more data. Data is currently one of the most valuable resources that each company could own, since it can potentially generate many precious business insights.
Technologies are continuously evolving and, with them, the usage of data is constantly improving. Nowadays, we don't simply analyse data but we try to predict them, making machines learn from the past data. We try, in some ways, to "forecast the future", investigating over the possible connections between past and future data.
Here the Data Science finds place: it is able to manipulate large amounts of historical data to obtain these insights. The Forecasting Science is a clear example of technology at the service of the business: it supports companies of different markets in executing several core activities, such as organising the processes, managing the flows of both materials and information and monitoring the business KPIs. All these aspects of the business give to the decision makers precious information about the development of the business; thus, the Forecasting Science represents a way to make conscious, science-based and bias-free decisions.
The objectives of this thesis are to investigate over the most largely used mathematical methods that enable the time series forecasting and to identify the most appropriate ones in terms of predictions' accuracy. The aim is to evaluate them not only in absolute terms, but also in relation to the hyperparameters they assume. Moreover, the aim is to spot any eventual connection between the performances of each model and the main characteristics of the time series such as granularity, seasonality, trend, noise and autocorrelation.
The results clearly show that the choice of the model significantly impacts the accuracy of the forecasting: a good choice of the model is able to generate quite affordable predictions. The main evidences suggest the existence of a model that usually outperforms all the other algorithms. Moreover, the choice of the hyperparameters that fit each model strongly affects the performances: great attention should be put on the tuning process. At the same time, it is not possible to find any high correlation between the character-

istics of the time series and the optimal model identified. Thus, a big effort should be put on the automatization of the whole process of model testing to try each (time series - model - hyperparmeters) combination and identify the optimal model tuned for each dataset. Finally, results clearly evidence the possibility to obtain different conclusions - and therefore to make different decisions - depending on the choice of the accuracy metric.

Keywords: Time Series, Statistical Models, Machine Learning, Data Science, Sales Forecasting, Business Insights

Abstract in lingua italiana

Viviamo oggi nell'era dei dati. La quantità di dati generata ogni giorno cresce esponenzialmente. Ogni impresa, a prescindere dal settore di appartenenza, si trova quotidianemente a gestire un ammontare sempre crescente di dati. I dati sono attualmente tra le risorse più preziose che ogni azienda possa possedere, poiché forniscono informazioni di grande valore per il business.
Le tecnologie digitali sono in continua evoluzione e, con loro, anche l'utilizzo dei dati. Oggi il mercato non si limita all'analisi degli stessi, ma è in grado di predire i dati futuri. In un certo senso, l'obiettivo oggi è "predire il futuro", analizzando le possibili connessioni presenti nei dati storici.
Qui entra in gioco la Data Science, che permette di ottenere informazione sul futuro analizzando grandi quantità di dati relativi a serie storiche. La scienza delle previsioni è un chiaro esempio di tecnologia al servizio del business: supporta aziende di diversi settori nell'esecuzione di fondamentali attività quali l'organizzazione dei processi, la gestione dei flussi di materiali e informazioni e il monitoraggio dei principali indicatori di business. Tali aspetti legati al business forniscono una serie di preziose informazioni a coloro che hanno un ruolo decisionale nell'organizzazione. Dunque, la scienza delle previsioni fornisce un metodo per prendere decisioni consapevoli e basate su reali evidenze.
L'obiettivo di questa tesi è analizzare i più diffusi modelli matematici per la previsione di serie storiche ed individuare i più appropriati al perseguimento di questo scopo. I modelli vengono valutati rispetto alla loro accuratezza. Inoltre, essi non vengono valutati solo in termini assoluti, ma anche relativamente ai parametri che li regolano. L'obiettivo è anche quello di legare l'accuratezza dei vari modelli in relazione alle caratteristiche della serie storica, quali la granularità, la stagionalità, il trend, la variabilità e l'autocorrelazione. Infine, viene analizzato l'impatto della scelta della metrica sui risultati.
Questi ultimi mostrano chiaramente l'impatto della scelta del modello sull'accuratezza delle previsioni: una buona scelta dell'algoritmo genera previsioni sufficientemente affidabili. Generalmente, una classe di modelli sembra performare meglio delle altre. Inoltre, la scelta degli iperparametri impatta significativamente sulla perfromance del modello: dunque, è importante dedicare risorse al processo di tuning degli iperparametri. Allo

stesso tempo, è molto difficile stabilire una connessione tra le caratteristiche del dataset e il modello più accurato. Di conseguenza, non potendo selezionare a priori il modello ottimale, la scelta migliore sembra essere l'automatizzazione dell'intero processo di sperimentazione sui modelli, con l'obiettivo di testare tutte le possibili configurazioni migliorando le performances. Per concludere, è evidente l'impatto che la scelta della metrica ha sui risultati degli esperimenti. Infatti, è possibile ottenere risultati diversi - e dunque prendere decisioni diverse - sulla base della scelta dell'indicatore o di una combinazione di indicatori.

Parole chiave: Serie Storiche, Modelli Statistici, Machine Learning, Data Science, Previsione delle Vendite, Visione di Business

Contents

Introduction

Business Objective

The forecasting process is one of the main fields of data science and it helps companies of different markets in executing several planning activities. It is a clear example of technology at the service of the business: the forecasting techniques help corporations in organising the processes, structuring the production line, and managing the flows of both materials and information.

From an operational viewpoint, making business forecasts of many different variables drives the organization of scarce resources such as materials, information and goods, the capacity planning - both in terms of budgets and in terms of time -, and the monitoring of business KPIs and performance trends. Therefore, as all these aspects of the business give many insights to the decision makers about the business development, the data science can be considered as the first supporter of each company management since it represents a way to make conscious, science-based and bias-free decisions. Therefore, if the result of the analysis is a high-quality prediction, then it enables the corporation to be both more efficient in managing resources and reducing costs and more effective in offering a better service, at the same time.

Since the quality of the predictions is fundamental, the forecasting techniques should be analysed in order to identify the best one – or the best combination of them - in each specific case use. Furthermore, a set of metrics to evaluate the performances of the different methods must be identified.

The aim of this thesis is, in some ways, to "forecast the future", to make predictions about the trend of some business-related time series and to investigate the mathematical methods that enable this powerful instrument. The ultimate goal is to support companies in the decision making process, giving precious insights and reporting results obtained through the scientific method.

Methodology Overview

A first research activity is performed in order to understand the theoretical background and the state of the art of the research in the time series forecasting field. The main key points are reported in Chapter 1, together with the business objective and the final aim of this research work. Then, we define the perimeter of the research work setting the experiments in terms of datasets, models, hyperparameter spaces and tuning and accuracy metrics in Chapter 2. The study has the aim to observe the behaviour of the traditional forecasting methods on the datasets under exam and to assess their performances. After that, also machine learning algorithms and experimental methods are tested.
The following step is the data exploration phase in which the time series under analysis are studied in order to spot their main peculiarities in statistical terms. This phase has the final aim to link the evidences of the experiments with the characteristics of the times series and it is reported in Chapter 3.
Then, in Chapter 3.2.2, we report the main results, evidences and considerations that emerge from our analysis.
Finally, a series of possible future developments is reported in Chapter 4. It is aimed at reporting the points that are still open and the ones that can be touched in the future analysis in order to go in depth with this topic and improve the core points of this thesis.

1 | Theoretical Background and Business Objectives

This chapter is dedicated to the description of the theoretical background that lies under the theme of the time series forecasting. Indeed, the theoretical fundamentals of time series and of the currently most used models, of the parameters that optimize each algorithm and of the metrics that evaluate every method are provided. Moreover, a brief description of the characteristics of the forecasting models' mathematical properties, of their performance and of their current use cases is reported. Finally, a range of real-life examples and business applications is presented.
The theoretical background of time series, forecasting models and accuracy metrics comes from a careful examination and study of the literature.

1.1. Time Series

In mathematics, a **time series** is a series of data points indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. As it is stated in [7], time series are used in statistics, signal processing, pattern recognition, econometrics, mathematical finance, weather forecasting, control engineering, astronomy, communications engineering, and, more in general, in any domain of applied science and engineering which involves temporal measurements.

1.1.1. Mathematical Approach to Time Series

"A **time series** is a sequence $\{y_t\}$ of values assumed by a quantity under interest indexed in time order t" [7]. In general, as it is largely explained in [44], time series can be classified in the following two categories.

- Discrete time series present time values that belong to a discrete set; most commonly, a time series is a discrete one and it is taken at successive equally spaced points in time.

- Continuous time series assume values that are gathered in a continuous time interval.

In this thesis, the focus will be posed on discrete time series only and the time periods will be natural uniform time intervals like hours, days, weeks, months and years.
From a mathematical viewpoint, the stochastic nature of the values taken by the time series is evident. Indeed, the observations $\{y_t\}$ can be considered as the realizations of the random variable Y_t in the various time instants t. Therefore, a time series model $\{y_t\}$ will include the assignment of the probability distribution of the random variables sequence $\{Y_t\}$, which can be considered by definition as a **stochastic process**.
In most of the models, the first $E[Y_t]$ and second-order moments $E[Y_{t+h}Y_t]$ are among the most important indicators to evaluate the model.

1.1.2. Main Components of the Time Series

The time series are usually characterised by three main properties, called **components**: trend, seasonality and random noise. We report the main ones, as explained by literature in [44], [23].

- **Trend.** The trend components is responsible for the long-time behaviour of the time series. Usually, this component is indicated by M_t, and it is approximated by simple functions such as the linear, polynomial (usually of second or third order), exponential or logarithmic ones.
  The mathematical object which is responsible for the modeling of the trend component is called moving average $m_t(h)$ with parameters h and t. It is defined as the arithmetic average of h successive values that the time series $\{y_t\}$ assumes. In this case, it is natural to suppose that the index t is one of the indexes of the h observations considered. In general, this t is the central point of the h observations taken into account. Equation (1.1) shows the mathematical expression of the simple centered moving average when h is odd:

$$m_t(h) = \frac{y_{t+(h-1)/2} + y_{t+(h-1)/2-1} + ... + y_{t-(h-1)/2}}{h}. \tag{1.1}$$

  If h is even, the expression above becomes:

$$m_t(h) = \frac{y_{t+h/2} + y_{t+h/2-1} + ... + y_{t-h/2+1}}{2h} + \frac{y_{t+h/2-1} + y_{t+h/2-2} + ... + y_{t-h/2}}{2h}. \tag{1.2}$$

  When h is equal to the whole temporal horizon that is considered during the time series analysis L, then the trend component of the time series is calculated since the

moving average captures only the long-term variations.
Moreover, the moving average can be used to generate future predictions centering m_t in the last of the h observations. A more accurate estimation can be computed attributing different weights w_i to the different observations and calculating in this way the so-called weighted moving average.

- **Seasonality.** The seasonality component is responsible for the short-term fluctuations, which often present a regular frequency during the time span considered. The seasonality component is denoted by Q_t and it is mathematically represented by a periodic function.

- **Random noise.** The random noise is a component which is able to model the irregular and unexpected fluctuations that a random variable usually has and that any other component can explain. In mathematical terms, the random noise is the time series $\{\varepsilon_t\}$, which is obtained removing from the original time series $\{y_t\}$ the trend and seasonality component. It represents the so-called white noise process, which is equivalent to a sequence of independent random variables which are normally distributed with mean equal to 0 and constant variance: $\mathrm{E}_t \sim N(0; \sigma^2)$.

The main components of time series are graphically represented in Figure 1.1, which shows the evolution of the Google job search market during time[1].

[1] https://www.researchgate.net/

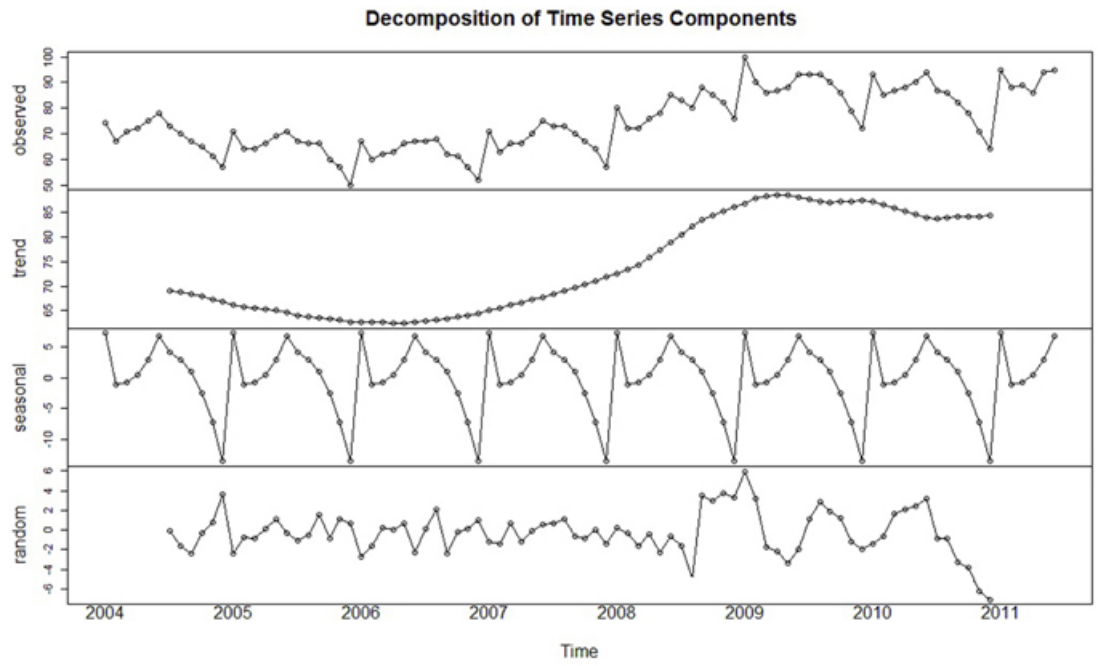


Figure 1.1: Original time series of the Google job search market decomposed into trend, seasonal, and irregular components

The above-mentioned components easily find a practical application in the business field and in many different markets. In general, the trend component can be stationary, increasing or decreasing and represents the behaviour of the market in the long term. For example, after Covid-19 spread in 2020, the trend component of the time series of the biotechnology companies' stocks has been increasing much faster.
On the other hand, the seasonality component is determined by the natural cycles or the seasonality of the product (and consequently of their demands).
Sometimes, a fourth component can be added to the model, and it represents the oscillations that are due to the economic cycle. In all the other cases, the economic cycle is considered as impacting in the medium-long term, and therefore it is included in the trend component.
In this use case, as it is appreciable by the graphic, the trend component increases in the early 2000s, while it is more stable from 2009 on. Moreover, after 2009 the noise increases, maybe because of the economic crisis of that year.

In mathematical terms, each time series can be expressed as a combination of these components:

$$Y_t = g(M_t, Q_t, \varepsilon_t) \tag{1.3}$$

where g is a function that should be selected in every specific case. In most of the cases, g expresses an additive or multiplicative relationship among the components of the time series. However, some models such as exponential smoothing models are obtained as the combination of additive and multiplicative components.
A multiplicative model is the most largely used and it has the following shape:

$$Y_t = M_t \times Q_t \times \varepsilon_t. \tag{1.4}$$

Otherwise, the additive model

$$Y_t = M_t + Q_t + \varepsilon_t. \tag{1.5}$$

can be considered.
In order to perform the time series analysis and forecasting, it is useful to decompose the time series in its components. Indeed, time series decomposition provides a useful abstract model for thinking about time series generally and for better understanding problems during time series analysis and forecasting.
In general, the trend component M_t can be approximated as the moving average $m_t(L)$ centered in $h = L$. As usually the decomposition model is multiplicative, then the removal of the trend component simply consist in the division by $M_t \approx m_t(L)$:

$$B_t = Q_t\varepsilon_t = \frac{Y_t}{M_t} \approx \frac{Y_t}{m_t(L)} \tag{1.6}$$

. Alternatively, when the model is additive, then the removal of the trend component is performed computing successive differences between adjacent values of the time series such as:

$$D_t = Y_t - Y_{t-1} \tag{1.7}$$

. Similarly, it is possible to deseasonalise the time series dividing it by the seasonality component. The seasonality component $Q_{l(t)}$ is a function of $l(t)$, the set of time points that belong to the frequency period, observed plotting the time series subtracted by its trend component. The deseasonalised time series together with the linear trend is calculated as the following expression:

$$\frac{y_t}{Q_{l(t)}} = M_t\varepsilon_t \tag{1.8}$$

. Made these considerations, it is possible to make predictions for the future periods that are based on the decomposition of the time series. Indeed, the trend component is projected in the future period, and then the seasonality component is added to this

model, extending the periodic oscillations identified.

1.2. Traditional Forecasting Models

The most traditional and largely used methods for the time series analysis and forecasting are the **statistical methods**, which are based on a strong theoretical basis explained in [44] and are, at the same time, empirically proven to be particularly efficient in the economic context. In general, traditional time series models account for the fact that data points taken over time may have an internal structure that should be accounted for. These methods take into account as independent variable the one individuated by the points in time only: no other attributes are - in general - taken into consideration when producing an analysis through these models. This is the characteristic that distinguish between the traditional (also called statistical) models and other kinds of algorithms.
The traditional forecasting models for time series include the following ones.

- **Exponential smoothing models** are based on the decomposition of the time series in its main components, as explained in Paragraph 1.1.2, and they are deeply described in [12]
- **Autoregressive models** are based on the process of identification of patterns and correlations between different observations, homogeneously distributed on the time span considered.

1.2.1. Exponential Smoothing Models

"The exponential smoothing class includes simple, single parameter models that predict the future as a linear combination of a previous value and a current shock. Exponential smoothing assumes that a series extends infinitely into the past, and that this influence of past on future decays smoothly and exponentially. The smooth rate of decay is expressed by a smoothing constant."[44]
The most simple model of this class is called **simple exponential smoothing** (SES) or brown model, and it relies on the smoothed mean s_t, which is computed as the average value of the observations of the time series until time t. It is defined through the recursive expressions:

$$s_t = \alpha y_t + (1 - \alpha) s_{t-1} s_1 = y_1, \tag{1.9}$$

where $\alpha \in [0, 1]$ is the parameter that regulates the relative importance of the recent value y_t with respect to the smoothed mean of the previous values. The expression (1.9) can be recursively applied to obtain the following relationship between the forecast of the future

period and the observations in the past:

$$f_{t+1} = \alpha[y_t + (1-\alpha)y_{t-1} + ... + (1-\alpha)^{t-2}y_2] + (1-\alpha)^{t-1}y_1. \tag{1.10}$$

From equation (1.10) it is evident that the prediction is expressed as a linear combination of the past observations of the time series, with weights exponentially decreasing as the time goes back. Moreover, the parameter α indicates the relative importance that is given to the past observations with respect to the most recent ones: if $\alpha \simeq 0$, the model gives almost the same importance to all the past observations; instead, if $\alpha \simeq 1$ then the model is more responsive and it weights more the most recent values of the time series.

It is possible to extend the simple model to those time series which present a strong trend component. It is called **Holt exponential smoothing** model. This model is based on two components: the smoothed mean s_t and the linear smoothing trend m_t, which approximates the trend component M_t. The mathematical expression of the smoothed mean is adjusted in order to take into account the smoothed trend too:

$$s_t = \alpha y_t + (1-\alpha)(s_{t-1} + m_{t-1}), \tag{1.11}$$

and, at the same time, the smoothed trend is defined as:

$$m_t = \beta(s_t - s_{t-1}) + (1-\beta)m_{t-1}. \tag{1.12}$$

Similarly, also $\beta \in [0, 1]$ modulates the importance of the most recent value of the trend and assumes the same mathematical value that α has.
The prediction for period $t+1$ is made by the additive composition of these two components:

$$f_{t+1} = s_t + m_t. \tag{1.13}$$

The exponential smoothing model can be extended when it presents a seasonality component. The model is called **Winters** and it presents an additional components called smoothed seasonal index q_t, that approximates the multiplicative seasonality component Q_t. From a mathematical point of view, the model assumes the following shape:

$$s_t = \alpha \frac{y_t}{q_t - L} + (1-\alpha)(s_{t-1} + m_{t-1}), \tag{1.14}$$

$$m_t = \beta(s_t - s_{t-1}) + (1-\beta)m_{t-1}, \tag{1.15}$$

$$q_t = \gamma\frac{y_t}{s_t} + (1-\gamma)q_{t-L}, \tag{1.16}$$

where L is the number of periods that are considered and $\gamma \in [0,1]$ is the parameters that weights the relative importance of the most recent values of seasonality $\frac{y_t}{s_t}$ with respect to the ones of the past.
Finally, the prediction for the future period $t+1$ is defined as:

$$f_{t+1} = (s_t + m_t)q_{t-L+1}. \tag{1.17}$$

For every exponential smoothing method we then need to choose the value for the smoothing parameters α, β, γ. In general, the most robust and objective way to choose the values of the parameters included in any exponential smoothing method is to estimate them from the observed data, by evaluating some specific metrics. The most common and efficient way consists in the minimization of the **Sum of Squared Errors (SSE)**. The errors are specified as $e_t = y_t - \hat{y}_{t|t-1}$ for $t = 1, ..., T$. Therefore, the metric that must be minimized is defined as:

$$SSE = \sum_{t=1}^{T}(y_t - \hat{y}_{t|t-1})^2 = \sum_{t=1}^{T} e_t^2 \tag{1.18}$$

To solve this optimization problem it is necessary to use linear regression techniques or non-linear optimization tools.

1.2.2. Autoregressive Models

As stated in [44], "**Autoregressive (AR)** methods are based on the idea of identifying possible relationships between the observations of a time series analysing the autocorrelation that exists among observations taken at different time moments". The values of the time series are usually separated by a fixed time interval in autoregressive models. Moreover, the output variable of an AR model depends linearly on its own previous values and on a stochastic term; thus the model is in the form of a stochastic difference equation, as explained in [19]. The idea behind the AR models is to analyse the autocorrelation through the backshift B, as it is clear from:

$$B^h Y_t = T_{t-h}, \tag{1.19}$$

with $t > h$. We analyse the correlation between variables Y_t and $B^h Y_t$. If there is for example a seasonality component with period L, then it is expected that the time series Y_t and $B^L Y_t = Y_{t-L}$ are strongly positively correlated.
In general, the AR model is particularly efficient if it is applied to a stationary time series; therefore, it is always useful to decompose the time series before applying the AR model. In order to make the mean stationary, the differencing operator of order h should be applied between successive terms of the time series:

$$\nabla^h Y_t = Y_t - Y_{t-h}, \tag{1.20}$$

with $t > h$.
In general, an autoregressive model of order p creates a linear regression relationship between the original time series and the one created through the use of the backshift operator B, until order p:

$$Y_t = \gamma + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \varepsilon_t. \tag{1.21}$$

The term ε_t is a random variable with normal distribution and zero mean and it represents the noise. All the parameters that Equation (1.21) presents must be determined minimizing the sum of squared errors.
Made all these considerations, the prediction for period t through an AR model has the mathematical shape that follows:

$$f_{t+1} = \gamma + \phi_1 Y_t + \phi_2 Y_{t-1} + ... + \phi_p Y_{t-p+1}. \tag{1.22}$$

Another model that is classified as autoregressive is the **moving average** model (MA). It is based on the objective to establish a linear regression between the original time series and the time series composed by the prediction errors in the previous periods.
Starting from this model, the most general **autoregressive integrated moving average** models (ARIMA) integrates the two previously described approaches. Indeed, when the time series Y_t is non-stationary, then it is possible to apply a model whose variables are the prediction errors $z_{t-1}, ..., z_{t-q}$ of the past q time periods and $A_1, .., A_t$. In general, the term A_t is obtained differencing the term $Y_t d$ times: $A_t = \nabla^d Y_t$. The general shape of the ARIMA model is:

$$A_t = \gamma + \varepsilon_t + \phi_1 A_{t-1} + \phi_2 A_{t-2} + ... + \phi_p A_{t-p} - \theta_1 z_{t-1} - \theta_2 z_{t-2} - ... - \theta_q z_{t-q}. \tag{1.23}$$

Also in this case, the search for the optimal parameters is performed by minimizing the sum of squared errors.
The prediction for period t is then formulated as:

$$f_{t+1} = \gamma + \phi_1 A_t + \phi_2 A_{t-1} + ... + \phi_p A_{t-p+1} - \theta_1 z_t - \theta_2 z_{t-1} - ... - \theta_q z_{t-q+1}. \quad (1.24)$$

If the time series includes a component of seasonality, then it is often also appropriate to develop an ARIMA model for it. This model is usually denoted by ARIMA (P, D, Q) to distinguish its order from the corresponding order (p, d, q) of the non-seasonal component. In particular, in this model the time series is obtained by successive differences of order DL, so using multiples of the seasonality cycle.

To establish the eventual existence of a seasonality component, autocorrelation (ACF) and partial autocorrelation (PACF) diagrams should be created. The autocorrelation coefficient for lag h is defined as:

$$\text{ACF}_h = \text{corr}(Y_t, Y_{t-h}) \quad (1.25)$$

and indicates the degree of correlation between the values of the time series Y_t and the values of the series Y_{t-h}. Instead, the partial autocorrelation coefficient expresses the correlation between Y_t and Y_{t-h} that is not accounted for by shorter lags; it is defined as:

$$\text{PACF}_h = \text{corr}(Y_t, Y_{t-h} | Y_{t-1}, Y_{t-2}, ..., Y_{t-h+1}). \quad (1.26)$$

In general, the ACF and PACF can be observed in order to establish the most appropriate parameters p and q of the ARIMA model. Anyway, sometimes it is not so easy to find their optimal values by the simple observation of these two graphics. The so called auto-ARIMA model is able to calculate them using an empirical machine learning model.

1.3. Machine Learning Models: Regression

The most largely used machine learning model for time series forecasting exploits supervised learning, as explained in [22], [24]. Supervised learning algorithms aim at estimating the value of an output variable y through some input variables X, and through the construction of an algorithm to learn the mapping function from the input to the output.
Given a sequence of values of a time series dataset, it is possible to restructure the data to look like a supervised learning problem, in particular like a regression problem. Indeed,

the objective of time series forecasting is to predict the future values of a specific variable, not to split the observations in groups as the classification does. We can do this by using previous time steps as input variables and use the next time step as the output variable. The purpose of the regression models is to find a functional relationship between the attributes X (or a subset of them) and the target variable y. In mathematical terms, the regression models find out the existence of a function f that relates the n attributes and the dependent variable y, as explained in Equation 1.27:

$$y = f(X_1, X_2, ..., X_n). \tag{1.27}$$

The function f can assume many different shapes, such as the linear, quadratic and exponential, but also more complex ones. These functions identify the algorithms that will then able to predict the values assumed by the target.
In general, when a regression model is developed, all the records of the train dataset are taken in an unspecified order, since ex ante all the observations equally influence the target variable. Anyway, in the specific case of the time series forecasting, the past values of the variables can't be considered randomly, but the temporal sequence is essential. Therefore, in time series analysis and forecasting, the order between the observations is preserved, and it must continue to be preserved when using the same dataset to train a supervised model.

1.3.1. Sliding Window for Time Series

The use of previous time steps to predict the next time step is called the **sliding window** method or simply window method, as it is explained in [22], [30]. In statistics and time series analysis, it is also called lag method. The classic name comes from the number of past time steps, which is called the window width. In general, the width sliding window can be increased to include different numbers previous time steps.
It is useful to say that a time series dataset is prepared in such a way that any of the standard linear and nonlinear machine learning algorithms may be applied, as long as the order of the rows is preserved. The following schema explains the basic idea behind the window method (each x_i represents an observation of the time series):

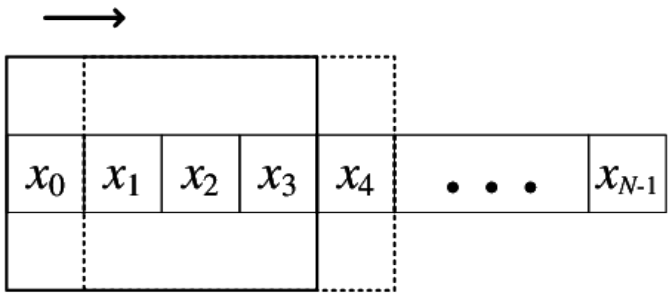


Figure 1.2: Schematic representation of the sliding window for time series forecasting

The sliding window approach can be used on a time series that has more than one value, the so-called multivariate time series. The approach is exactly the same as the one that treats with univariate time series.
The number of future time steps to be forecasted is important. It is traditional to use different names for the problem depending on the number of time-steps to forecast:

- One-Step Forecast: when the next time step $(t + 1)$ is predicted.
- Multi-Step Forecast: when two or more future time steps want to be predicted.

The approach to multi-step forecast consists in the iteration of the one-step forecast. It can be decided to re-tune the parameters and re-train the model at each step or to keep the same model with the same parameters at each iteration.

1.4. The Prophet Forecasting Model

Prophet is a procedure for forecasting time series data which has been developed by the core data science team of Facebook. It is an additive model that is able to handle non-linear trends, yearly, weekly and daily seasonality and holiday effects. It works optimally with data that present seasonal effects and that are distributed on a long time span. Section 1.4 is aimed at presenting the main characteristics of the Facebook Prophet model, deeply explained in [5], [41], [42], [35].

1.4.1. The Model

In general, the standard mathematical shape of a **Prophet model** is similar to a Generalised Additive Model (GAM), a class of regression models with potentially non-linear smoothers applied to the regressors. Therefore, in some sense the Prophet model can be interpreted as the integration of statistical (traditional) models and ragressors (machine learning models). In general, the Prophet model is executed by framing the forecasting

problem as a curve-fitting exercise, which is strongly different from time series models that account for the temporal dependence structure in the data.

The main components of the time series are combined as follows:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t. \tag{1.28}$$

In addition to the traditional trend or growth component ($g(t)$) and seasonality component ($s(t)$), we add in this case the holiday effect $h(t)$, which potentially has irregular schedules over months, weeks and days.
Going more in depth, in general there is nonlinear growth, which is usually modeled through the logistic growth model as is clear from the following equation:

$$g(t) = \frac{C}{1 + e^{-k(t-m)}}, \tag{1.29}$$

where C is the capacity, k the growth rate and m an offset parameter.
Then, for what regards the seasonality, it is generally modeled as a periodic function, whose effects can be approximated with the following formula:

$$s(t) = \sum : i = 1^N (a_n cos(\frac{2\pi nt}{P} + b_n sin(\frac{2\pi nt}{p})), \tag{1.30}$$

which is the standard expression of a Fourier series of periodicity P. Usually, the choice of the best parameters that fit the model can be automated using a model selection technique such as AIC (Akaike Information Criterion), which by calculating and comparing the AIC scores of several possible models, chooses the one that is the best fit for the data.
Holidays and events provide some predictable shocks to many business time series and often do not follow a periodic pattern, so their effects are not well modeled by a smooth cycle. In general, the impact of a particular holiday on the time series is often similar year after year, so it is important to incorporate it into the forecast. Therefore, a custom list of past and future holidays and events is provided and it can be identified by the event or holiday's unique name, as shown in Table 1.1.

Holiday	Country	Year	Date
Christmas	*	2020	25 Dec 2020
Christmas	*	2021	25 Dec 2021
Christmas	*	2022	25 Dec 2022
Christmas	*	2023	25 Dec 2023
Easter	Italy	2020	12 Apr 2020
Easter	Italy	2021	4 Apr 2021
Easter	Italy	2022	17 Apr 2022
Easter	Italy	2023	9 Apr 2023

Table 1.1: Example list of holidays in Italy. The country and the year are specified because holidays may occur on different days in different countries and different years.

For a given forecasting problem we use both the global set of holidays and the country-specific ones.
From a mathematical viewpoint, it is important to say that, to include the effect of holidays into the model, they are assumed as independent. For each holiday i, D_i presents each past and future date for a holiday. Moreover, an indicator function represents whether time t is during holiday i, and it assigns to each holiday a parameter κ_i which is the corresponding change in the forecast, as it is clear from Equation 1.31 and Equation 1.32:

$$Z(t) = [\mathbb{1}(t \in D_1), ..., \mathbb{1}(t \in D_L)] \tag{1.31}$$

and taking

$$h(t) = Z(t)\kappa. \tag{1.32}$$

Thanks to the simplicity of the model, the fitting process is very fast and it has easily interpretable parameters that can be changed by the analyst to impose assumptions on the forecast. Moreover, it is easy to extend the model to include new components.

1.5. Time Series Analysis and Forecasting Process

The **time series analysis** comprises methods for analysing time series data in order to extract meaningful statistics and other peculiar characteristics of the series itself. The

entire process is well explained in [6]. In particular, the first step consists in the decomposition of the time series identifying the components in the most precise way possible (see paragraph 1.1.2). Then, an evaluation of the volatility of the datum under analysis is made, in order to estimate the expected accuracy of the predictions.
The **time series forecasting** consists in building models through the observation of historical data, in using them to make predictions for future time periods and to drive future strategic decision-making. An important distinction in forecasting is that, at the time of the work, the future outcome is completely unavailable. Moreover, it can only be estimated through a careful analysis and some evidence-based priors.
The process aimed at forecasting time series is a standard process based on some standard steps and it basically has the same structure for all the possible mathematical models that can be chosen. In general terms, the forecasting process is structured as Figure 1.3 shows.

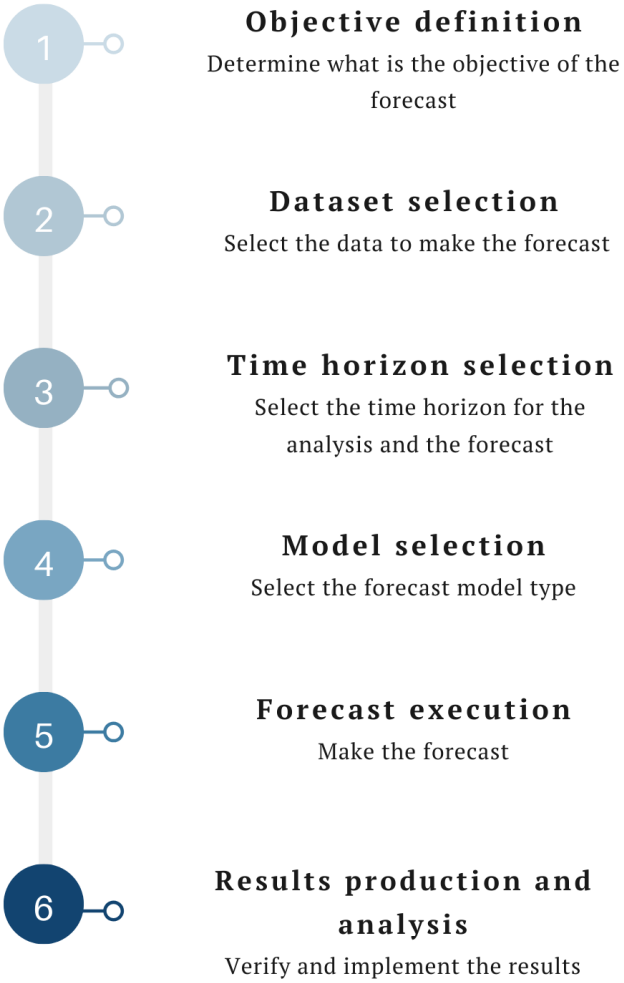


Figure 1.3: Main steps of the quantitative forecasting process

The first step of the roadmap highlights the importance of the definition of the business objective the forecasting wants to reach. It should be clear to identify the most significant variables that are going to be analysed and the one that is going to be predicted. All the following steps must be executed keeping the final objective clear in mind.
The final step regards the measurement and the evaluation of the results. Since the scope of this thesis comprises the evaluation of the forecasting accuracy that the various methods present, it is also necessary to estimate it, referring to the real data than compose the prediction. On the other hand, it is also necessary to elaborate a set of metrics that measure the confidence level the forecasting method has "ex ante", without knowing the real data in advance as it happens in a real-world situation when a forecasting on future

data is made. This is the reason why, in Section 1.6 both some "ex post" metrics and some "ex ante" ones will be presented.

1.6. Forecasting Accuracy Metrics

This section is based on the literature of [1] and [44]. The accuracy metrics are able to measure the accuracy of the predictions generated using a time series model. Accuracy is a measure of observational error. Accuracy is how close a given set of observations are to their true values. The two main objectives of the accuracy measures are reported here.

- The first one is to assess alternative models and to determine the parameters that are assigned to the model. To determine which is the most appropriate model, each model is applied to the past data and the metrics are evaluated on this dataset. The model that presents the minimum total error is selected.
- The second aim is to assess the accuracy of the predictions that have been made through the comparison of the forecasts with the real observed data. The periodic evaluation of the accuracy of predictions makes it possible to determine if a model is accurate or if a revision is required.

The first category of measures is the one of the **distortion indices**, which are used to discriminate among models using signed mean errors. In general, the **prediction error** is defined as the difference between forecasts f_t and sales y_t:

$$e_t = y_t - f_t. \tag{1.33}$$

In a very similar way, the **percentage prediction error** is defined as:

$$e_t^P = \frac{y_t - f_t}{y_t} \times 100. \tag{1.34}$$

This second metric is independent of the scale on which the observations are measured and is therefore a more reliable measure, especially for comparing the accuracy of different time series.
Then, it could be necessary to estimate the overall differences between the forecasts and the sales, both in absolute and average terms. It is called **Forecast Bias** or **Mean Signed Deviation**:

$$Forecast\ Bias = \sum_{t=1}^{n}(y_t - f_t) \tag{1.35}$$

$$Mean\ Signed\ Deviation = \frac{1}{n}\sum_{t=1}^{n}(y_t - f_t) \tag{1.36}$$

Forecast bias is the difference between forecast and sales, and it captures the overall bias that the forecast has. If the forecast over-estimates sales, the forecast bias is considered negative. If the forecast under-estimates sales, the forecast bias is considered positive.

Then, it can be useful to assess how large the forecast error is on average in absolute terms. These metrics are called **dispersion indices**. The **Mean Absolute Error (MAE)** is the metric based on this idea, and it is defined as:

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - f_t| \tag{1.37}$$

The **Mean Absolute Percentage Error (MAPE)** has the same objective of the MAD metric, but it expresses the forecast error in relation to sales volume:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{|y_t - f_t|}{y_t} \tag{1.38}$$

Basically, it tells you by how many percentage points your forecasts are off, on average. This is probably the single most commonly used forecasting metric in demand planning. Other largely used dispersion indexes are the **Mean Square Error (MSE)**, whose shape is the following:

$$MSE = \frac{\sum_{t=1}^{n}(y_t - f_t)^2}{n} \tag{1.39}$$

, which is the same as the sum of square errors SSE but divided by n, and the **Root Mean Square Error**, calculated as:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(y_t - f_t)^2}{n}} \tag{1.40}$$

.

It is important to highlight that, despite their names, the forecasting accuracy indexes measure forecast error, meaning that 0 or 0% is the target and larger numbers indicate a larger error.

In general, among the models that show null (or almost null) distortion, the model with the least dispersion is usually preferred.

Table 1.2 reports the above described measures and their mathematical formulations.

Forecast Accuracy Measure	Formula
Prediction Error	$e_t = y_t - f_t$
Percentage Prediction Error	$e_t^P = \frac{y_t - f_t}{y_t} \times 100$
Forecast Bias	$Forecast\ bias = \sum_{t=1}^{n}(y_t - f_t)$
Mean Signed Deviation %	$Forecast\ bias\ \% = \frac{1}{n}\sum_{t=1}^{n}(y_t - f_t)$
Mean Absolute Error	$MAE = \frac{1}{n}\sum_{t=1}^{n}\lvert y_t - f_t\rvert$
Mean Absolute Percentage Error	$MAPE = \frac{100}{n}\sum_{t=1}^{n}\frac{\lvert y_t - f_t\rvert}{y_t}$
Mean Square Error	$MSE = \frac{\sum_{t=1}^{n}(y_t - f_t)^2}{n}$
Root Mean Square Error	$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(y_t - f_t)^2}{n}}$

Table 1.2: Main accuracy metrics and related mathematical formulations

1.7. Business Objectives, Use Cases and Applications

1.7.1. Data Science at the Service of the Business

The **global Big Data Analytics** market size is expected to be worth USD 103 billion in 2027. The market size was USD 56 billion in 2020 and USD 64 billion in 2021 [2]. Further, the market is estimated to grow at a **CAGR of 13.2%** during the 2021-2028 period [3]. This growth is evident from Figure 1.4. Indeed, Big Data Analytics examines the unstructured and structured record to envision and deliver understandings regarding connections, hidden patterns, varying market trends, and many more. Therefore, its application fields are growing fast in number and many more companies exploit the power of Big Data Analytics to develop their businesses.

[2]Source: Statista 2021
[3]Source: www.globenewswire.com

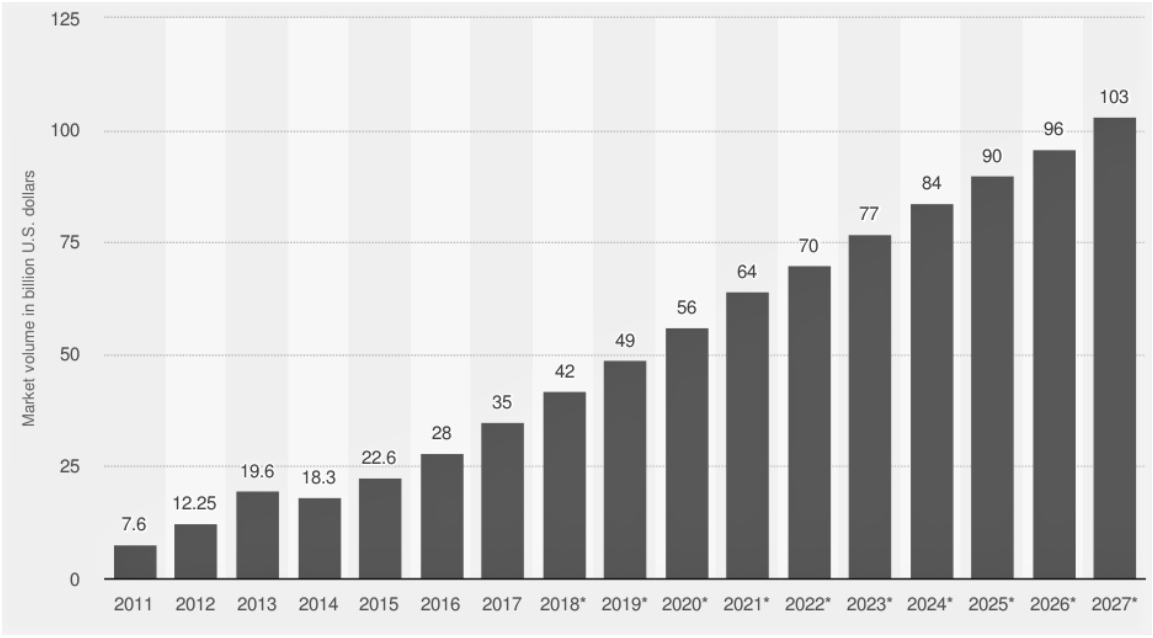


Figure 1.4: Big Data market size revenue worldwide from 2011 to 2027 (in billion USD), source: Statista 2021

Regarding the industries, the growth is expected in almost every field, with investments in Data Management and Analytics that touch the 10%. Insurance, Manufacturing and Telecommunication fields present the strongest growth. Indeed, almost 8 big companies out of 10 integrate data coming from different internal and external sources and the 54% is experimenting in the Advanced Analytics field.
At the same time, the **volume of data** that companies are going to handle is growing very fast, following an exponential curve. The total amount of data created, captured, copied, and consumed globally is reached 64.2 zettabytes (10^21 bytes) in 2022 [4]. Over the next years up to 2025, global data creation is projected to grow to more than 180 zettabytes, as it is clear from Figure 1.7.

[4]Source: Statista 2021

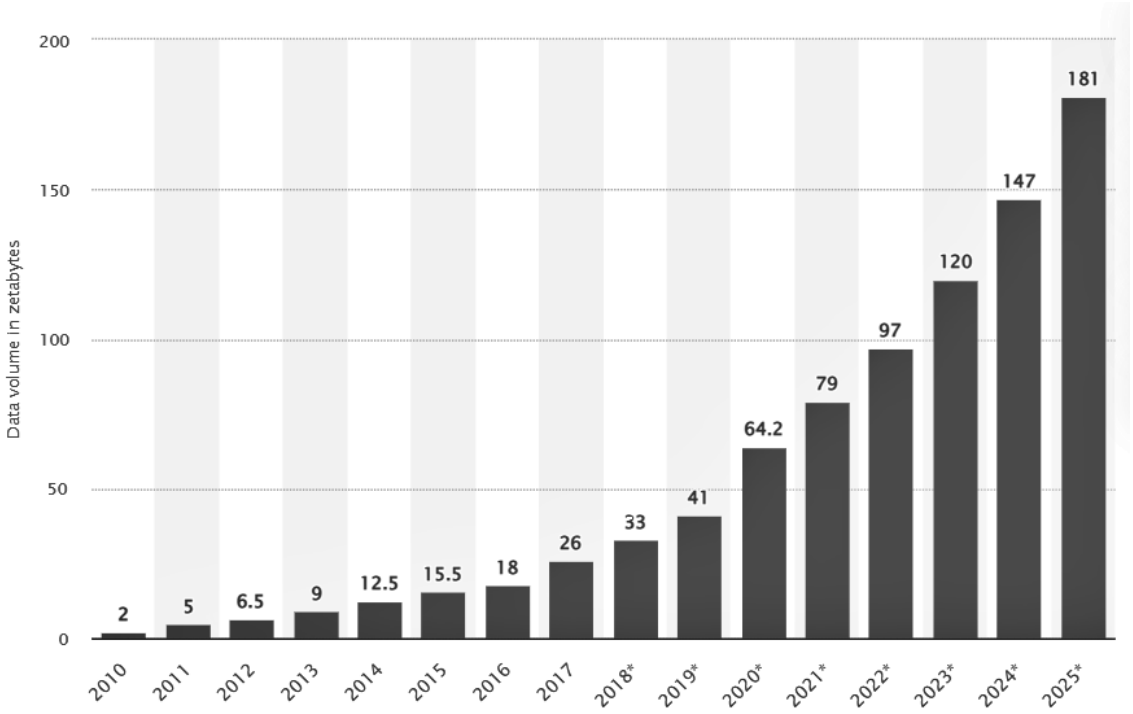


Figure 1.5: Volume of data created, captured, copied, and consumed worldwide from 2010 to 2025 in zettabytes, source: Statista 2021

As a consequence, companies need to manage increasing amounts of data to **extract value** from them. **Data Science** is becoming key in the decision making process for all the big companies and it constitutes an important source of competitive advantage. Therefore, also the demand of competencies is continuously increasing: the number of Data Scientists is increasing with a growth rate of 28% in the big companies. This is the reason why Data Science is something many more companies that operate in different fields are investing in and will invest in even more in the future.

1.7.2. Business Applications and Impacts

The **time series forecasting** is considered a key activity that comes under the Data Science ones. It has many possible applications and it is one of the most common areas where research and business applications go hand in hand. Indeed, time series forecasting is based on data science and it is applied in many business cases.
For example, price prediction in time series forecasting can produce great opportunities and increase the customer experience in many fields, such as the **flight industry**. Indeed, prices of airplane tickets are mostly fluctuating and price aggregators can offer a better experience to the customers if they are able to predict future prices since they can send

notifications to clients who want to know whether the price drops down. This encourages customers using this platform as their go-to platform for optimizing their travel budgets. Consequently, the time series forecast constitutes, in this case, a strong source of competitive advantage.

Then, during the pandemic, time series forecasting has become the key technique applied in **healthcare** to predict the spread of Covid-19. It has been used for predicting the transmission rate, the mortality ratios, the spread of the epidemic, and more. From this moment on, the time series prevision has become fundamental in many healthcare fields such as genetics, diagnosis and treatment and the results reported an important progress.

Time series forecasting is important also in the **financial industry** since investors and traders try to forecast the behaviour of financial markets through some variables such as the price of stock options, the volatility of the markets and the foreign currency risks. The estimation of these indicators of the future market will enable investors to construct portfolios and estimate the risk they bring. In Figure 1.6 the forecasting of the stock price of Google for 2021 has been executed using the Facebook Prophet model, as it is clear from the example in [35].

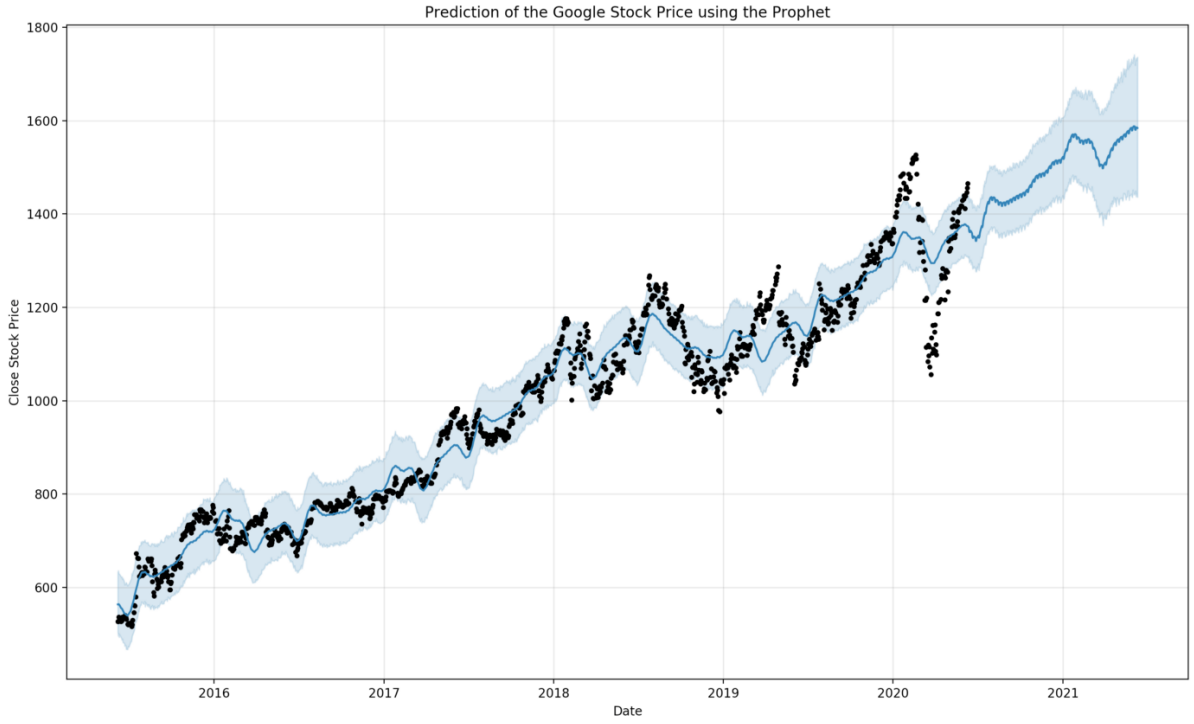


Figure 1.6: 2021 Google stock price predicted in 2020 using the Facebook Prophet model

Finally, the most general applications regard **business processes** in general, and therefore they are applicable to many different companies in different fields. Predicting cus-

tomer demand is a core objective which is fundamental for every company since it allows the enterprise to offer a better service and therefore to gain competitive advantage over the competitors. It then relies on the effectiveness layer. Moreover, all the internal activities can be better managed, correctly allocating the scarce resources such as time, money, materials, technical competencies that any company has. In this way, the waste is reduced and therefore a higher level of efficiency is reached. Moreover, data analysed, manipulated, aggregated and visualized in an effective way can constitute a solid base that relies on the scientific method and that can help managers to take the most complicated decision in strategic, tactical and operational terms. The following figure shows the most important **benefits**, presented in literature in [20], [36], that the Forecasting Science and, more in general, the Data Science, bring to almost every company.

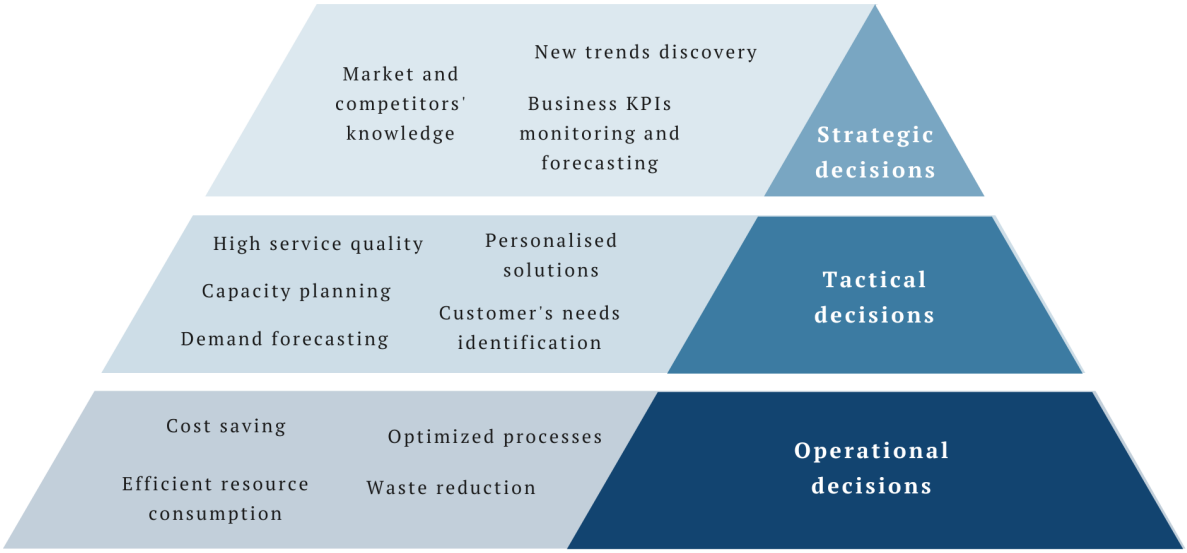


Figure 1.7: Main benefits of the forecasting science at strategic, tactical and operational levels

1.7.3. Business Objectives

The objective of this thesis is to analyse the most largely used forecasting models and algorithms, in order to understand if one is more appropriate than the other ones for predicting time series. Their performances are assessed not only in absolute terms, but also in relation to the hyperparameters they assume and in connection with the characteristics that the time series presents. Therefore, more than one dataset, with different characteristics, are treated. Moreover, an effort is made also in analysing the impact that

the choice of the evaluation metric has on the drafts that emerge.
The final aim, from a business viewpoint, consists in supporting companies in pursuing a very common goal: forecasting the sales of the future periods to better plan the organization of their internal resources. This is the reason why the datasets under analysis will treat **business time series** exclusively **related to the sales of different manufacturing companies** such as food retailers, pharmaceutical firms and more.

2 | Experiments settings

This chapter is dedicated to the description of the way the experiments on the various model are conduced in terms of datasets' choice, tested models, parameters' tuning methodology and calculated accuracy metrics. Moreover, we present a general overview of the preliminary analysis that are made on each dataset.

2.1. Datasets Description

In this section, we present the structure of the datasets under analysis in terms of records and attributes, the context they are inserted in and their main general characteristics.

2.1.1. Dataset A: Pharma Dataset

The first dataset that is going to be analysed is taken from [45], and it is constituted by a **multivariate time series of pharmaceutical products' sales**. The time series contains data recorded at a small scale, for a single distributor, pharmacy chain or individual pharmacy. Even if this dataset is made by a multivariate time series, we treat it in this case in a set of univariate time series forecasting problems.
The initial dataset consisted in 600000 records of transactional data, collecting in almost 6 years from 2014 to 2019 (the last one is not complete). It contained information regarding the drug category, the date and time of sale and the category the product belonged to. Even if the original time series presented a very low frequency of transactions, other datasets with (more) aggregated data are available in the same source. For this analysis, the main dataset that is considered is the one that presents a daily frequency of observations, with 2106 records. This choice is mainly due to the very large applicability of the methods developed on this dataset: the daily sales time series are very commonly used by companies. At the same time, a daily frequency is a good compromise between easiness in manipulation and accuracy in representing short term fluctuations.
The transformation of the dataset from the original one to the one that is used in this thesis is well explained in Figure A.1, taken from [45].

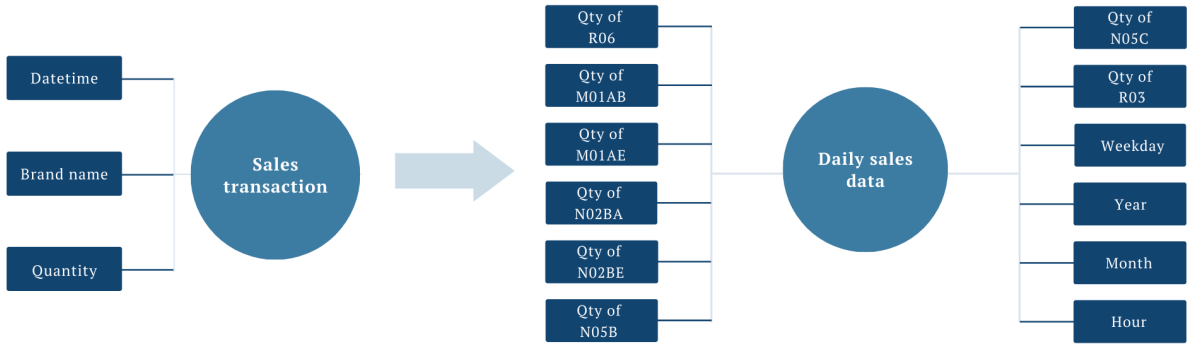


Figure 2.1: Tranformation from the original dataset to the daily time series

Regarding the time, the dataset presents a first column, named *datum*, that contains the timestamp at which the transactions take place and other features that contains other derived temporal variables. The list of these variables is presented below:

- *Year*;
- *Month*;
- *Hour*;
- *Weekday Name*.

The column named *Hour* is available but it is clearly meaningful only in the hourly dataset.
Moreover, a series of variables that indicates the 8 categories of the original 57 sold products are available as columns of the dataset.

- *M01AB* indicates the sales volume of anti-inflammatory and antirheumatic products, non-steroids, acetic acid derivatives and related items.
- *M01AE* indicates the sales volume of anti-inflammatory and antirheumatic products, non-steroids and propionic acid derivatives.
- *N02BA* indicates the sales volume of other analgesics and antipyretics, salicylic acid and derivative substances.
- *N02BE/B* indicates the sales volume of other analgesics and antipyretics, Pyrazolones and Anilides.
- *N05B* indicates the sales volume of psycholeptics and anxiolytic drugs.
- *N05C* indicates the sales volume of psycholeptics drugs, hypnotics and sedatives drugs.

- *R03* indicates the sales volume of drugs for obstructive airway diseases.
- *R06* indicates the sales volume of antihistamines for systemic use.

It is important to highlight the fact that data cleaning and feature engineering have already been performed before loading the dataset, therefore, it does not present any missing data and the time series results as complete and cleaned, and the records are already sorted in temporal order.
Last, before making any kind of analysis on the dataset, it is split in 2 subdatasets: train and test ones. The train set presents 1825 rows and it covers exactly 5 years, from 2014 to 2018 included; it presents all the features. The test set, instead, presents 281 records that cover almost 10 months of 2019. It presents the column containing the timestamp and the temporal features only, since the ones related to the volumes must be predicted. From now on, we'll indicate these datasets as *train dataset* and *test dataset*. The dataset containing the real data of 2019 will be named as *real dataset*.
The head of the complete dataset is reported in Appendix A.

2.1.2. Dataset B: Food Demand Forecasting Dataset

The second dataset we are going to analyse is taken from [33], and it is constituted by a **univariate time series of food sales**. Data are taken from the sales of a meal delivery company that operates in many different cities. The dataset presents also some geographical variables that authorize the treatment of this forecasting problem in a set of univariate time series forecasting problems.
The original main dataset is composed by 456548 records and 9 columns which contain both temporal and geographical variables and variables related to the price of the order. Regarding the frequency and the deepness of the dataset, it presents weekly data, distributed along 145 weeks. It is important to highlight the fact that, differently from the previous dataset, in this case only the number of the week is known, but any information regarding the timestamp of the transaction is available: it means that week 1 can be any week of any year, it is not known as a specific week.
Moreover, another additional dataset is given: it contains information regarding the fulfillment center. The first and simplest action that is made is the joining of the two tables in a unique dataset, using the center unique identification code as the key attribute. The final dataset presents the attributes that follows.

- *Id* is the identification code.
- *Week* is the number of the week the transactions refer to.

- *Meal Id* is the identification code of the product that is sold.
- *Checkout Price* is the price of the checkout for that demand.
- *Base Price* is the base price for that demand.
- *Emailer for promotion* is a flag that indicates the subscription to promotions.
- *Homepage featured* is a flag related to the featured homepage.
- *City code* is the code that identifies the specific city of those orders.
- *Region code* is the code that identifies the specific region in which those orders take place.
- *Op area* is the geographic area that identifies the operative area with that demand.
- *Center type* indicates the type of center in which that number of orders takes place.

The target variable that should be predicted is called *Num orders* and it refers to the number of orders that are made in the specific geographic area and in the specific week indicated by the other features. It is a measure of the demand of each segment and it is expressed in terms of volumes, not of revenues.
A schematic view of the dataset structure and its attributes is available in Figure A.2.

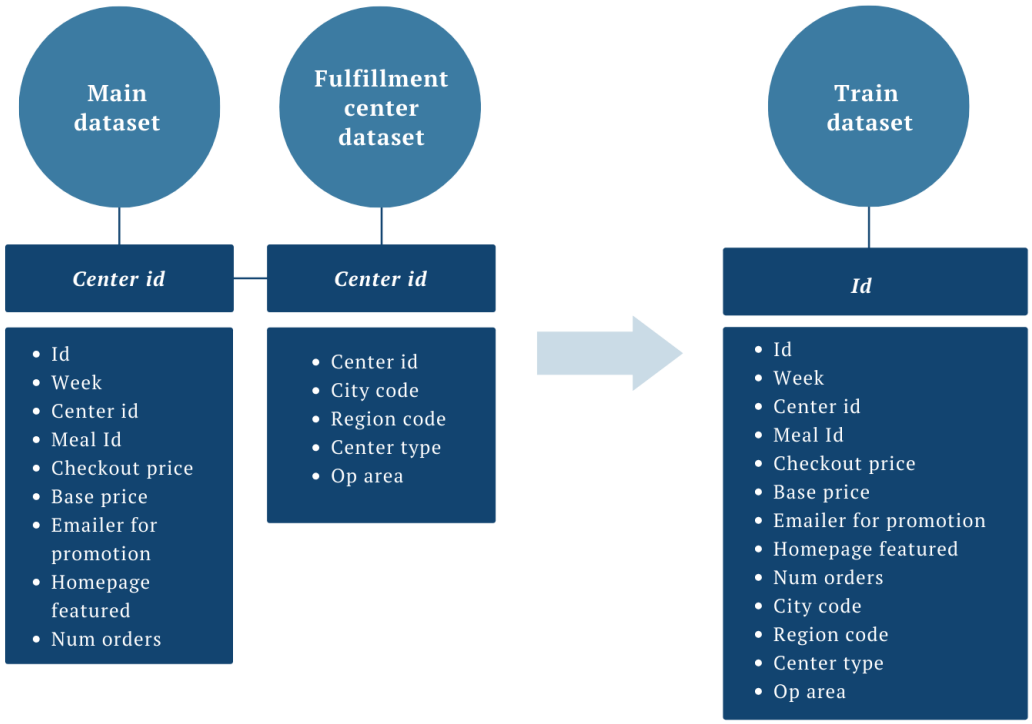


Figure 2.2: Tranformation from the original datasets to the Food Demand Forecasting dataset

Also in this case, before analysing the dataset, it is divided in 2 subdatasets: train and test ones. The train set presents 430283 rows and it covers a time horizon of exactly 137 weeks; it presents all the features. The test set, instead, presents 35265 records that cover the remaining 8 weeks. The latter presents all the informative features except the one named *Num orders*, since the sales volumes must be predicted. From now on, we'll refer to these datasets as *train dataset* and *test dataset*. The dataset containing the real data of 2019 will be named as *real dataset*.
The head of the complete original dataset is again reported in Appendix A.
It is important to highlight the fact that this dataset does not present a unique value for moment in time. For this reason, it is necessary to create many sub time series filtering the dataset with respect to the values of some features and aggregating taking the timestamp as index. Each of the columns of the final dataset - apart from the timestamp one - is one of the time series obtained as explained above. In particular, each column is named with the original column name and the value on which the dataset has been filtered. Moreover, the *Aggregate* column is obtained without filtering the dataset and the *ts* one contains the timestamp that refers to the first day of the respective week indicated in the original dataset. Indeed, the time series forecasting requires a specific time moment for every

observation: the first day is set as January 1^{st}, 2018 in order to avoid any bias due to the Covid-19 emergency.
The head of this final dataset is reported in Appendix A.

2.2. Tested Models

The models that are going to be tested in this thesis refer to the theoretical frameworks deeply described in Chapter 1. From a practical viewpoint, the code that trains and tests all the models is created following the main guidelines available in literature [39], [34], [25], [26], [37], [46].
More details regarding the parameters of the functions mentioned below and their configuration spaces are deeply explained in Section 2.3.

2.2.1. Exponential Smoothing Models

The models that belong to the Exponential Smoothing category we are going to test in this thesis are listed below. The functions are taken from the Python library `statsmodels`.
Holt's Winter Seasonal Exponential Smoothing includes a trend component and a seasonal component. In this case the function is named `ExponentialSmoothing` and it is set with many different parameters that indicates the way of aggregating components (additive or multiplicative) the presence of a damped trend, the use of a Box-Cox transformation and the initialization method. It is the most general method of this class and, for this reason, we will test this method only, changing the parameters that regulates the weights of the various components, falling in different particular cases such as the Simple Exponential Smoothing and the Holt's Exponential Smoothing.

2.2.2. Autoregressive Models

Also in this case, the models of the Autoregressive category are tested through the functions of the Python library `statsmodels`. The models that are trained and tested as well as the functions that are used to test the characteristics of the time series are listed below.

- The **check for stationarity** of the time series is performed through the function `adfuller`.
- The **order of the AR model** to be trained is determined by the partial autocorrelation plots, made through the function `plot_pacf`.
- The function `auto_arima` is used to **train the Autoregression model** in an auto-

matic way. This function does not require any kind of preliminary study to identify the most appropriate parameters of the model, but it tests many different combinations of parameters and it creates the model with the optimal one. Regarding the assessment of the different configurations, it uses the **Akaikeś information criterion (AIC)** to evaluate which is the optimal configuration. Indeed, it estimates the quality of each model, relative to each of the other models; thus, it provides a sort of method for model selection. AIC is founded on information theory. It estimates the relative amount of information lost by a given model when it is trained: the less information a model loses, the higher the quality of that model.

2.2.3. Regression models

Regarding the machine learning (regression) algorithms, some of them are proven to be more efficient in forecasting time series. Several studies such as [40], [43] have been done that show a higher effectiveness in this application field. The most effective Regressors for time series prevision that are going to be tested in this thesis are listed below.

- **Linear Regressor** is the most simple regressor and the one that represents the baseline. The other regressors are usually tested starting from it. This model is tested though the function `LinearRegression` of the module `linear_model` belonging to the library `scikit`.
- **Support Vector Machines (SVMs))** indicates a set of supervised learning methods. As it is stated in [29]: "The ability of SVMs to solve nonlinear regression estimation problems makes SVMs successful in time series forecasting". Indeed, this set of methods is based on the subdivision of the dataset in sub datasets using (linear or non linear) hyperplanes. SVMs are implemented through the functions `LinearSVR` and `SVR` of the Python module `svm` of the library `sklearn`, as stated in the `scikit-learn` documentation [18].
- **Ridge Regression** is a method of estimating the coefficients of multiple-regression models. It is a model belonging to the class of the elastic nets. Also in this case, the function `Lasso` belongs to the library `sklearn`.
- **Lasso Regression** is a regression analysis method that performs both variable selection and regularization in order to enhance the prediction accuracy and interpretability of the resulting statistical model. The function that is used is the `Ridge` one, belonging to the `sklearn` library.
- **K-nearest neighbors algorithm (k-NN)** is a non-parametric supervised learning

method used for classification and regression. The input consists of the k closest training examples in a data set. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. The KNN method is implemented through the function `KNeighborsRegressor` of the module `neighbors` of the `sklearn` library.

- **Gradient Boosting Regression Method** is an efficient implementation of a stochastic mathematical model. It is an ensemble of decision tree algorithms where new trees fix errors of those trees that are already part of the model. Trees are added until no further improvements can be made to the model. The function that tests the behaviour of this model is `GradientBoostingRegressor` from the library `sklearn.ensemble`, and the details of the implementation are deeply treated in [27], [16].

All these machine learning models are tested with different sets of hyperparameters. The tuning process of these latter is deeply explained in 2.3.1. Then, the forecasting is executed using the sliding window through the class `ForecasterAutoreg` of the library `skforecast`, as explained in [13], [31]. For this forecasting, the subseries that are not predicted are used as features to explain the target one and the timestamp is taken as the row index of the dataset. The predictions produced by each model are then compared through the metrics explained in Section 2.4.

2.2.4. Prophet Model

The **Prophet Model** is a particular procedure implemented by Meta (Facebook) and available both in Python and R. It is an open-source library called `fbprophet` that offers many methods for the forecasting of univariate time series. The specific function that we present and use in this thesis is called `Prophet` and it is able to create a model, to fit it and to use the model to predict the future data.

2.3. Parameters

In this section, we present the hyperparameters tuning methods that are used to test the models and to optimize the performance of each algorithm. Moreover, a description of the parameters of the functions and of the domain in which they take their values is provided.

2.3.1. Hyperparameters Tuning

As stated in [17]: "in machine learning, hyperparameter tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process".
The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. They must be tuned so that the model can optimally solve the problem. Hyperparameter optimization finds a combination of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss.
In general, the hyperparameter optimization can be approached in two main ways, described below.

- **Grid search** searches the optimal parameters in a manually specified subset of the hyperparameter space. It exhaustively searches the space in a sequential manner and trains a model for every possible combination of hyperparameter values. The grid search algorithm trains multiple models (one for each combination) and finally retains the best combination of hyperparameter values, evaluated by some performance metrics.
- **Random search** define a search space as a bounded domain of hyperparameter values and randomly samples points in that domain. It can outperform Grid search, especially when only a small number of hyperparameters affects the final performance of the machine learning algorithm. It is very simple and it is one of the important base-lines for the hyperparameter optimization methods.

The main advantage of the Grid search optimizer is the completeness of the hyperparameter space in which it executes the search process. On the other hand, it is a very complex method in computational terms and therefore it is a very slow tuning method. The random optimizer instead is a faster but less complete method.
The two methods are visually represented in Figure 2.3 and Figure 2.4[1]. Note that the blue lines indicate regions with strong results, while red ones show the regions of the domain with the worst results.

[1]Source: www.wikipedia.org

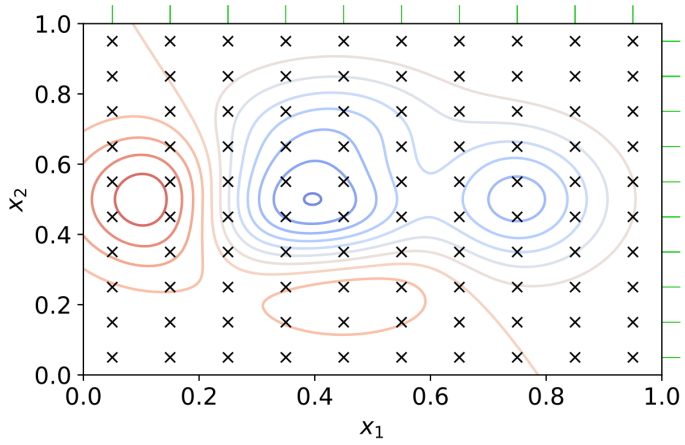


Figure 2.3: Grid search: different values of two different hyperparameters

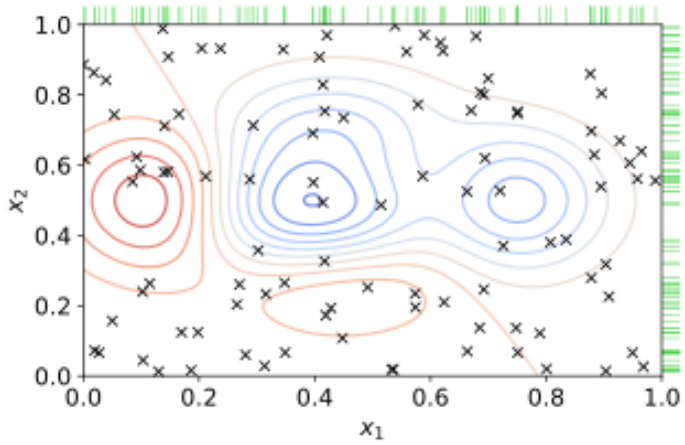


Figure 2.4: Random search: different values of two different hyperparameters

In this thesis, the hyperparameters are tuned in the same way, regardless the model we are testing and the level of complexity that the hyperparameter space presents. This choice is due to the fact that, since the objective of this thesis is to assess the models, we want to be as consistent as possible and to avoid any bias related to the different hyperparameter tuning methods. In this thesis, we will assess the models through the **Grid search optimizer**, since most of the parameters of the models present a discrete state space.

2.3.2. Hyperparameter Spaces

The models that are tested in this thesis present different sets of parameters. Each parameter of each model has a different domain and therefore the values that it assumes belong to this domain. The lists that follow in this Section include all the values that are tested during the experiments (grids), that constitute a discrete subset of the whole hyperparameters space of each model. The whole list of values is reported since it is meaningful in terms of model interpretability.
Instead, due to the high complexity of the machine learning models, the values assumed by their hyperparameters are not reported.
The **ExponentialSmoothing** method presents the following parameters:

- `trend: [additive, multiplicative, None]`,
- `damped_trend: [True, False]`,
- `seasonal: [additive, multiplicative, None]`,
- `seasonal_periods: [365, 30, 7]`
- `remove_bias: [True, False]`

It is important to highlight the fact that the parameter called `optimized` is always set as `True` since it automatically detects the best parameters that regulates the smoothing levels, α, β and γ.

Then, the **auto ARIMA** model presents some parameters that are correlated and that assume different values depending on the results of two tests: the `adfuller` test for stability and the `PACF` test for the auto-correlation of the time series. The `auto_arima` function automatically iterates over the parameters (P, D, Q) and the seasonal orders (p, d, q), considering the characteristics of the time series such as the stationarity. The parameters on which the model iterates are listed below:

- `seasonal: [True, False]`,
- `stationary: [True, False]`.

Then, the parameter `with_intercept` is set with the value `auto` because in this way the model automatically detects the trend component of the time series. The models are tested with many parameter configurations and the best one is assessed through the AIC.

Regarding the **Regression models**, each of them presents specific hyperparameters. Each of the following lists includes the space of the hyperparameters that is automati-

cally tuned by every regressor. In particular, only the most impactful hyperparameters' spaces are reported below. Regarding the **Linear Regressor**, it does not present any parameter since it is a very simple model. Instead, the **SVR**'s hyperparameters take values in the following space:

- `C: [10, 5, 15, 20, 50]`,
- `degree: [3, 2, 1, 0]`,
- `epsilon: [0.05, 0.01]`,
- `gamma: [0.5, 1]`,
- `max_iter: [-1, 20, 50, 100, 1000]`,
- `shrinking: [True, False]`.

Regarding the **Ridge** and **Lasso** regressions, they are trained with the following parameters:

- `normalize: [True, False]`,
- `alpha: [0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]` that, as it is noticeable, is a logarithmic parameter space.

The **kNN** model is then tested with the parameters listed below:

- `n_neighbors: [5, 10, 20, 25, 30, 35, 40, 45, 50, 55, 60]`,
- `p: [1, 2]`.

The **GradientBoosting** model is then tested with these main hyperparameters:

- `learning_rate: [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8]`,
- `max_depth: [2, 3, 4, 5, 6, 7, 8, 9]`.

Finally, the **Prophet** model is tested tuning some parameters whose names and respective domains are reported below:

- `changepoint_prior_scale: [0.001, 0.01, 0.1, 0.5]`,
- `seasonality_prior_scale: [0.01, 0.1, 1.0, 10.0]`,
- `holidays_prior_scale: [0.01, 0.1, 1.0, 10]`,
- `seasonality_mode: ['additive', 'multiplicative']`.

2.4. Evaluated Metrics

Regarding the metrics that are evaluated for each model, they are chosen with the objective to compare all the models, and therefore they should be applicable to all of them and also easy to compute. Moreover, they should be easy to interpret once the results are available. Among all the indexes presented in Section 1.6, the following ones are calculated for each model and each set of parameters:

- Mean Absolute Error (MAE),
- Mean Absolute Percentage Error (MAPE),
- Mean Absolute Percentage Error Adjusted (MAPE_adj),
- Mean Squared Error (MSE),
- Root Mean Squared Error (RMSE).

More precise considerations should be made for the Adjusted MAPE. This metric is defined exactly as the MAPE, but it is computed only on those observations of the future time series that are non null. This little trick makes the construction of a relative metric possible and, at the same time, enables the assessment of different time series with different orders of magnitude. On the other side, this metric presents a big drawback: it is computed only on a part of the previsions' vector, and therefore it is not computed exactly in the same way in every sub series.
Then, it is important to highlight the fact that all these metrics are computed for each experiments but only the **Mean Squared Error** is used to choose the best model. This choice is due to the fact that it should be evaluated on the same series and therefore it is not necessary to use a relative measure. Moreover, it is easily interpretable and it can be quickly computed when testing the regression models. It is computed taking as reference values the real values that the time series assumes in the future period.
Anyway, all the metrics are evaluated at every step since their values are assessed and their behaviours can be analysed. It would be easy to change the metric responsible for the choice of the best model, if necessary.

2.5. Experiments and Methodologies

Regarding the experiments that are executed and the methodologies that we follow in this thesis, many points must be investigated. The process that is described below will be followed for the analysis of the two datasets.

As a first step, a deep analysis is performed in order to highlight the main characteristics of the time series in terms of frequency of the time steps and, above all, statistical terms. The datasets are described and they are (eventually) split in more than one subsets using one or more filters. Indeed, it could be useful to split the time series in more than one sub-time series in order to assess whether the errors of the sub series are compensated in aggregating them in a unique time series or the greater accuracy of the previsions of the sub series leads to a greater accuracy in the comprehensive dataset.
Once the time series is analysed and the eventual sub time series are identified, all the methods mentioned in Section 2.2 are tested with the different parameters configurations listed in Section 2.3.1.
For each method, the optimal set of parameters is identified comparing the predicted time series with the *real dataset*, using the metrics above-mentioned in Section 2.4.
Then, all these best models are compared through the same metrics and the optimal model with the optimal set of parameters is identified.
At this point, also some combinations of the models are tested and compared with the previously mentioned algorithms. In this case, in order to simplify the experiments, the optimal combination of parameters is considered in each case.
The whole process is shown in a schematic way in Figure 2.5.

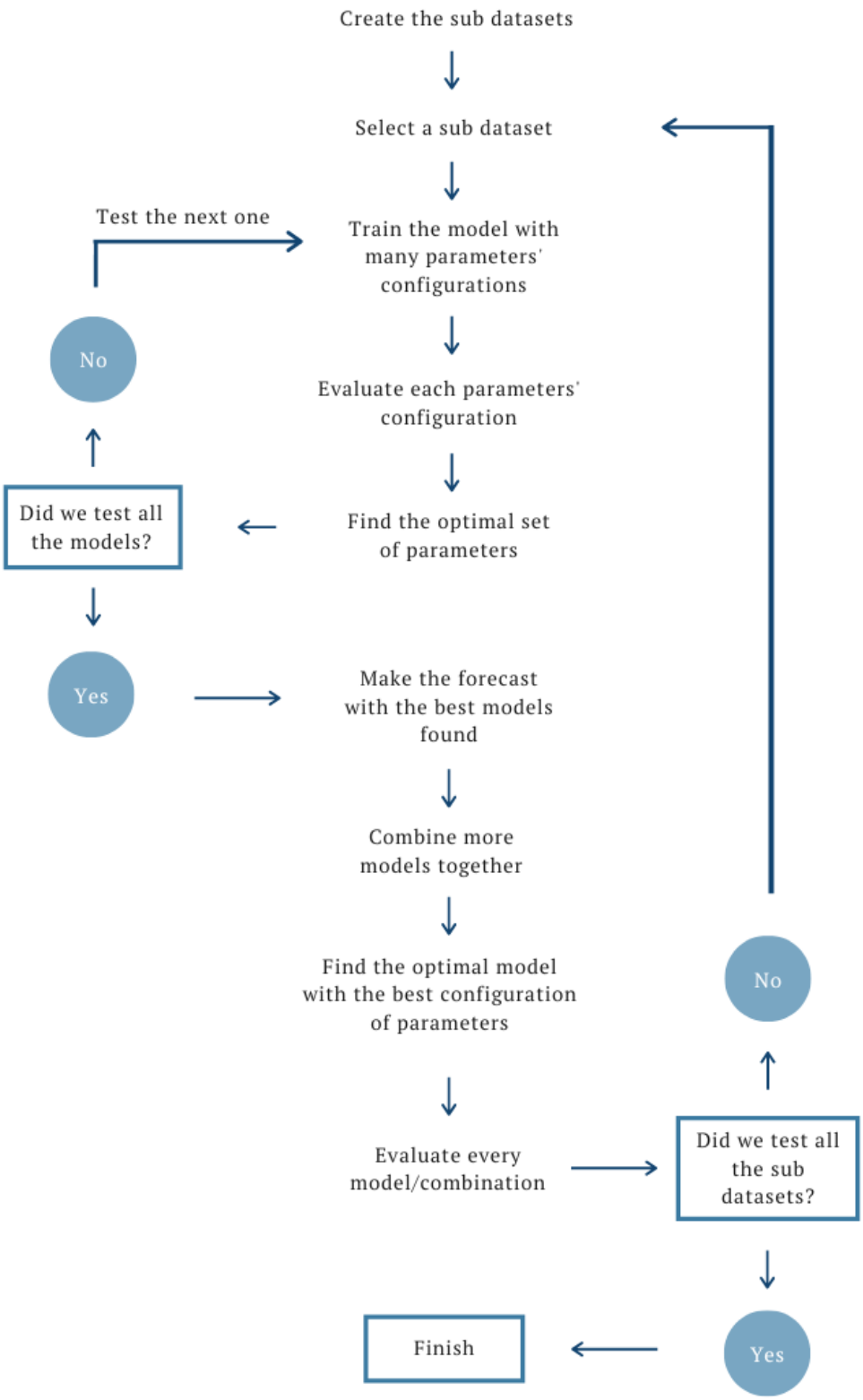


Figure 2.5: Experiments outline: process of forecasting trials

Regarding the process, it is important to introduce the method that is followed to execute the experiments. The algorithm to test the models is set trying to automate the process of assessment of the models and the parameters. In particular, some python functions are set to execute some tasks making them modular.

- The `preprocessing` function is aimed at executing the data preparation for all the models under analysis.
- The `fit_predict_evaluate` function is the most important one. For each model, it defines the hyperparameters space and it tunes them. For each combination of parameters, it fits the model and it makes the predictions on the future time interval. Then, it evaluates each model through the metric mentioned in 2.4.
- The `update_results` function writes the results in a file to make them easily readable and always available.

Then, we call all these functions into a cycle that iterates over subdatasets and models. In this way, the same process is made for all the experiments and it is - at least partially - automated.
In Algorithm **??**, the main logical steps of the experiments' outline are reported.

Algorithm 2.1 Experiments outline

```
define sub series, models and hyperparameter spaces
for eachtimeseriests do
    import time series ts
    for eachmodelm do
        preprocess data for m
        tune hyperparameters for m on ts
        fit moel m on d
        predict future data
        evaluate predictions
        save metrics
    end for
end for
```

The complete cycle that regulates the process of experimentation that we follow in this thesis is reported in Appendix B. The implementation of this algorithm is executed in Python and it exploits the functions explained above.

Finally, an analysis is made creating an aggregated time series which is computed as

the sum of all the sub series of the Pharma dataset analysed and forecasted in the previously explained process. It is predicted testing the same sets of models and parameters listed above. From now on, this first future time series will be denoted as *Configuration B*. It is compared with the prediction of the future series obtained as the sum of the sub time series predicted with the best configuration established through the experiments. This second series will be indicated as *Configuration A*. The two different terms of the comparison are obtained from the process explained in Figure 2.6.

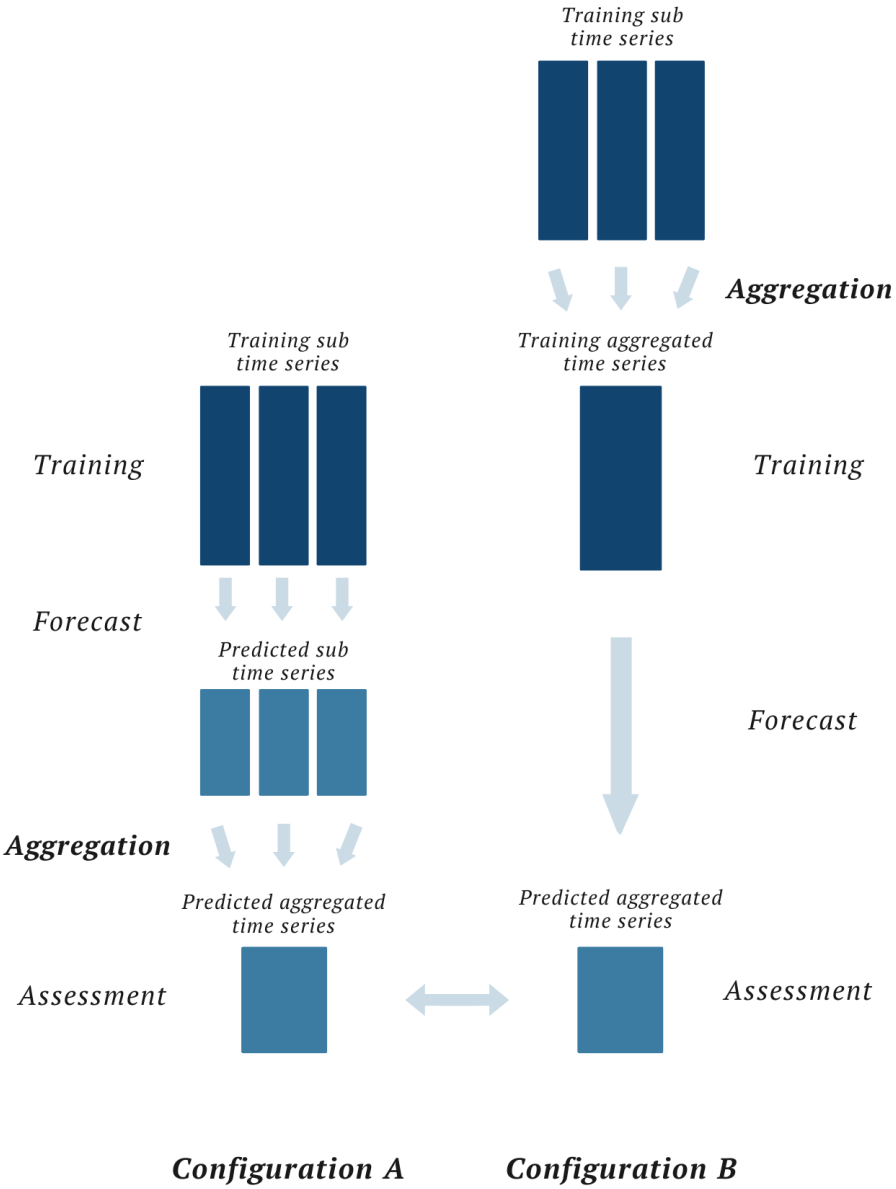


Figure 2.6: Aggregation of the sub time series at different levels: *Configuration A* on the left and *Configuration B* on the right

The objective of this final analysis is to verify the impact of having a high accuracy in time series predictions. Indeed, if the gain in terms of accuracy is not significant, it is easy to figure a better performance in the predicted aggregated time series, due to the compensation of opposite errors in the sub series. This analysis can suggest the subdivision of the time series into more sub series, created filtering the dataset with respect to some business variables. However, in this specific case, the sub series are treated as independent ones, ignoring the correlation that obviously exists between them.

3 | Data Exploration and Preliminary Analyses

This chapter is dedicated to the preliminary analysis of the time series that are going to be treated in this thesis. The objective is to evaluate which are the most important features of the time series and the technical characteristics they have. Moreover, a general description of the dataset is fundamental in order to perform the best possible data preparation.

3.1. Pharma Dataset

First of all, it is important to make a consideration regarding the granularity of the Pharma dataset. In particular, each of the 9 time series coming from this dataset presents a **daily frequency**; therefore, each of them presents all the fluctuations that are appreciable in a short time span such as any eventual weekly periodicity.

3.1.1. Time Series Decomposition

The first step consists in the decomposition of the time series in its main components, to analyse ad report its main characteristics. The decomposition of the time series is supported by the function `seasonal_decompose`. To perform this analysis, a choice among a multiplicative and an additive composition model must be made. As it is stated in [38], "we can usually identify an additive or multiplicative time series from its variation: if the magnitude of the seasonal component changes with time, then the series is multiplicative; otherwise, the series is additive". Therefore, an estimation of the seasonality magnitude is made for each sub series through the plot of the seasonality in the first and last year of analysis.
The result is the same for all the sub series that present a magnitude of the seasonal component which varies over time. The aggregated time series is presented as an example. Its seasonal component is graphically represented in 2014 and 2018 in Figure 3.1.

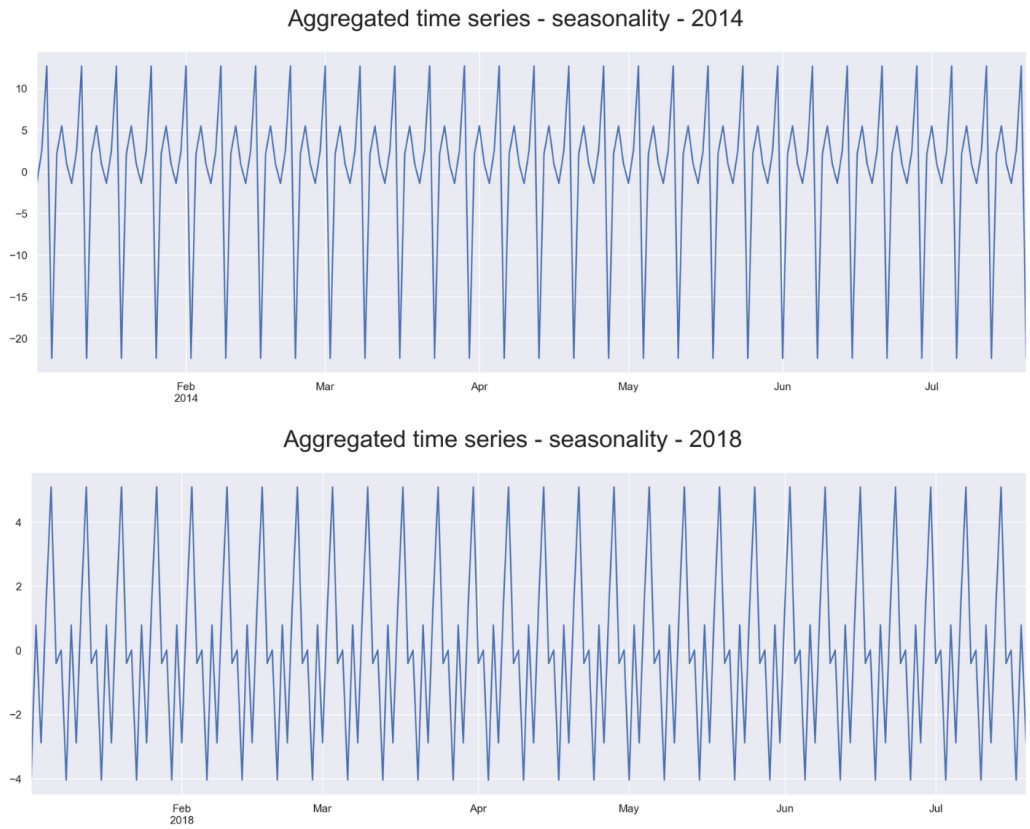


Figure 3.1: Magnitude of the seasonal component of the aggregated time series, at the beginning and at the end of the time span of analysis (2014 and 2018)

As it is clear from the graph, the magnitude of the seasonality effect changes over time. Therefore, for the decomposition of the time series, we will consider a multiplicative model. The decomposition into the main components is performed through the function `seasonal_decompose` of the `statsmodels` library and it is performed for every sub time series. Regarding the behaviour of the time series, every sub time series presents a clear **weekly periodicity**, that remains quiet constant as the time goes by. This phenomenon is visible in Figure 3.2, where the aggregate time series is represented on a time horizon of one month.

Figure 3.2: Weekly periodicity of the aggregate time series in January 2014 and January 2018

As it is clear from the graph, the time series is periodic in a week time span even if the shape of the function changes over the five years of analysis. The aggregate time series is taken as an example, but the same behaviour is visible also analysing the sub time series. An exhaustive representation of this phenomenon in the sub time series is provided in Appendix C.

At this point, all the time series are decomposed into the main components and they are represented in a graphic. In Figure 3.3, we report again the aggregate time series as an example.

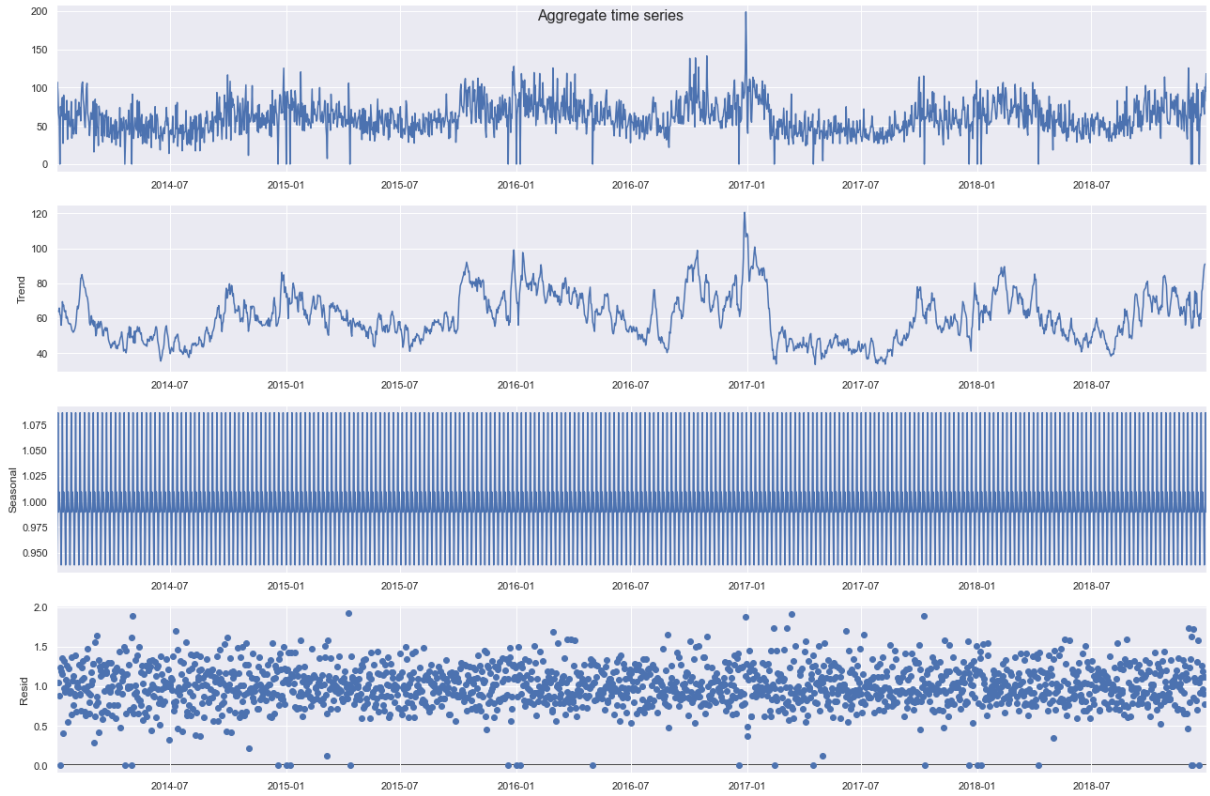


Figure 3.3: Decomposition of the aggregate time series into its main components through a multiplicative model

Regarding seasonality, in this time series the seasonal component seems to be very weak. Anyway, the behaviours of its sub series are **very different one from another**: some series present a very strong seasonal effect during the year while other products seem not to have a yearly seasonal pattern. We report in the Figure 3.4 and Figure 3.5 the two most significant examples, while a complete representation of all the sub series is available in Appendix C.

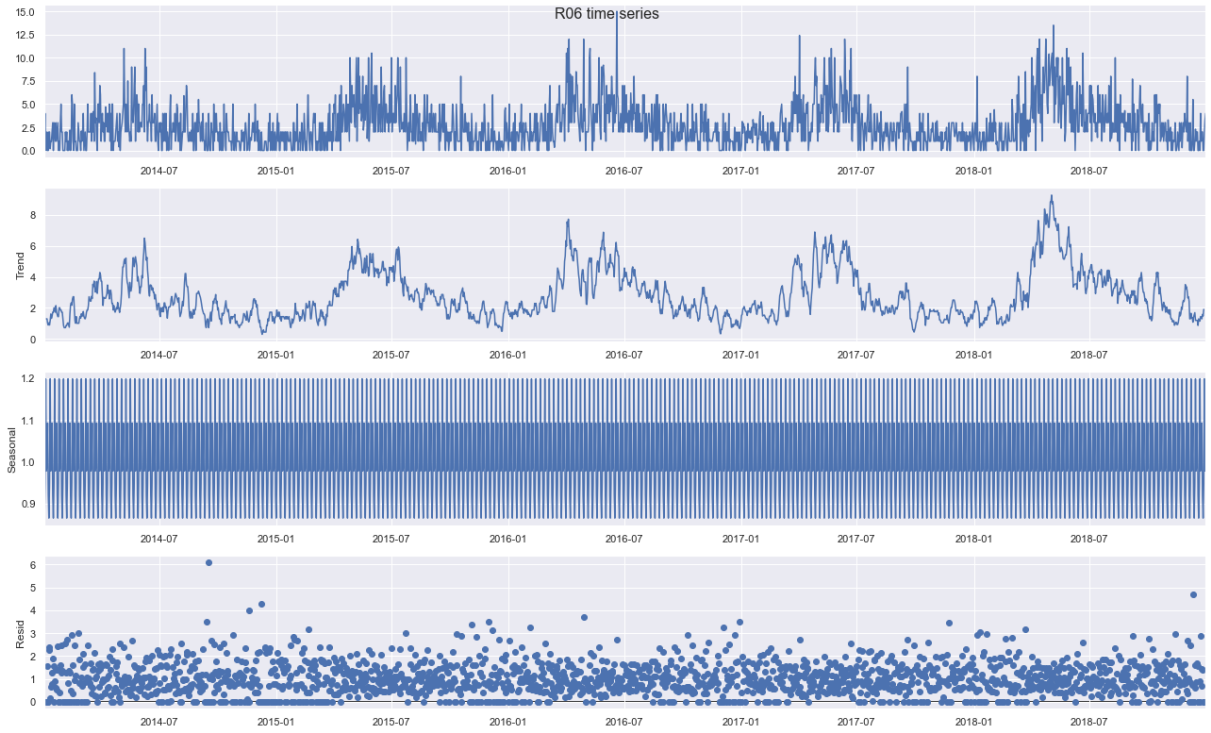


Figure 3.4: Decomposition of the *R06* item time series into its main components: the seasonal behaviour along the year is evident

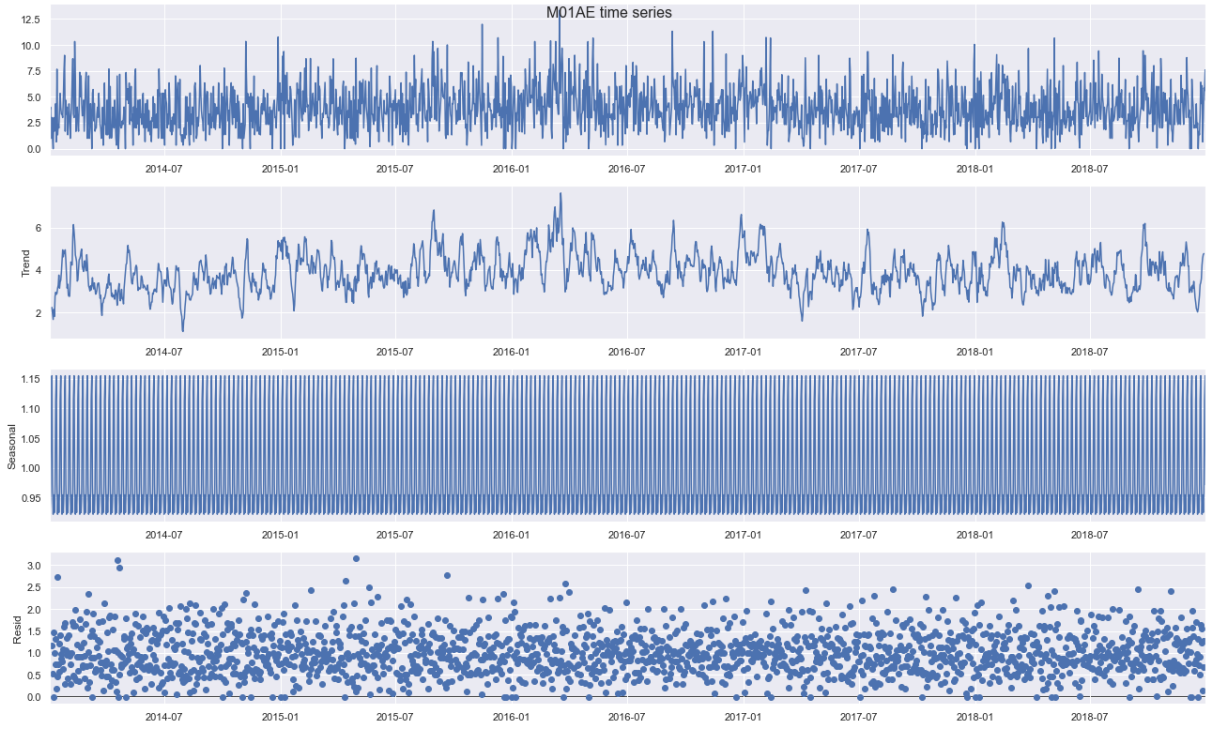


Figure 3.5: Decomposition of the *M01AE* item time series into its main components: the seasonal behaviour along the year is almost absent

On the other hand, the **trend component** is not so evident in all the cases: all the time series seem to be almost stationary and any particular tendency or monotony is visible from the above reported graphs. Moreover, an Augmented Dickey–Fuller test is executed to verify the stationarity of the time series, as explained in [8]. This test is a hypothesis test: the null hypothesis is that the time series is non-stationary, while the alternative one states that the time series is stationary. The result of the test is determined by a threshold value that is 1% in this thesis:

- if the *p-value* of the test is greater than the threshold of 0.01 then the null hypothesis is accepted and the time series is considered as non-stationary;
- if the *p-value* of the test is lower than the threshold of 0.01 then the null hypothesis is rejected and the time series is considered as stationary.

In the following table the *p-values* resulting from the Adfuller test are reported for every time series that is analysed.

Time series	p-value	flag stationarity
Aggregate	0.000700	1
M01AB	0.000000	1
M01AE	0.000000	1
N02BA	0.000000	1
N02BE	0.002897	1
N05B	0.000157	1
N05C	0.000000	1
R03	0.000618	1
R06	0.007405	1

Table 3.1: P-values of the Adfuller test for all the time series related to the Pharma dataset

In this table, the last column indicates the result of the test, so the stationarity of the time series. It assumes value 1 when the the series is stationary, otherwise, it assumes value 0.

3.1.2. Statistical Evidences

Regarding the time series under analysis in statistical terms, a series of indicators is computed for each of them, in order to catch their differences and similarities in these terms. It must be considered that each of these time series is composed by **1825 time steps** distributed with a **daily frequency**.

Time series	mean	std	cv	min	max	25%	50%	75%
Aggregate	60.55	21.26	35.11	0.0	198.95	46.66	58.52	73.33
M01AB	4.98	2.71	54.54	0.00	17.00	3.00	4.68	6.66
M01AE	3.90	2.09	53.67	0.00	13.34	2.34	3.67	5.19
N02BA	4.00	2.43	60.76	0.00	16.00	2.00	4.00	5.30
N02BE	30.15	15.46	51.28	0.00	161.00	19.30	27.20	38.60
N05B	8.90	5.78	64.99	0.00	54.83	5.00	8.00	12.00
N05C	0.58	1.09	188.95	0.00	9.00	0.00	0.00	1.00
R03	5.29	6.13	115.86	0.00	41.00	1.00	3.00	8.00
R06	2.76	2.34	84.82	0.00	15.00	1.00	2.00	4.00

Table 3.2: Main statistical indicators for all the time series related to the Pharma dataset

At this point, the correlation and autocorrelation of the various time series are analysed. As stated in [11], in statistics, the correlation is a statistical relationship between two random variables or bivariate data.
The autocorrelation is conceptually similar to the correlation between two different time series. It is the correlation that incurs within the same time series and basically links the past data with the future ones, but it does not touch any other variable.
In this thesis, we analyse the correlation that exists between the sub time series in couples. Of course, the *Aggregate* time series is excluded from this work. In Figure 3.6, the complete correlation matrix of the *Pharma dataset* is reported.

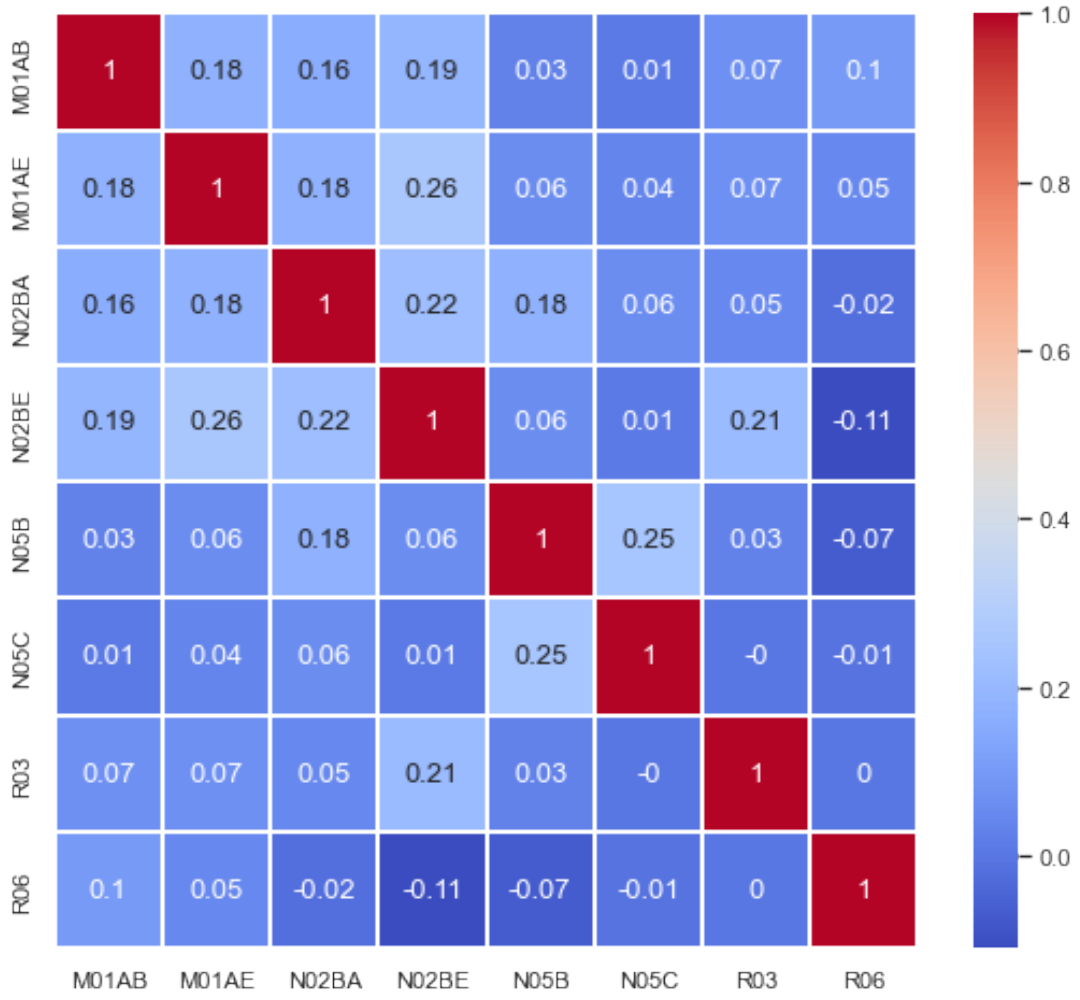


Figure 3.6: Correlation matrix of the sub series of the *Pharma dataset*

As it is clear from the matrix, a weak correlation of around 20% exists between the *M01AB*, *M01AE*, *N02BA*, and *N02BE* time series. In this thesis, we don't take into consideration any potential correlation between two time series but they are analysed separately. Anyway, as a future development of this thesis, these correlations can be taken into consideration taking as exogenous variables the other time series, as it will be well explained in 4.

Regarding the autocorrelation, it is assessed through the representation of the time series in a 2D plot showing the lag value along the x-axis and the correlation on the y-axis between -1 and 1. Confidence intervals are drawn as a cone. In general, it is set to a 95% confidence interval, suggesting that correlation values outside of this area are very likely a correlation and not a statistical fluctuation. Figure 3.7 and Figure 3.8 present this kind of plot in two very different cases.

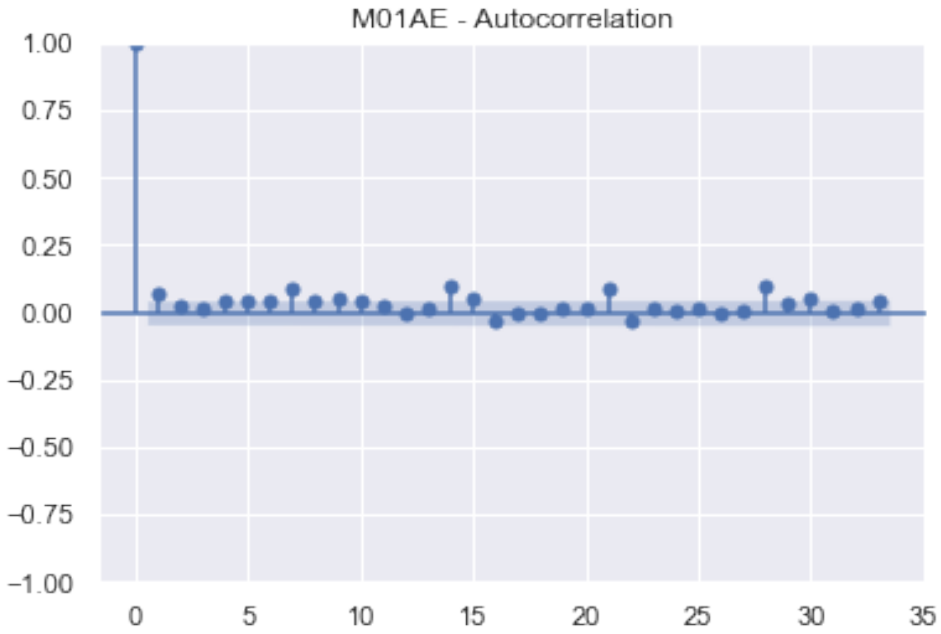


Figure 3.7: Plot of the autocorrelation of the *M01AE* time series

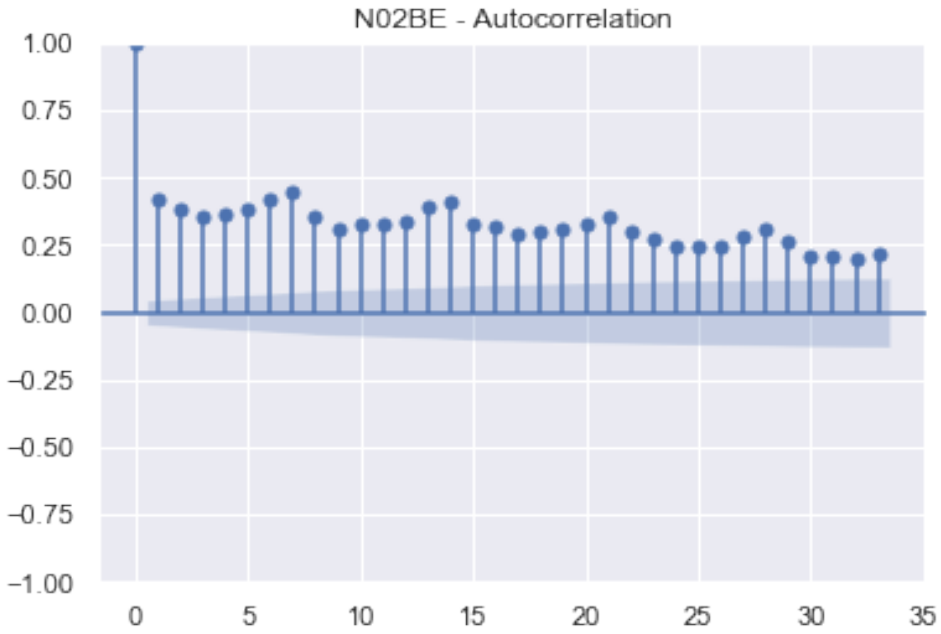


Figure 3.8: Plot of the autocorrelation of the *N02BE* time series

As it is evident from the two graphs reported above, the two time series under analysis present very different behaviours in terms of autocorrelation: the *M01AE* time series' plot is always inside the confidence interval while in Figure 3.8, *N02BE* series presents a strong positive autocorrelation.

A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps. The partial autocorrelation at lag k is the correlation that results from the removal the effect of the correlations due to the terms at previous lags.
Taking as examples the two time series mentioned above, we can see also in this case very different behaviours, as it is shown in Figure 3.9 and Figure 3.10

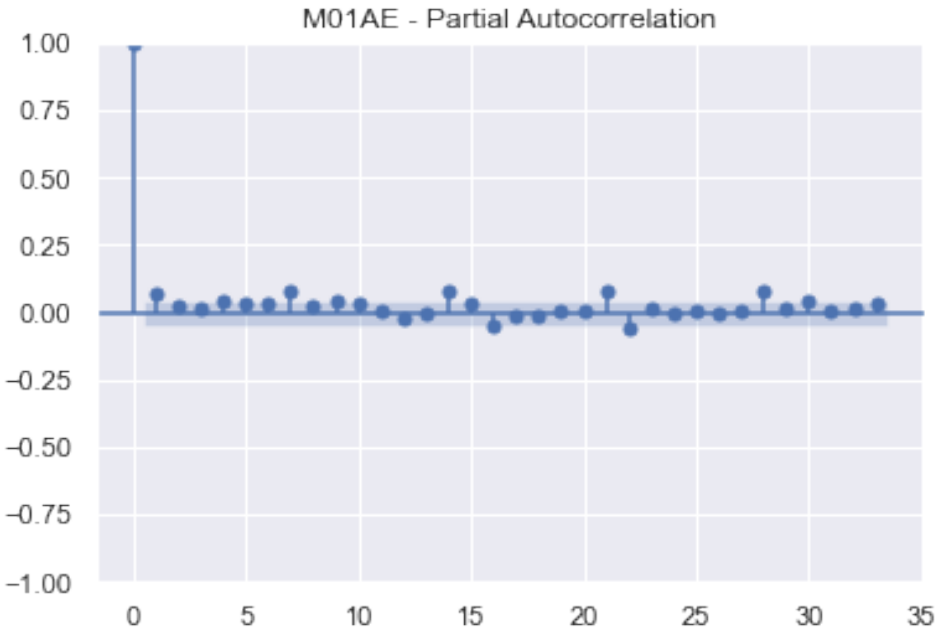


Figure 3.9: Plot of the partial autocorrelation of the *M01AE* time series

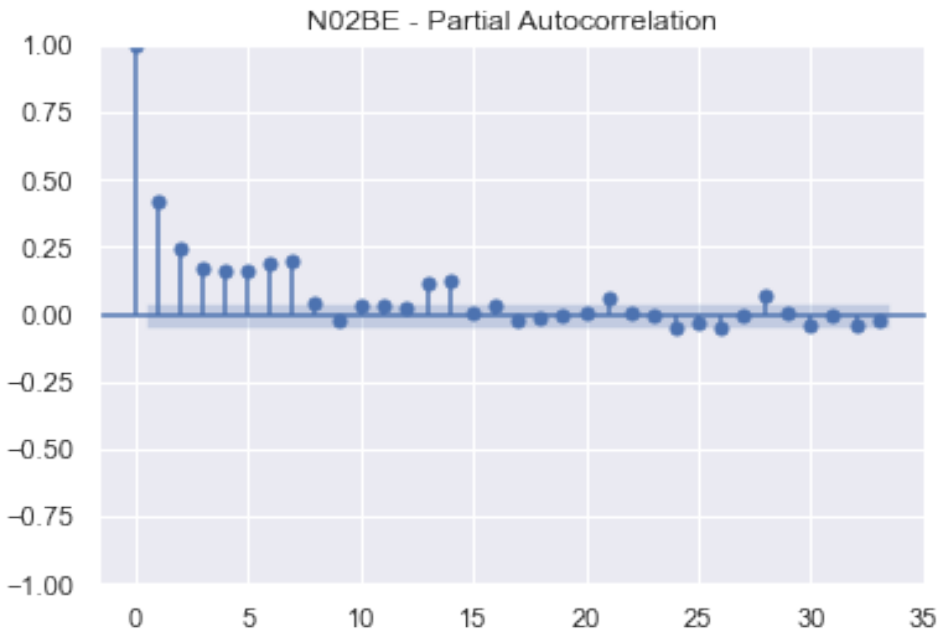


Figure 3.10: Plot of the partial autocorrelation of the *N02BE* time series

The whole autocorrelation analysis, supplemented with the plots of `ACF` and `PACF` tests of all the time series is reported in Appendix C.
What we expect is to link these two characteristics - autocorrelation and partial autocorrelation - with the behaviour of the AutoRegressive methods.

3.2. Food Demand dataset

The Food Demand dataset presents 11 sub series which are much less granular than the ones coming from the Pharma dataset. Indeed, they present a **weekly frequency**, which determines a much more smoothed graphic.

3.2.1. Time Series Decomposition

This section is very similar to Section 3.1.1. Also in this case the decomposition of the various time series in its components is essential. But before doing that, an analysis on the magnitude of the variation should be performed in order to choose among an additive and a multiplicative composition model.
Since the **weekly frequency** and the deepness of the time series (**120 steps**) are much lower than the the ones of the *Pharma dataset*, then the magnitude of the seasonal component is estimated plotting the whole time series on which we train the models. The result suggests an almost constant magnitude in the seasonal component of all the time

series under analysis. The *Aggregate* time series is reported as an example in Figure 3.11.

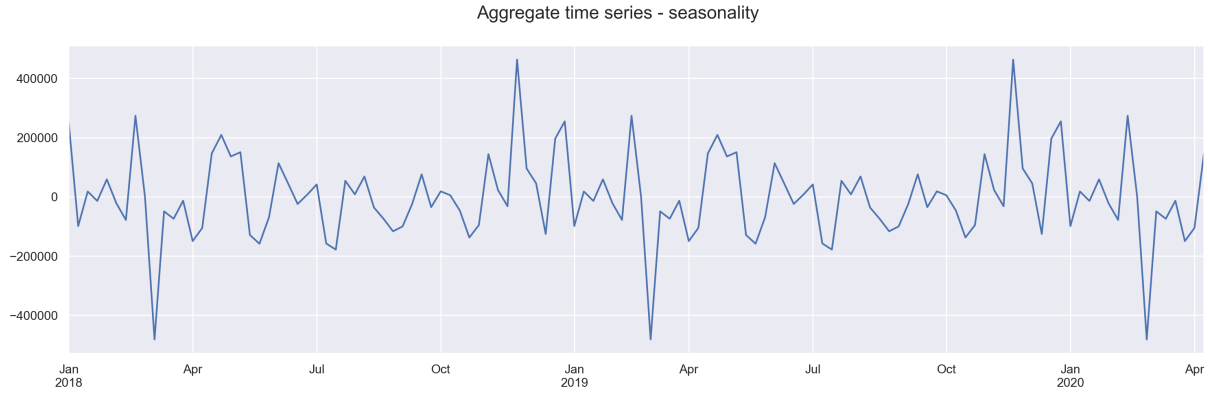


Figure 3.11: Magnitude of the seasonal component of the aggregated time series during the time span of analysis

Since this behaviour in terms of seasonal magnitude is common to every sub time series, an **additive model** is chosen to perform the decomposition of all of them. Anyway, it is important to highlight that an eventual evolution of the seasonal magnitude would be less evident on a dataset with the low frequency and the short time horizon that these time series present.
Regarding the behaviour of the time series, it is very different among the sub series, as it is clearly visible from the observed time series decomposed in Figure 3.12 and Figure 3.13.

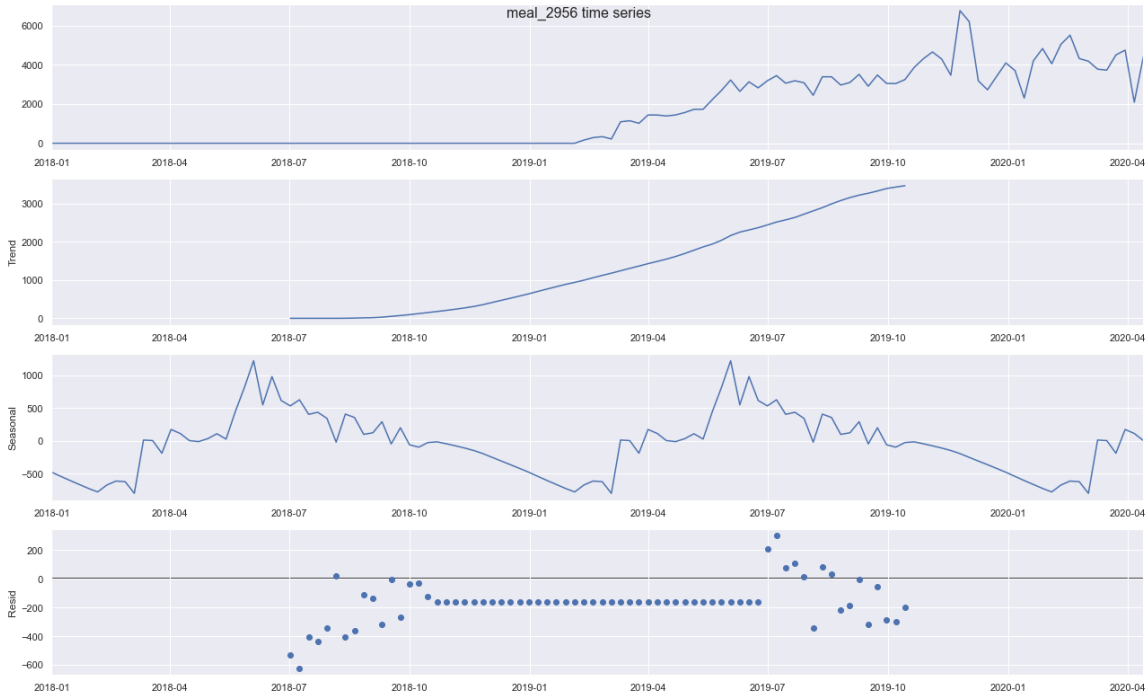


Figure 3.12: Decomposition of the *meal_2956* time series into its main components: the seasonal behaviour along the year is evident

Figure 3.13: Decomposition of the *meal_2290* item time series into its main components: the seasonal behaviour along the year is evident, but very different to the one of the previous example

Regarding the **trend component** of the time series, it is also very different in the cases under analysis. Indeed, we can see a monotonically increasing trend in *meal_2956* time series, while a much more variable trend in the case of *meal_2290* series. The main components of all the sub series under analysis are reported in Appendix C.
Also in this case, an Augmented Dickey–Fuller test is executed to verify the stationarity of the time series. This hypothesis test is assessed with the same criterion that is explained in Section 3.1.1. The results of this analysis are reported in Table 3.4.

Time series	p-value	flag stationarity
Aggregate	0.141780	0
TYPE_A	0.101892	0
TYPE_C	0.077651	0
region_56	0.150183	0
region_93	0.003958	1
meal_2290	0.000000	1
meal_2956	0.948512	0
city_473	0.000000	1
city_713	0.016371	0
email	0.000000	1
homepage	0.001366	1

Table 3.3: P-values of the Adfuller test for all the time series related to the Food Demand dataset

As it can be seen from Table 3.4, the behaviour of these time series is much variable, differently from the time series belonging to the Pharma dataset.

3.2.2. Statistical Evidences

The statistical analysis of these time series is almost identical to the previous one, so a very similar table is reported below.

Series	mean	std	cv	min	max	25%	50%	75%
Aggregate	825763	132578	16	380065	1303457	743129	810667	892979
TYPE_A	477859	76312	16	219225	752163	429081	470631	519103
TYPE_C	142021	31907	22	63982	270164	120597	136739	155809
region_56	415939	68028	16	165490	621244	373972	412951	451389
region_93	9606	2539	26	5402	19975	7943	8992	10565
meal_2290	69038	64755	94	18658	488252	45264	50433	60629
meal_2956	1587	1842	116	0	6752	0	253	3182
city_473	8367	1764	21	5271	14730	7175	8044	9094
city_713	15901	2384	15	6736	22265	14354	16042	17151
email	162932	129203	79	0	667183	61842	149136	233220
homepage	207696	103904	50	52778	654956	132589	198997	257536

Table 3.4: Main statistical indicators for all the time series related to the Food Demand dataset

Also in this case, the correlation and autocorrelation of the various time series are analysed, excluding the *Aggregate* time series. In Figure 3.14, the complete correlation matrix of the *Food Demand dataset* is reported.

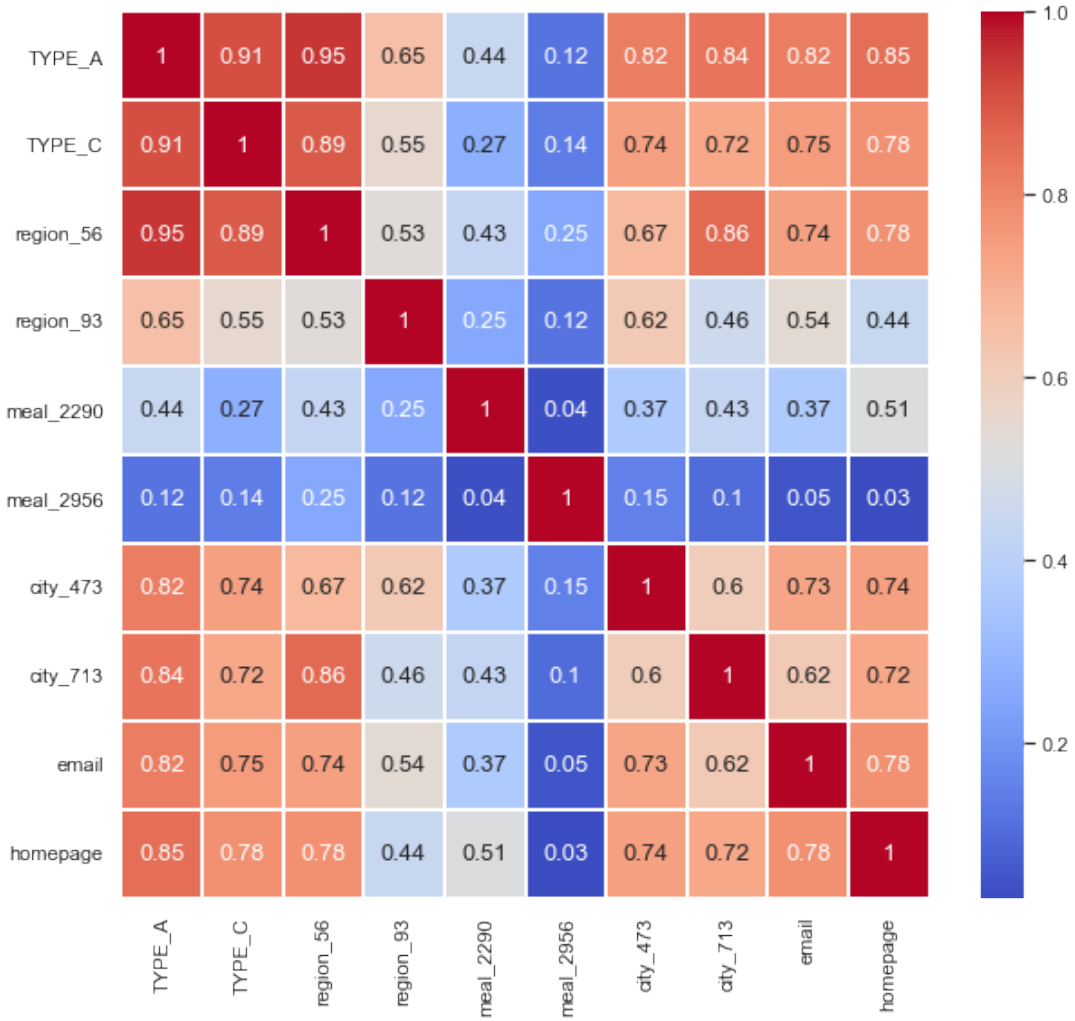


Figure 3.14: Correlation matrix of the sub series of the *Food Demand dataset*

Differently from the previous dataset, this one presents strong correlations between some series. Indeed, especially *TYPE_A*, *TYPE_C* and *region_56* time series present linear correlation coefficients around 90%.

Regarding the autocorrelation, it is quiet similar for every series, with the exception of *meal_2956* time series. Figure 3.15 shows the former one while Figure 3.16 the latter and they present the differences of these two cases.

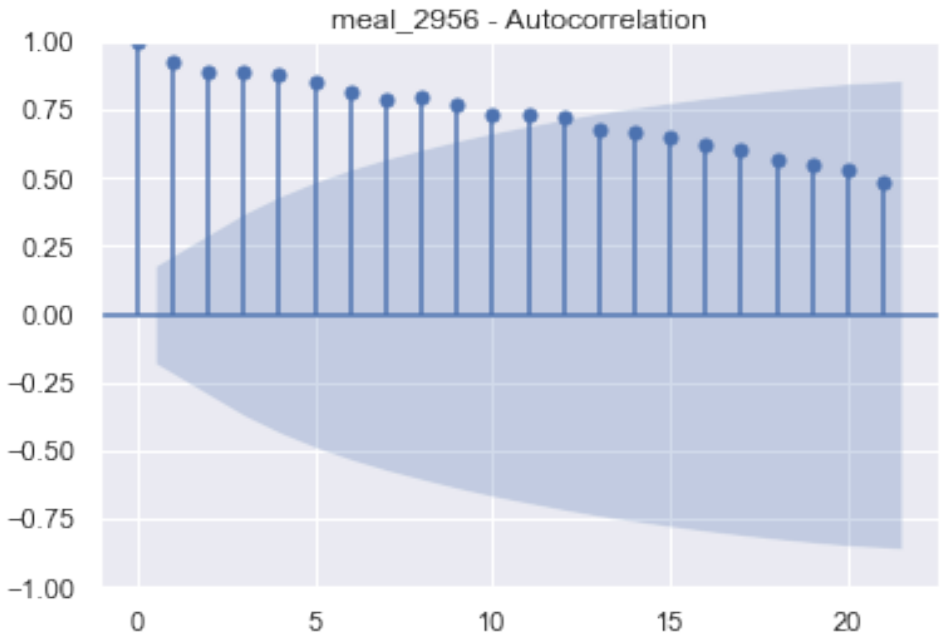


Figure 3.15: Plot of the autocorrelation of the *meal_2956* time series

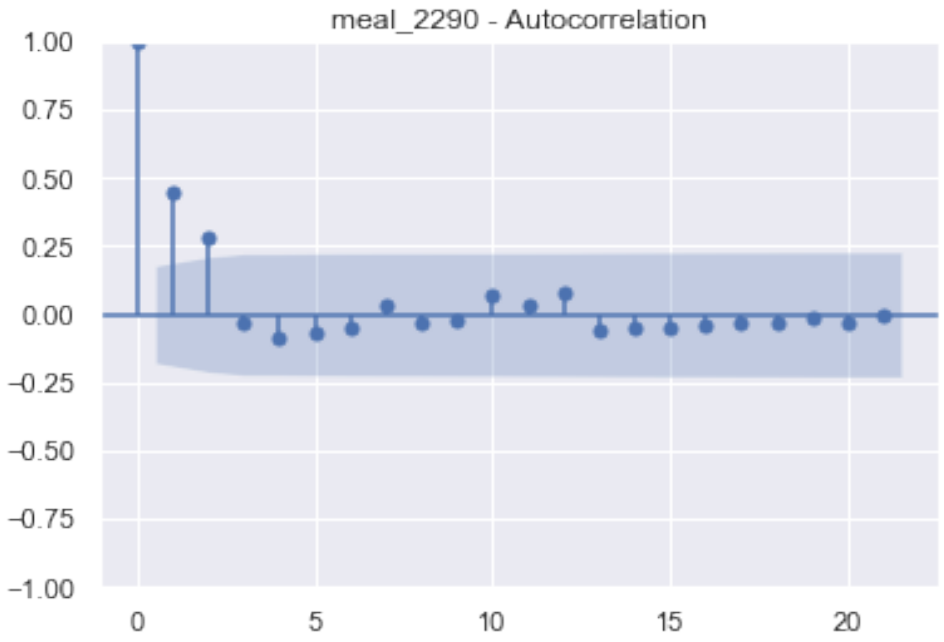


Figure 3.16: Plot of the autocorrelation of the *meal_2290* time series

The *meal_2290* time series' plot is almost always inside the confidence interval while in Figure 3.15,*meal_2956* series presents a strong positive autocorrelation.
Instead, regarding the partial autocorrelation, the phenomenon of correlation is much less

evident than before. Taking as examples the two time series mentioned above, we can see that the behaviours are aligned with the ones visible in the autocorrelation plots, even if they are less clear. This fact is shown in Figure 3.17 and Figure 3.18

Figure 3.17: Plot of the partial autocorrelation of the *meal_2956* time series

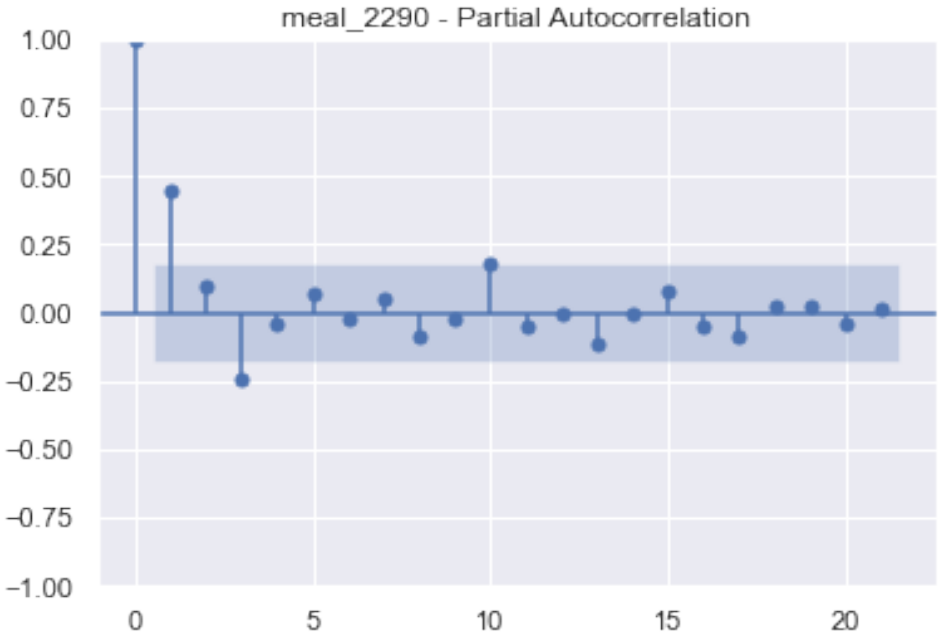


Figure 3.18: Plot of the partial autocorrelation of the *meal_2290* time series

Also in this case, the whole autocorrelation analysis, supplemented with the plots of `ACF` and `PACF` tests of all the time series is reported in Appendix C.

This chapter is dedicated to the presentation of the main results and evidences that the analyses of this thesis highlighted. The results will be showed in order to compare the performances of the models themselves and also to link them to the characteristics of the time series analysed. Then, an analysis on the aggregation of the time series is presented. Finally, an impact of the hyperparameters on the goodness of the models is assessed.

3.3. Main Results and Evidences

3.3.1. Model Assessment

Regarding the assessment of the models that have been tested, both some differences and similarities can be appreciated in focusing on a dataset or on the other one.
An important premise that should be given is that all the models are evaluated comparing their **MAE**. This choice is due to the fact that the MAE is the most largely used metric for model assessment and moreover it gives a clear idea of the magnitude of the distance between the forecasted values and the real ones. Moreover, it is selected as the preferred metric also in order to maintain coherence in model assessment, since it is the measure that is used to evaluate the best parameter configuration for each model.
First of all, we have to say that for both the two datasets the **traditional (or statistical) models**, especially the ARIMA models, are in general the ones with the **worst performance**, even if they are tuned with different parameters. Indeed, they are the best models in very few cases. A possible cause of this behaviour could be the simplicity of the model which is not able to capture all the layers of complexity of the time series.
Table 3.5 and Table 3.6 list the best model of each sub series in terms of performance, and the best regressor for the same time series, with the respective MAE values.

Time series	Best model	MAE	Best regressor	MAE
Aggregate	Prophet	13.71	KNN	14.80
M01AB	ExpSmoothing	2.19	GradientBoosting	2.21
M01AE	Prophet	1.68	KNN	1.77
N02BA	Prophet	1.46	LassoReg	1.50
N02BE	GradientBoosting	8.85	GradientBoosting	8.85
N05B	ExpSmoothing	3.22	KNN	3.35
N05C	ExpSmoothing	0.70	GradientBoosting	0.81
R03	KNN	5.42	KNN	5.42
R06	Prophet	1.85	KNN	1.93

Table 3.5: The best model and best regressor reported with their respective MAE values for each sub series of the Pharma dataset

Time series	Best model	MAE	Best regressor	MAE
Aggregate	Prophet	68816	SVR	72918
TYPE_A	Prophet	41978	SVR	47593
TYPE_C	GradientBoosting	14620	GradientBoosting	14620
region_56	GradientBoosting	35357	GradientBoosting	35357
region_93	Prophet	1373	SVR	1699
meal_2290	SVR	11741	SVR	11741
meal_2956	ExpSmoothing	1486	RidgeReg	1498
city_473	SVR	742	SVR	742
city_713	Prophet	1447	LassoReg	1493
email	Prophet	77449	SVR	81614
homepage	Prophet	70128	KNN	80103

Table 3.6: The best model and best regressor reported with their respective MAE values for each sub series of the Food Demand dataset

Regarding the Pharma dataset, for many sub series the **Prophet model** is the **best one** in terms of MAE value. In this case, also some regressors such as the Gradient Boosting and the kNN present and some traditional models such as the Exponential Smoothing

present a good performance, that in some cases slightly exceeds the one of the Prophet model.
For what regards the best model among the regression ones, in the majority of the cases, the best model is the K-Nearest Neighbors or the Gradient Boosting, together with the SVR.
It is important to highlight this fact in order derive some general considerations about the complexity of the algorithms in connection to the accuracy of the predictions that it makes for that time series. Therefore, for these sub time series, in general the models which offer the best performances are the ones that present the **highest level of complexity**. This consideration is valid both for traditional and machine learning models.

Regarding the Food Demand dataset, for many sub series (6 cases out of 12) the **Prophet model** is the one that presents the **best performance** in terms of MAE. Again, in most of the cases, the Prophet and a regression model are the best ones, while the traditional models are the less accurate ones. When the best model is a regressor, this best model is not always the same but it is much variable.
Regarding the best model among the regression ones, it is evident that in the case of this dataset, the stepwise **Support Vector Regressor** and **Gradient Boosting Regressor** are the most accurate ones in most of the cases. They are among the most complex algorithms tested for regression.
This aspect, together with the bad performances of the traditional models and the good accuracy of the forecast obtained training the Prophet model, suggests also in this case the **dominance of the most complex algorithms** when determining the accuracy of the predicted time series.

So, we have mentioned the dominance of the Prophet model in many sub series of both the two datasets that are analysed. In addition, the Prophet model usually performs well, even when it is not the optimal model. Indeed, in most of the cases under analysis, **the performances of the Prophet are very close to the ones of the best model**. This consideration is true regardless the metric that is considered and for almost every time series, and it can be appreciated looking at Figure D.1 and Figure D.2 in Appendix D.
Then, also a consideration on the **model classes performances** (Exponential Smoothing, Auto Regressive, Regression and Prophet models) is reported. As already reported, the Prophet model seems to be the class which performs better, while the Auto Regressive models are the ones with the worst MAE. In Figure 3.19 and Figure 3.20, we report the performances of the model classes in terms of MAE, considering in *Regressor* the best regression model for each dataset. In both cases, the chart is divided into sub charts in

order to make them more readable, since the time series present very different orders of magnitude.

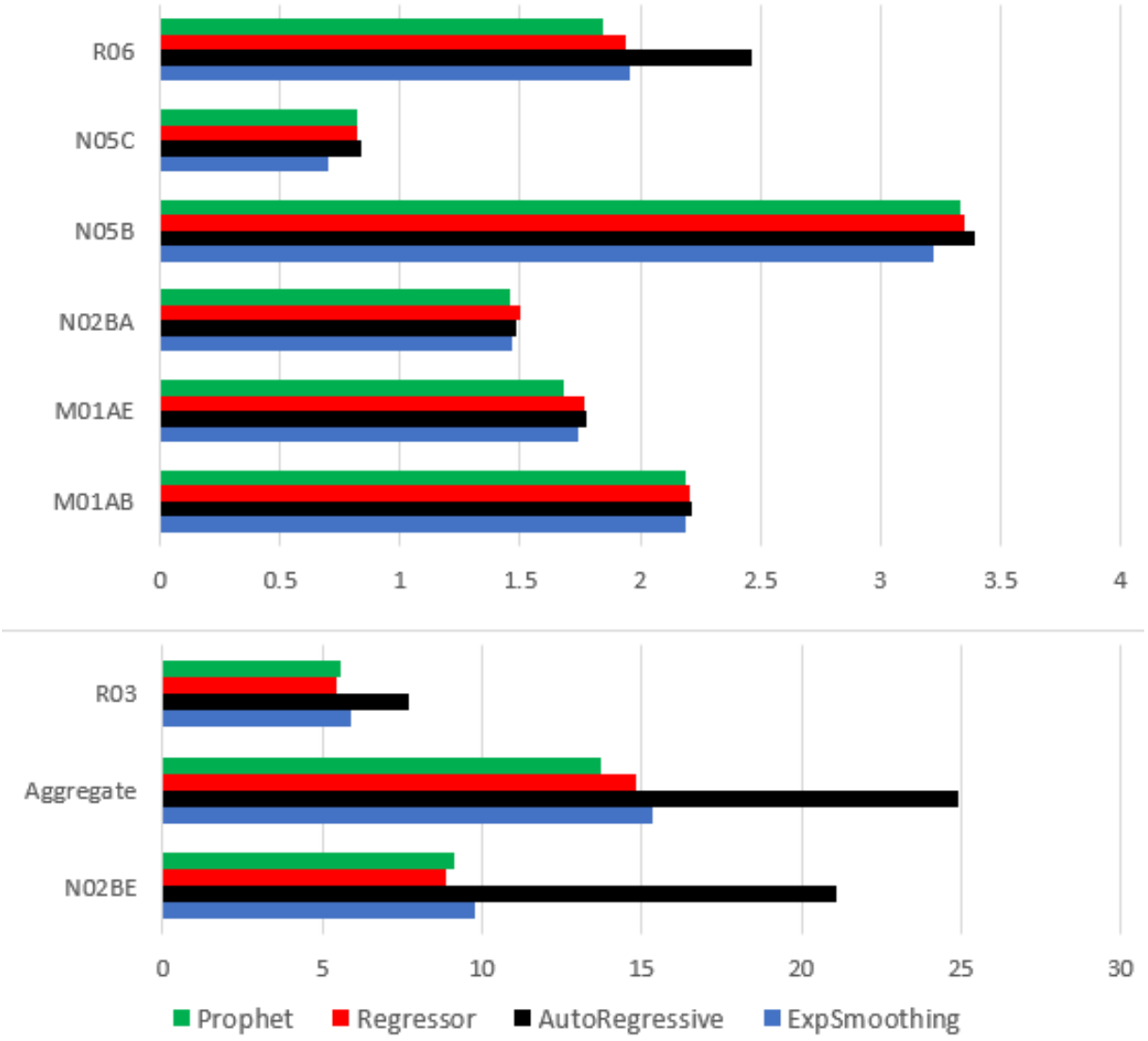


Figure 3.19: Plot of the performances of the models in terms of MAE - Pharma dataset

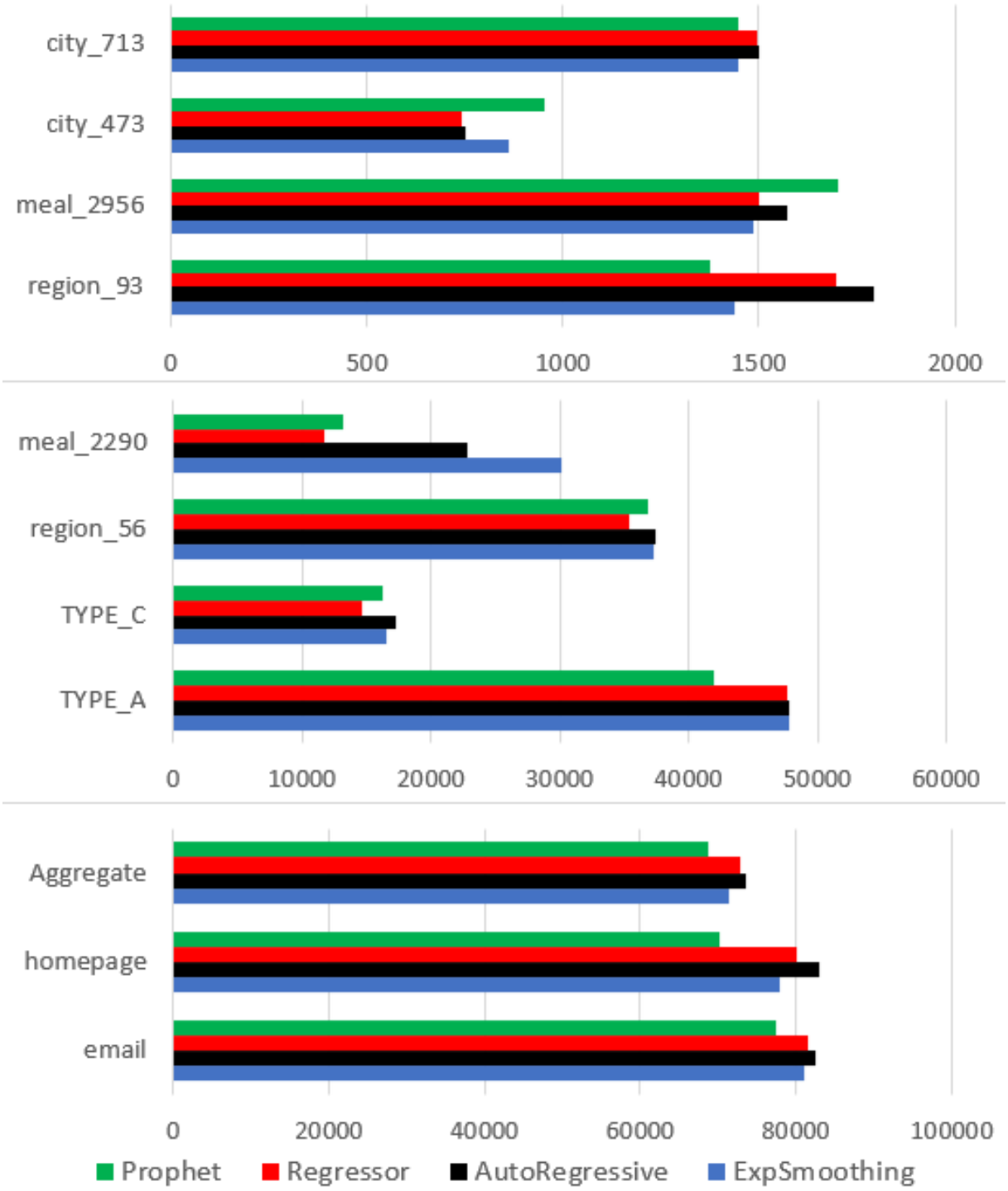


Figure 3.20: Plot of the performances of the models in terms of MAE - Food Demand dataset

Another important consideration that must be done is related to the general results that we obtained at the end of the analysis. The question we would like to ask is whether the

predictions are satisfying and if they can considered as reliable in any business application. This consideration can be done considering the adjusted MAPE as a metric in order to understand which is the average percentage error of each time series. The following Table 3.7 and Table 3.8 report the optimal model identified for each sub series of Pharma and Food Demand dataset respectively.

Time series	Best model	MAPE_adj
Aggregate	Prophet	28.83
M01AB	ExpSmoothing	68.18
M01AE	Prophet	114.57
N02BA	Prophet	86.67
N02BE	GradientBoosting	33.87
N05B	ExpSmoothing	63.09
N05C	ExpSmoothing	49.06
R03	KNN	110.17
R06	Prophet	76.64

Table 3.7: The best model reported with its respective adjusted MAPE values for each sub series of the Pharma dataset

Time series	Best model	MAPE_adj
Aggregate	Prophet	9.35
TYPE_A	Prophet	9.62
TYPE_C	GradientBoosting	10.95
region_56	GradientBoosting	8.58
region_93	Prophet	27.42
meal_2290	SVR	14.36
meal_2956	ExpSmoothing	25.11
city_473	SVR	8.32
city_713	Prophet	9.75
email	Prophet	187.34
homepage	Prophet	48.45

Table 3.8: The best model reported with its respective adjusted MAPE values for each sub series of the Food Demand dataset

For what regards the Pharma dataset, in general its sub series don't show a good adjusted MAPE, since in many cases it is around 50%. Therefore, the predictions that are made, related to this dataset, are not affordable for any business application. The only case in which they are quite good is the one of the *Aggregate* time series. In any case, it is important to highlight the fact that the adjusted MAPE should be compared with the variability of the time series itself.
Instead, regarding the Food Demand dataset, its sub series generally present a quite good adjusted MAPE, around the 10%. This aspect makes the predictions quite reliable, since the average relative error is low. Anyway, some of the sub series such as *email* and *homepage* time series, present a very bad adjusted MAPE, which makes the predictions too little accurate to be affordable for business application.

Then, we report below the plots of a good prediction and a bad one in terms of adjusted MAPE for both the two datasets.

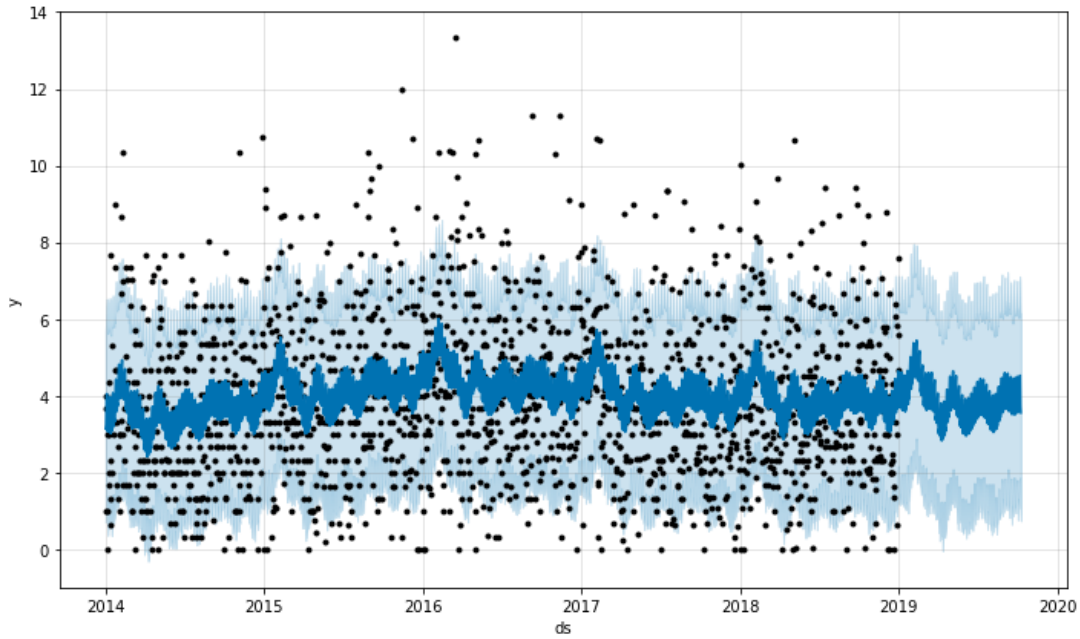


Figure 3.21: Plot of the *M01AE* time series and its future predictions - Pharma dataset

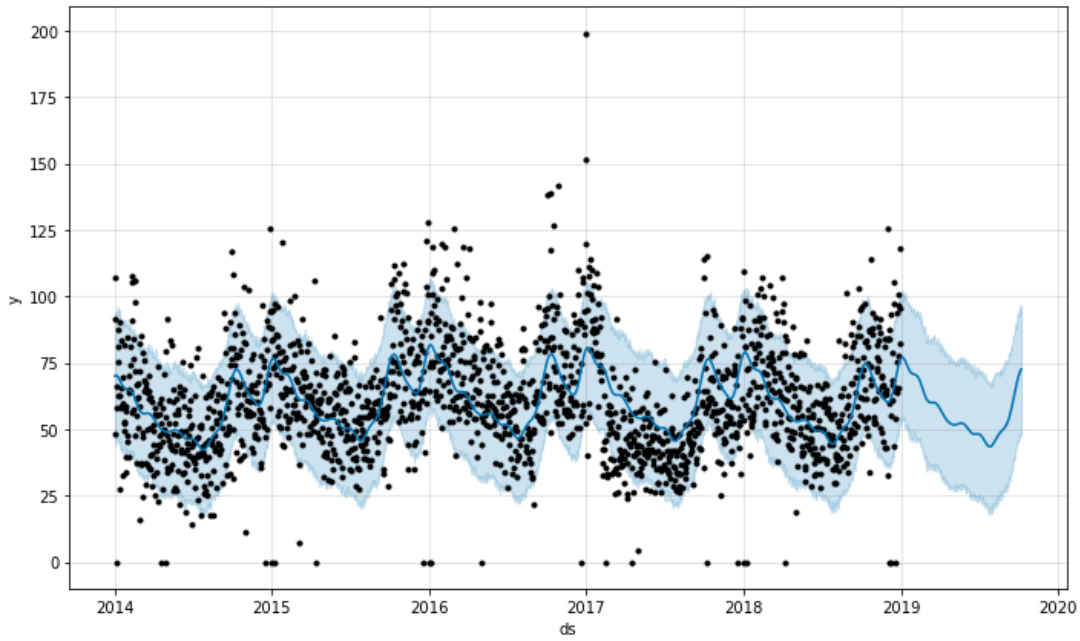


Figure 3.22: Plot of the *Aggregate* time series and its future predictions - Pharma dataset

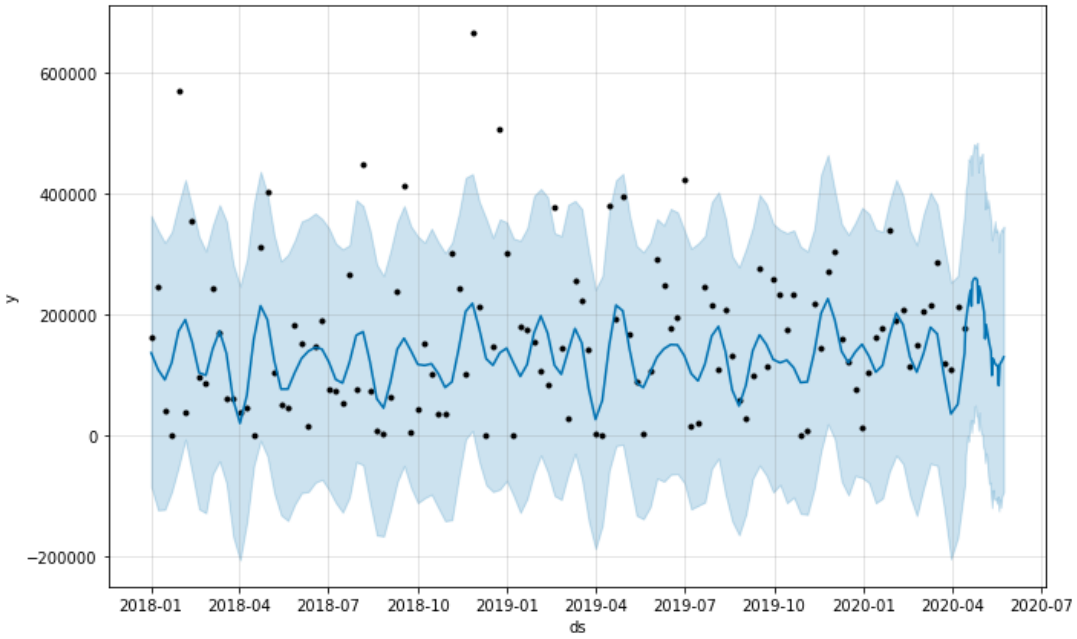


Figure 3.23: Plot of the *email* time series and its future predictions - Food Demand dataset

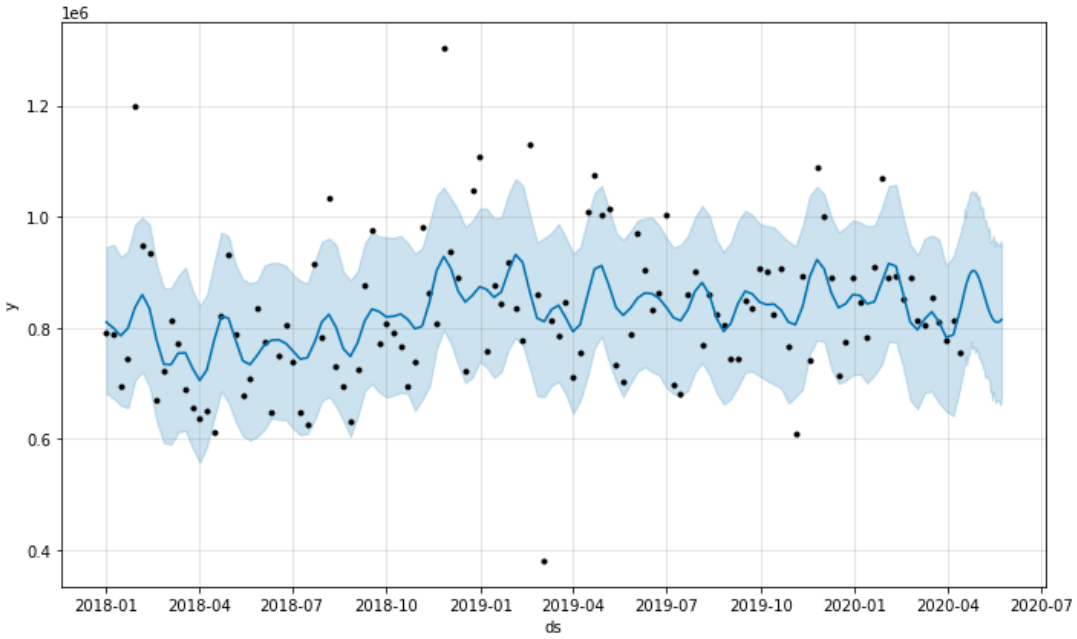


Figure 3.24: Plot of the *Aggregate* time series and its future predictions - Food Demand dataset

The light blue band in these graphs represents the range between a lower and an upper bound that constitute an uncertainty interval and are responsible for the residual compo-

nent of the time series (white noise). Then we can notice that the series which have the worst performances are those that presents the thickest dark blue line and largest band, as it was expected.

Additionally, it is important to underline the impact of the metrics that are considered to evaluate the accuracy of the predictions. Indeed, as it can be seen from the complete results in Figure D.1 and Figure D.2 in Appendix D, **switching the selected metric** from MAE to MAPE or MSD, **we obtain different results**. Therefore, the choice of the most appropriate metric to evaluate the accuracy of the predictions is fundamental and it must be taken considering the business requirements and the objective of the analysis.

3.3.2. Impact of Hyperparameters

In this section we analyse the impact that the choice of the set of hyperparameters has on the model's performances. Due to the considerations concerning the Prophet method, reported in Section 3.3.1, the impact of the hyperparameter tuning is assessed on the Prophet model only. A summary of the results of this analysis is reported in Table 3.9 and Table 3.10 below. It is important to specify that the improvements and degradation rates reported below are calculated referring to a **baseline value**, which is the MAE calculated fitting the model without **any parameter**. The baseline value is indicated as *default*. Then, the average, minimum and maximum values of MAE describe the dataset that is obtained from the Grid search tuning of the hyperparameters.

Series	default	avg	min	max	improve_%	degrad_%
Aggregate	16.05	15.60	14.21	18.64	11.46	16.14
M01AB	2.25	2.24	2.19	2.43	2.67	8.00
M01AE	1.67	1.76	1.68	1.88	−0.60	12.57
N02BA	1.56	1.50	1.46	1.74	6.41	11.54
N02BE	10.28	10.26	9.12	12.64	11.28	22.96
N05B	3.45	3.55	3.34	4.46	3.19	29.28
N05C	0.86	0.87	0.82	1.06	4.65	23.26
R03	6.08	6.37	5.60	7.12	7.89	17.11
R06	1.87	1.94	1.85	2.23	1.07	19.25

Table 3.9: MAE values of the Prophet model, compared with the default one, Pharma dataset

Series	default	avg	min	max	improve_%	degrad_%
Aggregate	88511	80663	68816	90942	22.25	2.75
TYPE_A	4896	46959	40941	58840	16.27	20.34
TYPE_C	2110	19298	16308	23807	22.89	12.56
region_56	41223	40752	36907	44464	10.47	7.86
region_93	1809	1875	1363	2866	24.67	58.42
meal_2290	29702	29731	13230	49046	55.46	65.13
meal_2956	2611	2204	1650	3636	36.81	39.25
city_473	1050	1080	953	1342	9.19	27.88
city_713	1568	1816	1438	4863	8.30	210.08
email	79776	80935	77449	87666	2.92	9.89
homepage	73928	74261	68462	83374	7.39	12.78

Table 3.10: MAE values of the Prophet model, compared with the default one, Food Demand dataset

As it is evident from these two tables, the impact of hyperparameters tuning is remarkable. The choice of the best model improves the performance evaluated on the Pharma dataset by 5.3% and a bad choice of the hyperparameters degradates the performance by the 17.8% on average. Regarding the Food Demand dataset, we obtain an average improvement around the 19.7% and an average degradation rate equal to 42.4% with respect to the baseline.
The improvements and degradation rates registered for the two datasets are represented in Figure 3.25 and Figure 3.26.

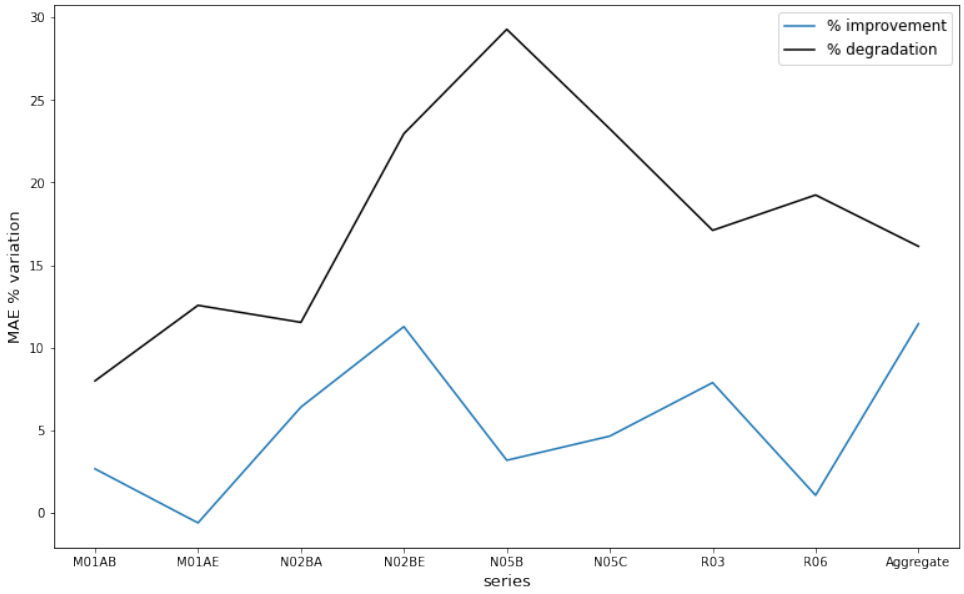


Figure 3.25: Plot of improvement and degradation rates - Pharma dataset

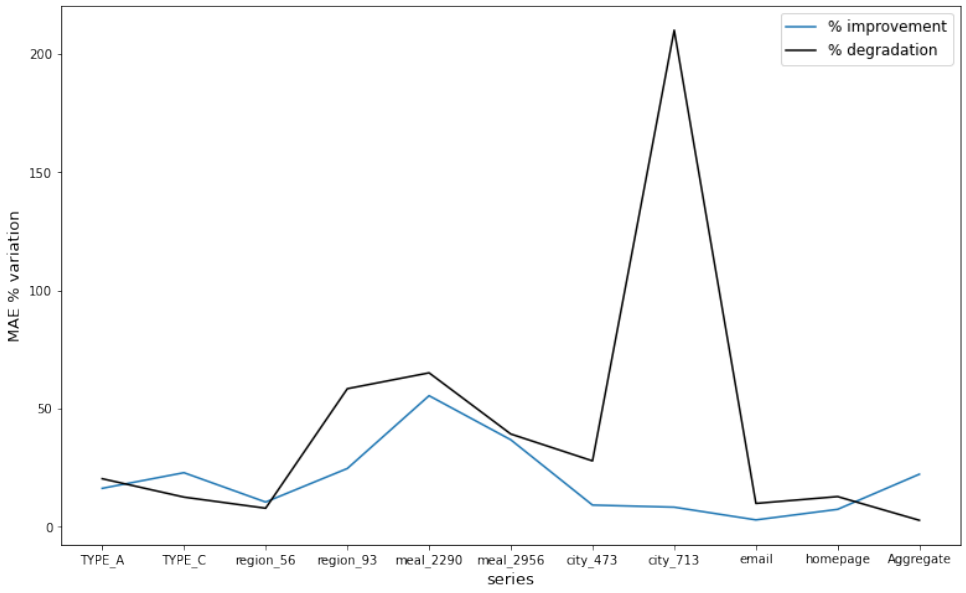


Figure 3.26: Plot of improvement and degradation rates - Food Demand dataset

As it is visible in the graphs, a big improvement does not always correspond to a big degradation and vice versa: this phenomenon cannot be spotted in advance. Moreover, in general, the **degradation rate** in case of a bad hyperparameter choice is **greater than** the **improvement rate** in case of an optimal tuning. These considerations can be made for both the two datasets.

3.3.3. Impact of Sub Series on Aggregated Time Series

This section is aimed at reporting the results that emerge from the analysis of a time series which results from the aggregation of its sub series, observation by observation. The objective and the procedure of this analysis is explained in Figure 2.6 of Section 2.5. We report in the following Table 3.11 the main evidences.

Metric	*Configuration A*	*Configuration B*
MAE	13.90	13.71
MSE	333.65	324.77
RMSE	18.27	18.02

Table 3.11: Main results of the analysis on the aggregation

As it can be seen from the table, regardless the metric that is considered, *Configuration B* of the experiment seems to produce the most accurate predictions, even if the difference between the two considerations is not so big. Therefore, it seems to be **convenient to forecast the aggregate time series** in the best possible way instead of dividing it into sub series and to predict them separately.
This result could be due to some particular behaviours that have a more marked effect in the aggregate time series. Moreover, another reason could be the compensation of the errors during the initial aggregating phase.
In Figure 3.27 the predictions (*Configuration A* and *Configuration B*) are represented in a graph.

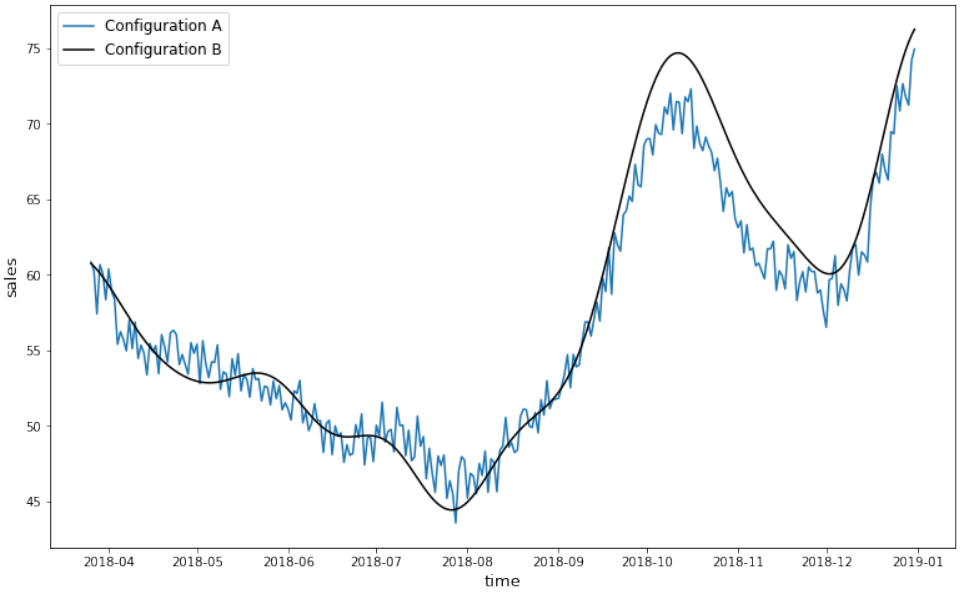


Figure 3.27: Plot of *Configuration A* and *Configuration B* predictions

The two curves have the same shape in terms of trend, even if *Configuration A* presents a higher noise, as it was exepcted considering the construction of the time series.

3.3.4. Model - Dataset Specificity Assessment

In this section the main results in relation with the characteristics of the time series are reported, in order to give a general overview of the behaviour of the various models on the different types of time series and to spot eventual correlations and patterns.
Table 3.12 and Table 3.13 show the MAPE of the best model and the coefficient of variation of the initial time series.

Time series	Best model	MAPE_adj	cv
Aggregate	Prophet	28.83	35
M01AB	ExpSmoothing	68.18	55
M01AE	Prophet	114.57	54
N02BA	Prophet	86.67	61
N02BE	Prophet	44.38	51
N05B	ExpSmoothing	63.09	65
N05C	ExpSmoothing	49.06	189
R03	KNN	110.17	11
R06	Prophet	76.64	85

Table 3.12: The best model reported with its respective adjusted MAPE values for each sub series of the Pharma dataset

Time series	Best model	MAPE_adj	cv
Aggregate	Prophet	9.35	16
TYPE_A	Prophet	9.62	16
TYPE_C	GradientBoosting	10.95	22
region_56	GradientBoosting	8.58	16
region_93	Prophet	27.42	26
meal_2290	SVR	14.36	94
meal_2956	ExpSmoothing	25.11	116
city_473	SVR	8.32	21
city_713	Prophet	9.75	15
email	Prophet	187.34	79
homepage	Prophet	48.45	50

Table 3.13: The best model reported with its respective adjusted MAPE values for each sub series of the Food Demand dataset

First of all, the best model seems not to be correlated with the MAPE or with the CV of the time series. We can say that in some cases, a much variable time series presents quite good MAPE: for example, *meal_2956* shows a not so bad MAPE with respect to

the other series, in the light of a time series with a coefficient of variation equal to 116%. A similar observation can be done for *meal_2290* and *N05C* series.
Instead, in the case of *email*, *M01AE* and *R03* series, a very bad MAPE is produced, even if the variance of the starting time series is not so high with respect to its average value.
Anyway, another aspect should be added to this consideration, that is the random noise component of the time series. Indeed, from a theoretical viewpoint, the residual is the part of the variance that the model is not able to explain with a trend or seasonal component of the time series. The best models identified - Exponential Smoothing and Prophet - are, theoretically speaking, the best ones in capturing the trend and seasonal components. Regarding this aspect, if we observe the Food Demand dataset, we see that the *email* time series, which is the one with the less evident trend and seasonality components, is also the time series which presents the worst performance in terms of MAPE. Moreover, in general the trend and the seasonality components are very clear observing the Food Demand dataset, while they are not looking at the Pharma dataset. In general, the MAPE is quite bad for the Pharma data, while it is low when looking at the Food Demand one. Therefore, we can say that the models, and the Prophet more specifically, make better previsions when the trend and seasonality components are marked, so when the white noise present a relatively low variance.
Then, a consideration that must be made is related to a specific sub series. The *meal_2290* time series is the one that presents the maximum coefficient of variation and the sub series that shows the highest autocorrelation. It is the only case in which the optimal model is the Exponential Smoothing, that is probably able to give much importance to the past observations tuning the hyperparameters in an effective way.

3.4. Conclusions and Takeaways

In this section we report the main points that have emerged as results of this research work and the main **takeaways** that should be considered in making time series forecasting.
First of all, we have to say that this kind of analysis - and consequently its result - is very **specific**: it strongly depends on the time series that is forecasted and its main characteristics. At the same time, it is very hard to find *a priori* a correlation between the best model in terms of accuracy and the characteristics of the time series, probably due to the **very complex** set of factors that drive the forecasting optimization.
Due to this consideration, a deep analysis must be made on each dataset in order to identify the best model and the optimal parameters it should be tuned with. In particular,

the best practice is to test each model tuned with its optimal hyperparameters on each dataset series. Due to the high number of possible combinations of a dataset and a model, the best idea is to **make the whole process automatic**, in order to test all the couples and therefore to obtain the best performances.
Moreover, the **impact of hyperparameters tuning** is remarkable: thus, a great effort should be put also on the tuning process, consequently identifying the optimal set and to obtain the best results from each single model.
Regarding the models, the **dominance of complex algorithms** such as the Prophet model and some regressors such as the Gradient Boosting, the Support Vector and the K-Nearest Neighbors Regressors is evident. In particular, a dominance of the **Prophet model** emerges from the analysis: even if the optimal model is not the Prophet, the latter usually obtains performances that are not so far from the ones of the optimal configuration.
Another important point that must be highlighted is the **strong dependency of the predictions' accuracy from the metric** that is chosen to evaluate them: changing the metric, we can also draw different conclusions.

In general, what emerges is the importance of **investing in the Forecasting Science**, trying to make even more accurate predictions.
From a business perspective, the great advantage of owning these accurate time series projections is remarkable. Thanks to forecasting, the planning of the company's resources in terms of time, money and materials sees a great improvement, thus enabling their optimal allocation and avoiding waste: the **improvement in terms of efficiency** is significant. On the other side, very precise sales predictions enables a good demand forecasting, which makes the company able to meet customers' requests better: the **upgrade in effectiveness** is great too.
Being able to forecast the future with a sufficiently high level of confidence makes every strategic choice as if it were be based on real data. The **decision-making process** sees a huge gain, thus ensuring a strong **competitive advantage** to the company.

4 | Future Developments

This chapter is dedicated to the description of some possible developments of this thesis that could be put in practise in the next future. They are both linked with both the most technical aspects and the ones related to the business applications of this thesis.

4.1. Experimental Models

As a first possible future development of this thesis there is the implementation of some experimental methods which proved in literature to be particularly efficient in learning sequences. These methods belong to the **Long Short-Term Memory (LSTM)** class, which is a particular category of **Recurrent Neural Networks**. Recurrent Neural Networks suffer from short-term memory.
The LSTM models try to overcome this problem and therefore they are very much effective for the time series analysis and forecasting, as it is well explained in [14], [15] and [28].

- **Long Short-Term Memory (LSTM)** networks are recurrent neural networks capable of learning order dependence in sequence prediction problems. LSTMs are a complex area of deep learning, particularly efficient in the analysis and forecasting of sequential data. The core concept of LSTM's are the cell state, together with it's various gates. It can be considered as the "memory" of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it's way to later time steps, reducing the effects of short-term memory. The gates can learn what information is relevant to keep or forget during training. This model is implemented in the function `LSTM` of the module `keraslayers`.
- **Gated Recurrent Unit (GRU)** So now we know how an LSTM work, let's briefly look at the GRU. The GRU is the new generation of Recurrent Neural networks and is similar to an LSTM. "It is a special type of optimized LSTM-based recurrent neural network" [28]. GRU has not got the cell state and it used the hidden state to transfer information. It has been implemented through the function `GRU` coming

from `keras.layers.recurrent`.

These particular deep learning algorithms can be tested with the sliding window technique, in addiction to the ones that are already tested in this thesis, as well as many other traditional regression models which are not currently tested.
Moreover, it could be a good idea to add some additional features not related to time in order to evaluate regression and Prophet models using also other information that can impact the time series values. For example, the type of product that is sold or the geographic area in which the transaction happens.

4.2. Optimal Hyperparameter Tuning Techniques

Since the parameter configuration heavily impacts the performance of the model, one possible development of this thesis could be the tuning of the hyperparameters of each model as effectively as possible. As it is stated in literature [21], [32].
Bayesian optimization is a global optimization method for noisy functions. It creates a probabilistic model of the function mapping from hyperparameter values to the objective. It evaluates iteratively promising hyperparameter configuration and it updates it. In this way it is able to find the optimum. It tries to balance exploration (hyperparameters for which the outcome is most uncertain) and exploitation (hyperparameters expected close to the optimum).
In practice, Bayesian optimization generally obtains **better results in fewer evaluations** compared to Grid search and random search, due to the ability to evaluate the quality of experiments at each iteration. Indeed, in the Grid search method, the individual experiments are executed by building multiple models with various hyperparameter values. All these experiments are independent of each other. Since each experiment is performed independently, the Grid search method is not able to use the information from one experiment to improve the next experiment.
Regarding the performances of the two algorithms, the bayesian optimizer is efficient because it selects hyperparameters in an informed and smart way. Indeed, prioritizing hyperparameters that appear more promising from the past results, this method can find the best hyperparameters in a shorter time period than grid search. On the other side, as it is easy to imagine, the bayesian optimizer for hyperparameter tuning increases the **complexity** that is necessary to learn the method. This is the reason why, also in this case, in order to improve the performance of the methods, we could exploit the cloud computing in order to deal with more complex algorithms such as the bayesian optimizer.

Figure 4.1[1] shows the way the bayesian optimizer explores the space of the hyperparameters.

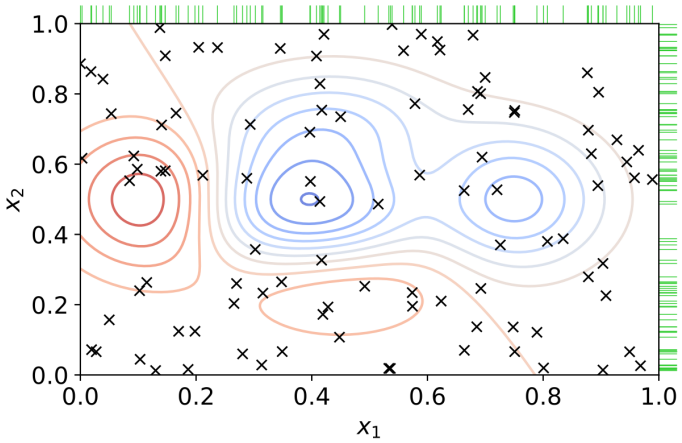


Figure 4.1: Bayesian optimizer: it smartly explores the space of potential choices

Comparing it with Figure 2.3 and Figure 2.4 in Chapter 2, the great potential improvement that this tuning method can bring to the accuracy of the methods is evident.

4.3. Multivariate Time Series Forecasting

A further analysis that has not been executed in this research work is the **multivariate time series analysis**, which can be performed efficiently on the Pharma dataset. Indeed, all the models that are tested in this thesis can be adapted to a multivariate time series forecasting problem. This allows us to spot any possible connection that exists between the sub series, that are not considered since in this thesis the sub series are analysed separately.

For example, in this thesis the Pharma dataset has been analysed and predicted considering the sub series as independent, but since they describe the sales of the various products, a correlation between them can be expected. Predicting all the time series at the same time with the multivariate forecasting would enable the identification of all these possible correlations.

[1]Source: www.wikipedia.org

4.4. Model Testing Automation

As it is clear from the results reported in Chapter 3.2.2, the accuracy of a forecasting process and the optimal model for performing it are very much volatile. Indeed, even if some correlations have been spotted between the accuracy of the model and the characteristics of the time series, it is still not possible to know in advance which is the optimal model and which is its optimal parameter configuration.
Thus, in order evaluate the performance of all the models and to choose the best one, all the models should be tested and all the hyperparameters belonging to their specific state space are tuned for each model. This requires many trials and many evaluations, one for each different experiment.
Due to the **complexity** of this process, it cannot be performed manually by humans. It must be **automated** as much as possible in order to make the machine perform the whole process and to reduce as much as possible the human intervention. The latter would ideally be necessary only in the initial and final phases: during the setting of the models and hyperparameter spaces and during the choice of the best model, given the performance of all the experiments made.
Since this process presents a very high level of complexity - especially in computational terms - a fully automated system that relies on cloud computing is necessary in order to increase the capacity of the system and not to run into issues due to computational reasons. Some companies are working in this field providing cloud-based solutions for data science automation.
One of the most largely used platforms is **DataRobot** [2]. As stated in [2] and [3], it is a service which offers solutions for managing the whole life cycle of Data Science projects based on Artificial Intelligence. From a practical viewpoint, with "the whole life cycle" we intend the execution of the whole process, starting from the data exploration and preparation, going to the models development, the hyperparameter tuning, the performance evaluation and the predictions creation. For the whole process, a user interface is available so that there is not need to write any code but only the business level insights must be managed by humans. The system relies on cloud computing and it is based on Python programming language and its related libraries.

4.5. Cloud Computing for Data Science

As already stated in Section 4.4, the increase in complexity of algorithms, layers and correlations must be managed since the machine learning and forecasting models bring

[2]https://www.datarobot.com/

much computational effort to the system. A fully automated system that relies on cloud computing is necessary in order to increase the capacity of the system and not to run into issues due to computational reasons. Some companies are working in this field providing cloud-based solutions for data science automation.
Moreover, since the amount of data processed every day by company is continuously growing, it is necessary to scale up the system relying on cloud-based solutions.
Some examples are the **DataRobot** platform already mentioned in Section 4.4 and the **DataBricks** platform [3]. As it is stated in [4], the latter offers a service for the development of data science projects and it offers an interface with the mos largely used instruments for Data Science such as SQL for data extraction, Python for data manipulation and model development and Tableau for data visualization.
In general, due to the well-noted benefits of the cloud computing, it represents the **future of the Data science** for business [9], [10].

[3]https://databricks.com/it/

Bibliography

[1] Measuring forecast accuracy: The complete guide. URL https://www.relexsolutions.com/resources/measuring-forecast-accuracy/.

[2] Datarobot documentation, . URL https://docs.datarobot.com/.

[3] Datarobot website, . URL https://www.datarobot.com/.

[4] Databricks website. URL https://databricks.com/it/.

[5] Prophet: Forecasting at scale. URL https://facebook.github.io/prophet/.

[6] Time series analysis: Definition, types, techniques, and when it's used. URL https://www.tableau.com/learn/articles/time-series-analysis.

[7] Time series. URL https://en.wikipedia.org/wiki/Time_series.

[8] Augmented dickey–fuller test. URL https://en.wikipedia.org/.

[9] Why cloud computing is important in data science?, . URL https://www.geeksforgeeks.org/why-cloud-computing-is-important-in-data-science/.

[10] The importance of data science with cloud computing, . URL https://www.edureka.co/blog/importance-data-science-cloud-computing/.

[11] Correlation. URL https://en.wikipedia.org/wiki/Correlation.

[12] Exponential smoothing - oracle documentation. URL https://docs.oracle.com/en/database/oracle/.

[13] Forecasterautoreg documentation. URL https://joaquinamatrodrigo.github.io/skforecast/api/ForecasterAutoreg.html.

[14] Lstm vs gru in recurrent neural network: A comparative study, . URL https://analyticsindiamag.com/lstm-vs-gru-in-recurrent-neural-network-a-comparative-study/.

[15] Illustrated guide to lstm's and gru's: A step by step explanation, . URL https://towardsdatascience.com/.

[16] Gradientboostingregressor documentation. URL https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html.

[17] Hyperparameter optimization. URL https://en.wikipedia.org/wiki/Hyperparameter_optimization.

[18] scikit-learn, machine learning in python. URL https://scikit-learn.org/stable/.

[19] Autoregressive models, statsmodels documentation. URL https://www.statsmodels.org/devel/generated/statsmodels.tsa.arima.model.ARIMA.html.

[20] The 4 layers of data analytics for business transformation, 2019. URL https://medium.com/.

[21] J. Bergstra. Algorithms for hyper-parameter optimization.

[22] J. Brownlee. Time series forecasting as supervised learning, 2016. URL https://machinelearningmastery.com/time-series-forecasting-supervised-learning/.

[23] J. Brownlee. How to decompose time series data into trend and seasonality, 2017. URL https://machinelearningmastery.com/decompose-time-series-data-trend-seasonality/.

[24] J. Brownlee. Multistep time series forecasting with lstms in python, 2017. URL https://machinelearningmastery.com/.

[25] J. Brownlee. Multistep time series forecasting with lstms in python, 2020. URL https://machinelearningmastery.com/.

[26] J. Brownlee. Time series forecasting with prophet in python, 2020. URL https://machinelearningmastery.com/.

[27] J. Brownlee. How to use xgboost for time series forecasting, 2020. URL https://machinelearningmastery.com/xgboost-for-time-series-forecasting/.

[28] B. C. M. M. M. J. T. F. R. A. A. M. Cardoso. Comparing lstm and gru models to predict the condition of a pulp paper press. 2021.

[29] F. K. Coşkun Hamzaçebi, Diyar Akay. Comparison of direct and iterative artificial neural network forecast approaches in multi-periodic time series forecasting. *Expert Systems with Applications: An International Journal*, 36(2):3839–3844, 3 2009.

[30] A. K. S. H. S. Hota, Richa Handa. Time series data prediction using sliding window based rbf neural network. *International Journal of Computational Intelligence Research*, 2017.

[31] J. E. O. Joaquín Amat Rodrigo. Skforecast: time series forecasting with python and scikit-learn, 2021. URL `https://www.cienciadedatos.net/documentos/py27-time-series-forecasting-python-scikitlearn.html`.

[32] J. Jordan. Hyperparameter tuning for machine learning models, 2017. URL `https://www.jeremyjordan.me/hyperparameter-tuning/`.

[33] E. Kannanaikkal. Food demand forecasting, 2020. URL `https://www.kaggle.com/datasets/kannanaikkal/food-demand-forecasting`.

[34] A. Kumar. Autoregressive (ar) models with python examples, 2022. URL `https://vitalflux.com/autoregressive-ar-models-with-python-examples/`.

[35] S. Loukas. Time-series forecasting: Predicting stock prices using facebook's prophet model, 2020. URL `https://medium.com/`.

[36] G. Myers. How data analysis improve decision making, 2019. URL `https://reflectivedata.com/how-data-analysis-improve-decision-making/`.

[37] P. Pathak. How to create an arima model for time series forecasting in python, 2020. URL `https://www.analyticsvidhya.com/`.

[38] A. Plummer. Different types of time series decomposition. URL `https://towardsdatascience.com/`.

[39] S. Prabhakaran. Arima model complete guide to time series forecasting in python, 2021. URL `https://www.machinelearningplus.com/time-series/`.

[40] G. A. Rob J Hyndman. *Forecasting: Principles and Practice*. PLOS One, 2018.

[41] B. L. Sean J. Taylor. Forecasting at scale. *PeerJ Preprints*, 2017.

[42] B. L. Sean J. Taylor. Prophet: Forecasting at scale, 2017. URL `https://research.facebook.com/blog/2017/02/prophet-forecasting-"at-scale/`.

[43] V. A. Spyros Makridakis, Evangelos Spiliotis. *Statistical and Machine Learning forecasting methods: Concerns and ways forward*. OTexts, 2018.

[44] C. Vercellis. *Business Intelligence: Data Mining and Optimization for Decision Making*. Wiley, 2009.

[45] M. Zdravkovic. Pharma sales data analysis and forecasting, 2020. URL https://www.kaggle.com/data/.

[46] A. Zhang. How to build exponential smoothing models using python: Simple exponential smoothing, holt, and holt-winters, 2018. URL https://medium.datadriveninvestor.com/.

A | Datasets

Figure A.1: Complete Pharma dataset

	datum	M01AB	M01AE	N02BA	N02BE	N05B	N05C	R03	R06	Year	Month	Hour	Weekday Name
0	01/02/2014	0.00	3.67	3.4	32.40	7.0	0.0	0.0	2.0	2014	1	248	Thursday
1	01/03/2014	8.00	4.00	4.4	50.60	16.0	0.0	20.0	4.0	2014	1	276	Friday
2	01/04/2014	2.00	1.00	6.5	61.85	10.0	0.0	9.0	1.0	2014	1	276	Saturday
3	01/05/2014	4.00	3.00	7.0	41.10	8.0	0.0	3.0	0.0	2014	1	276	Sunday
4	01/06/2014	5.00	1.00	4.5	21.70	16.0	2.0	6.0	2.0	2014	1	276	Monday
5	01/07/2014	0.00	0.00	0.0	0.00	0.0	0.0	0.0	0.0	2014	1	276	Tuesday
6	01/08/2014	5.33	3.00	10.5	26.40	19.0	1.0	10.0	0.0	2014	1	276	Wednesday
7	01/09/2014	7.00	1.68	8.0	25.00	16.0	0.0	3.0	2.0	2014	1	276	Thursday
8	01/10/2014	5.00	2.00	2.0	53.30	15.0	2.0	0.0	2.0	2014	1	276	Friday
9	01/11/2014	5.00	4.34	10.4	52.30	14.0	0.0	1.0	0.2	2014	1	276	Saturday
10	01/12/2014	2.00	0.66	2.5	12.00	8.0	0.0	1.0	1.0	2014	1	276	Sunday
11	1/13/2014	7.34	7.66	6.2	52.00	9.0	0.0	7.0	1.0	2014	1	276	Monday
12	1/14/2014	6.00	1.33	12.3	33.70	6.0	1.0	0.0	2.0	2014	1	276	Tuesday
13	1/15/2014	4.00	2.34	5.0	26.70	12.0	2.0	3.0	3.0	2014	1	276	Wednesday
14	1/16/2014	6.00	2.00	4.3	28.30	19.0	1.0	5.0	0.0	2014	1	276	Thursday
15	1/17/2014	2.00	3.68	8.3	20.40	15.0	0.0	6.0	3.0	2014	1	276	Friday
16	1/18/2014	1.00	5.33	5.8	43.20	15.0	4.0	7.0	2.0	2014	1	276	Saturday
17	1/19/2014	4.33	4.00	4.0	14.10	4.0	0.0	1.0	1.0	2014	1	276	Sunday
18	1/20/2014	6.00	3.34	3.3	11.90	18.0	2.0	12.0	3.0	2014	1	276	Monday
19	1/21/2014	2.00	3.34	4.0	42.00	15.0	2.0	0.0	1.0	2014	1	276	Tuesday
20	1/22/2014	7.00	3.00	7.0	18.10	10.0	0.0	5.0	2.0	2014	1	276	Wednesday
21	1/23/2014	4.00	4.67	5.2	26.20	2.0	3.0	1.0	3.0	2014	1	276	Thursday
22	1/24/2014	4.67	7.34	7.0	19.60	13.0	1.0	0.0	0.0	2014	1	276	Friday
23	1/25/2014	6.00	9.00	5.0	37.80	18.0	0.0	5.0	1.0	2014	1	276	Saturday
24	1/26/2014	4.33	1.68	0.0	24.00	4.0	0.0	0.0	0.0	2014	1	276	Sunday
25	1/27/2014	4.34	4.67	1.2	23.60	17.0	2.0	1.0	3.0	2014	1	276	Monday
26	1/28/2014	6.00	3.34	2.8	13.38	22.0	5.0	1.0	5.0	2014	1	276	Tuesday
27	1/29/2014	5.33	4.00	2.0	14.00	10.0	1.0	1.0	2.0	2014	1	276	Wednesday
28	1/30/2014	3.02	1.34	2.4	25.50	7.0	1.0	3.0	2.0	2014	1	276	Thursday
29	1/31/2014	1.00	2.68	7.1	26.90	9.0	0.0	1.0	0.0	2014	1	276	Friday

Figure A.2: Complete Food Demand dataset

	id	week	center_id	meal_id	checkout_price	base_price	email_for_prom	homepage_feat	num_orders	city_code	region_code	center_type	op_area
0	1379560	1	55	1885	136.83	152.29	0	0	177	647	56	TYPE_C	2.0
1	1466964	1	55	1993	136.83	135.83	0	0	270	647	56	TYPE_C	2.0
2	1346989	1	55	2539	134.86	135.86	0	0	189	647	56	TYPE_C	2.0
3	1338232	1	55	2139	339.50	437.53	0	0	54	647	56	TYPE_C	2.0
4	1448490	1	55	2631	243.50	242.50	0	0	40	647	56	TYPE_C	2.0
5	1270037	1	55	1248	251.23	252.23	0	0	28	647	56	TYPE_C	2.0
6	1191377	1	55	1778	183.36	184.36	0	0	190	647	56	TYPE_C	2.0
7	1499955	1	55	1062	182.36	183.36	0	0	391	647	56	TYPE_C	2.0
8	1025244	1	55	2707	193.06	192.06	0	0	472	647	56	TYPE_C	2.0
9	1054194	1	55	1207	325.92	384.18	0	1	676	647	56	TYPE_C	2.0
10	1469367	1	55	1230	323.01	390.00	0	1	823	647	56	TYPE_C	2.0
11	1029333	1	55	2322	322.07	388.00	0	1	972	647	56	TYPE_C	2.0
12	1446016	1	55	2290	311.43	310.43	0	0	162	647	56	TYPE_C	2.0
13	1244647	1	55	1727	445.23	446.23	0	0	420	647	56	TYPE_C	2.0
14	1378227	1	55	1109	264.84	297.79	1	0	756	647	56	TYPE_C	2.0
15	1181556	1	55	2640	282.33	281.33	0	0	108	647	56	TYPE_C	2.0
16	1313873	1	55	2306	243.50	340.53	0	0	28	647	56	TYPE_C	2.0
17	1067069	1	55	2126	486.00	485.00	0	0	28	647	56	TYPE_C	2.0
18	1058482	1	55	2826	306.58	305.58	0	0	188	647	56	TYPE_C	2.0
19	1240935	1	55	1754	289.12	289.12	0	0	485	647	56	TYPE_C	2.0
20	1044821	1	55	1971	259.99	320.13	1	1	798	647	56	TYPE_C	2.0
21	1149039	1	55	1902	388.03	446.23	0	0	14	647	56	TYPE_C	2.0
22	1263416	1	55	1311	196.94	320.13	0	0	176	647	56	TYPE_C	2.0
23	1323882	1	55	1803	117.40	188.24	0	0	150	647	56	TYPE_C	2.0
24	1338119	1	55	1558	583.03	610.13	1	0	162	647	56	TYPE_C	2.0
25	1188372	1	55	2581	583.03	612.13	1	0	312	647	56	TYPE_C	2.0
26	1440008	1	55	1962	582.03	612.13	1	0	231	647	56	TYPE_C	2.0
27	1336534	1	55	1445	628.62	627.62	0	0	13	647	56	TYPE_C	2.0
28	1242186	1	55	2444	627.62	626.62	0	0	15	647	56	TYPE_C	2.0
29	1012819	1	55	2867	628.62	626.62	0	0	13	647	56	TYPE_C	2.0
30	1038567	1	55	1525	243.50	282.33	0	0	15	647	56	TYPE_C	2.0
31	1325272	1	55	2704	243.50	282.33	0	0	13	647	56	TYPE_C	2.0
32	1412058	1	55	2304	484.03	484.03	0	0	13	647	56	TYPE_C	2.0
33	1040403	1	24	1885	136.83	136.83	0	0	1498	614	85	TYPE_B	3.6
34	1012104	1	24	1993	134.83	135.83	0	1	1243	614	85	TYPE_B	3.6
35	1277150	1	24	2539	135.86	135.86	0	0	391	614	85	TYPE_B	3.6
36	1323742	1	24	2139	340.50	435.53	0	0	55	614	85	TYPE_B	3.6
37	1417386	1	24	2631	247.38	248.38	0	0	67	614	85	TYPE_B	3.6
38	1367249	1	24	1248	247.35	248.35	0	0	81	614	85	TYPE_B	3.6
39	1168714	1	24	1778	177.51	179.51	0	0	377	614	85	TYPE_B	3.6
40	1351400	1	24	1062	184.36	182.36	0	0	514	614	85	TYPE_B	3.6
41	1496539	1	24	2707	190.18	188.18	0	0	1176	614	85	TYPE_B	3.6
42	1372248	1	24	1207	319.19	380.27	0	0	256	614	85	TYPE_B	3.6
43	1292501	1	24	1230	336.65	390.00	0	0	243	614	85	TYPE_B	3.6
44	1346854	1	24	2322	322.07	387.09	0	0	324	614	85	TYPE_B	3.6
45	1291642	1	24	2290	298.82	296.82	0	0	998	614	85	TYPE_B	3.6
46	1226536	1	24	1727	412.28	411.28	0	0	702	614	85	TYPE_B	3.6
47	1165348	1	24	1109	243.47	273.54	1	0	877	614	85	TYPE_B	3.6
48	1079470	1	24	2577	280.33	281.33	0	0	134	614	85	TYPE_B	3.6
49	1192984	1	24	2640	282.33	281.33	0	0	285	614	85	TYPE_B	3.6

Figure A.3: Final Food Demand dataset

	ts	TYPE_A	TYPE_C	region_56	region_93	meal_2290	meal_2956	city_473	city_713	email	homepage	Aggregate
0	2018-01-01	452098	140360	410398	7745	45326	0.0	6937	17290	162518.0	176567	792261
1	2018-01-08	463522	132468	386677	8189	37337	0.0	7154	16146	247037.0	244139	787084
2	2018-01-15	402058	108770	352075	7300	41190	0.0	7071	14188	40862.0	148255	695262
3	2018-01-22	432580	119622	401614	7197	45211	0.0	6105	16899	0.0	145571	743529
4	2018-01-29	710639	217394	621244	12092	488252	0.0	12576	21045	570427.0	654956	1198675
5	2018-02-05	560098	165589	481414	9882	301437	0.0	10485	20790	39430.0	455092	947288
6	2018-02-12	561359	154903	468844	10574	348976	0.0	10549	20639	355050.0	414347	934803
7	2018-02-19	400030	100860	306344	7944	103994	0.0	8534	13176	97241.0	211596	670518
8	2018-02-26	424272	121716	370842	7036	49948	0.0	6639	16164	85412.0	162787	723243
9	2018-03-05	480031	135430	404033	9750	45774	0.0	8226	16594	244199.0	234610	811825
10	2018-03-12	463268	118426	387746	7261	44177	0.0	8777	17956	170067.0	216243	772225
11	2018-03-19	406303	104096	360135	7528	43616	0.0	6142	14225	62325.0	164645	690259
12	2018-03-26	380525	114090	337674	6255	47065	0.0	6130	13446	60394.0	142010	656102
13	2018-04-02	379481	102404	342571	6441	45297	0.0	5357	14172	38715.0	134676	636981
14	2018-04-09	380719	102012	335174	7132	42909	0.0	6244	12570	46407.0	119106	651719
15	2018-04-16	359787	94700	304779	6504	43130	0.0	6252	15048	0.0	109347	611515
16	2018-04-23	487002	138177	412670	9366	167540	0.0	8587	16895	313177.0	231678	820285
17	2018-04-30	540223	179714	449938	11981	44936	0.0	11136	17609	403828.0	375212	932560
18	2018-05-07	458578	127362	418849	7938	45985	0.0	7182	16362	103362.0	238684	787196
19	2018-05-14	399855	106770	343524	8383	42970	0.0	6531	12943	52173.0	90569	677834
20	2018-05-21	414517	111278	353482	8046	47123	0.0	7457	13918	45032.0	95470	707013
21	2018-05-28	472764	166158	419419	8625	49492	0.0	8400	16422	182844.0	235073	834111
22	2018-06-04	443974	144738	376962	8992	46094	0.0	8311	16366	152271.0	209750	773271
23	2018-06-11	383165	100116	330060	6583	45220	0.0	6631	13203	15136.0	96662	647341
24	2018-06-18	448320	119108	379823	8802	45909	0.0	8139	15197	146388.0	198508	749583
25	2018-06-25	474935	145957	411789	8157	47377	0.0	7311	16646	190202.0	212785	805805
26	2018-07-02	423986	135076	376055	8348	31650	0.0	8209	13474	77421.0	132877	740014
27	2018-07-09	383408	100838	327861	7301	29339	0.0	6705	12129	73672.0	92419	648863
28	2018-07-16	371004	96631	314994	8022	27584	0.0	6500	11420	53640.0	86311	625414
29	2018-07-23	543228	148367	521995	9545	30341	0.0	7212	20586	266826.0	214581	915399
30	2018-07-30	440750	152663	381016	9437	47227	0.0	7384	15933	76391.0	130415	783214
31	2018-08-06	580994	198699	424954	13879	54565	0.0	14504	15113	447702.0	347092	1034202
32	2018-08-13	419756	116367	354000	8105	51646	0.0	8133	14036	73907.0	106763	730936
33	2018-08-20	397345	106138	339144	7990	47624	0.0	6686	13521	7750.0	96665	693603
34	2018-08-27	357278	114785	297522	7708	48335	0.0	6646	11249	3545.0	52778	630458
35	2018-09-03	411566	123938	344046	9672	47999	0.0	7322	11488	62972.0	69433	724865
36	2018-09-10	517737	145347	416510	10416	48151	0.0	10417	17097	238532.0	282882	877853
37	2018-09-17	573087	155121	467819	11544	265032	0.0	10442	17491	412547.0	331220	974566
38	2018-09-24	439949	136380	382713	9700	48281	0.0	7229	14849	4740.0	160909	770964
39	2018-10-01	463860	138641	410294	10343	46630	0.0	7681	15758	44233.0	161836	807159

B | Experiments Outline Algorithm

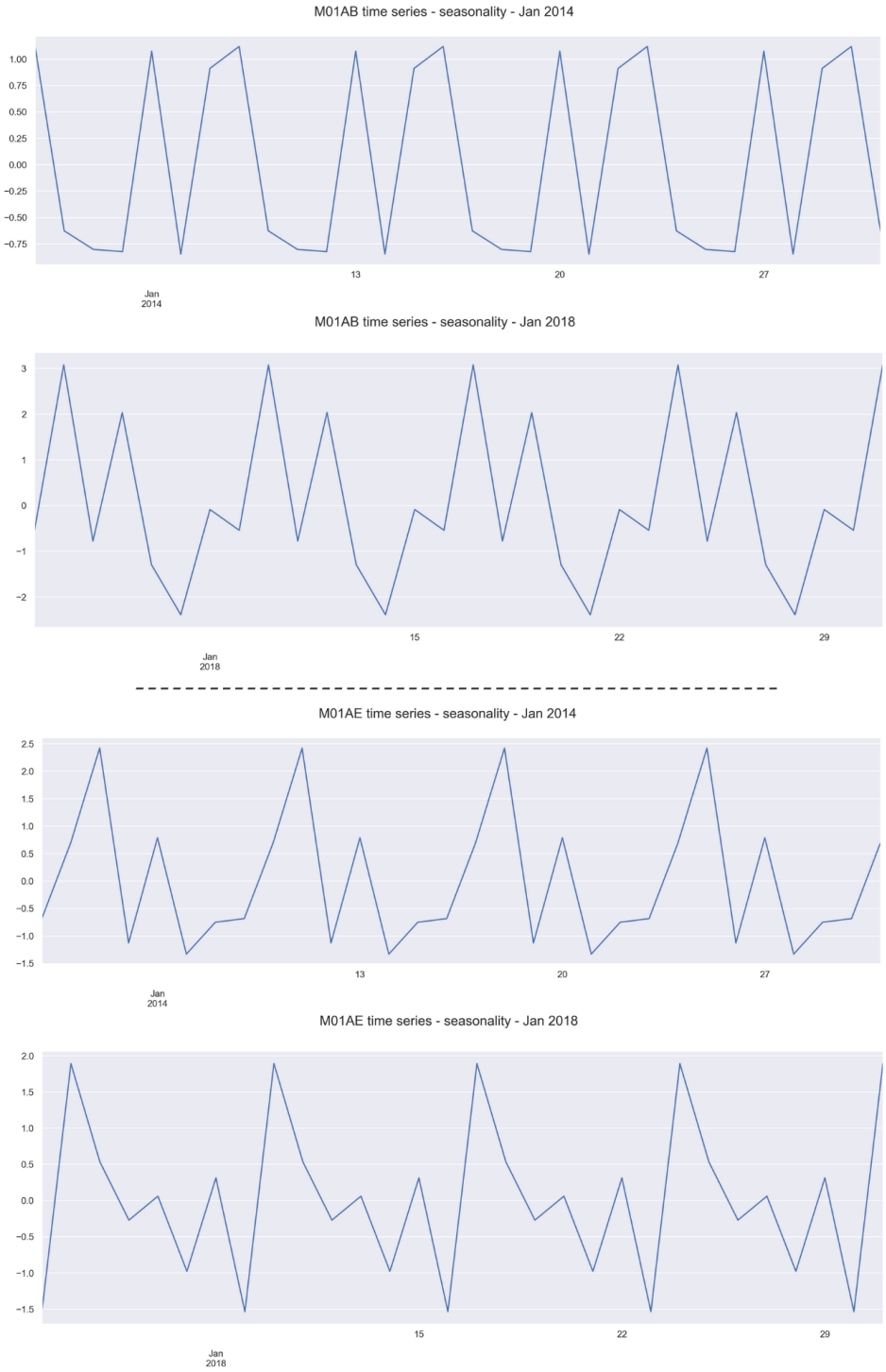
Figure B.1: Experiments' outline algorithm - main cycle (Pharma dataset)

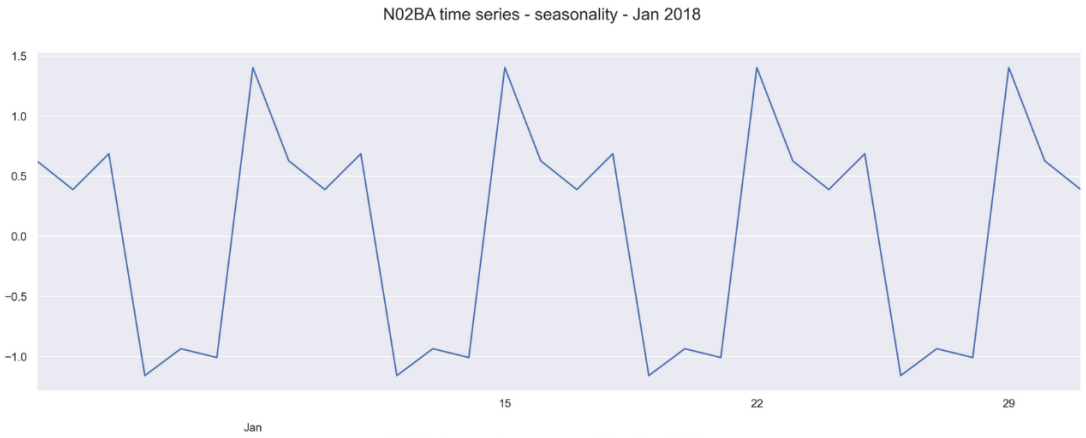
```
MODELS = {
    'exponential smoothing': ['ExponentialSmoothing'],
    'arima' : ['auto_arima'],
    'regressor': ['LinearRegression', 'SVR', 'Ridge', 'Lasso', 'KNeighborsRegressor'],
    'prophet': ['prophet']
}

SUB_SERIES = ['M01AB', 'M01AE', 'N02BA', 'N02BE', 'N05B', 'N05C', 'R03', 'R06', 'Aggregate']

RESULTS = {}

i = 1

for series in SUB_SERIES:
    print('Sub series: {0}'.format(series))

    for model_class in MODELS:
        print('Model class: {0}'.format(model_class))

        for model in MODELS[model_class]:

            # Load the train dataset and the real dataset
            original_train_data = pd.read_csv('data/train_data.csv')
            original_real_data = pd.read_csv('data/real_data.csv')

            # Load the results file to check if the experiment has been already tried
            with open("results_pharma_dataset.json") as infile:
                RESULTS = json.load(infile)

            try:

                # if RESULTS[i] is defined, experiment i already esists and it should be skipped
                _ = RESULTS[str(i)]
                print(' - Experiment ', str(i), ' already exists. Skip it - ')
```

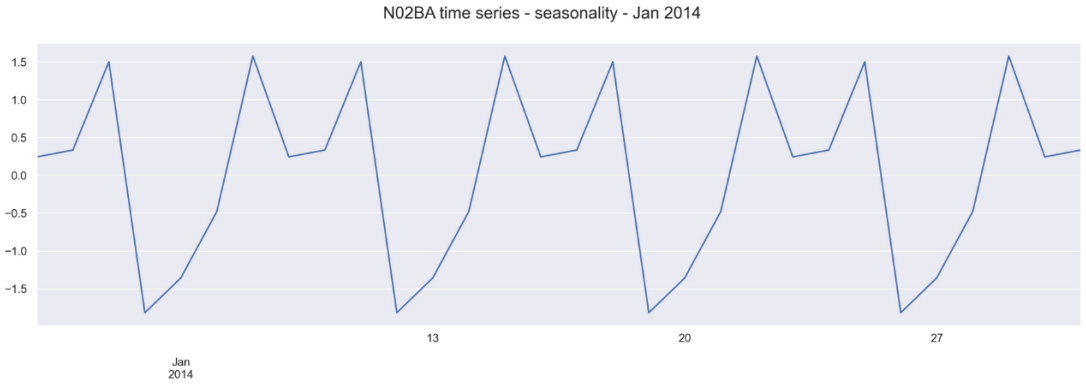
```
            except KeyError:

                # otherwise the experiments still has not been executed, so it should be done
                print(' - Experiment ', str(i), ' does not exist yet. Execute it - ')

                # Preprocess train and real data in the same way, depending on the model class considered
                train_data, real_data = preprocessing(original_train_data, original_real_data, series, model_class)

                # Tune the hyperparameters, fit the model, make the predictions and evaluate them
                parameters, mae, mape, mape_adj, msd, mse, rmse = fit_predict_evaluate(train_data, real_data, model)

                # Return the best model with its hyperparameters and its metrics
                RESULTS[str(i)] = {}
                RESULTS[str(i)][series] = {}
                RESULTS[str(i)][series][model] = {'parameters': parameters,
                            'mae': mae, 'mape':mape, 'mape_adj':mape_adj, 'msd':msd, 'mse':mse, 'rmse':rmse}

            # Save the results writing in the file
            update_json(RESULTS)

            i = i + 1

        print(' - - Change model class - - ')

    print('\n - - - Change sub series - - - \n')
```

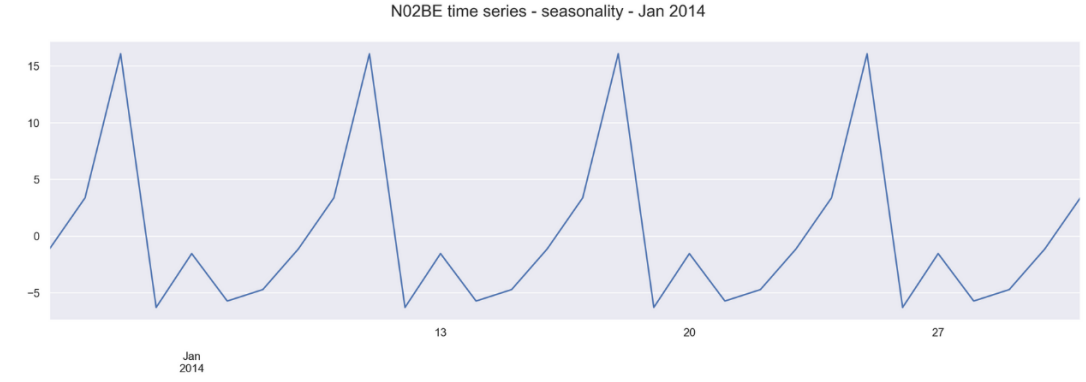
C | Time Series Graphs

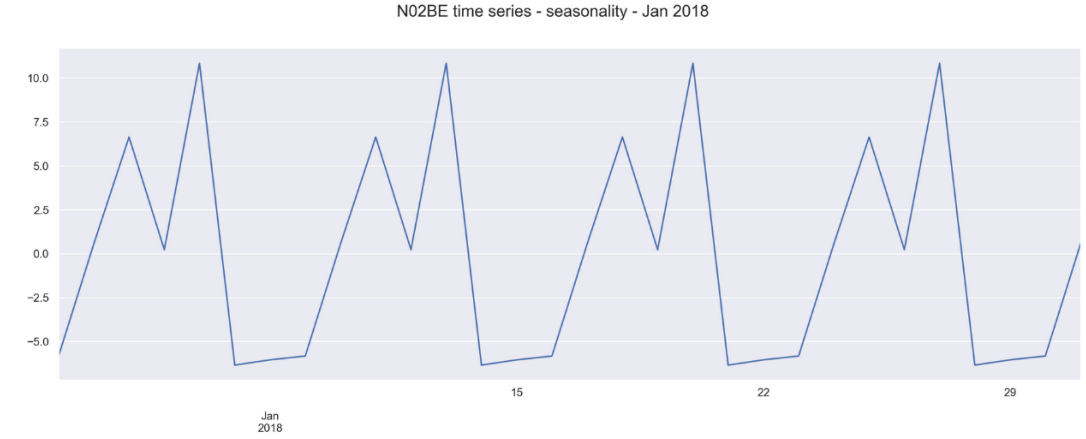
Figure C.1: Weekly periodicity of all the sub series - Pharma dataset

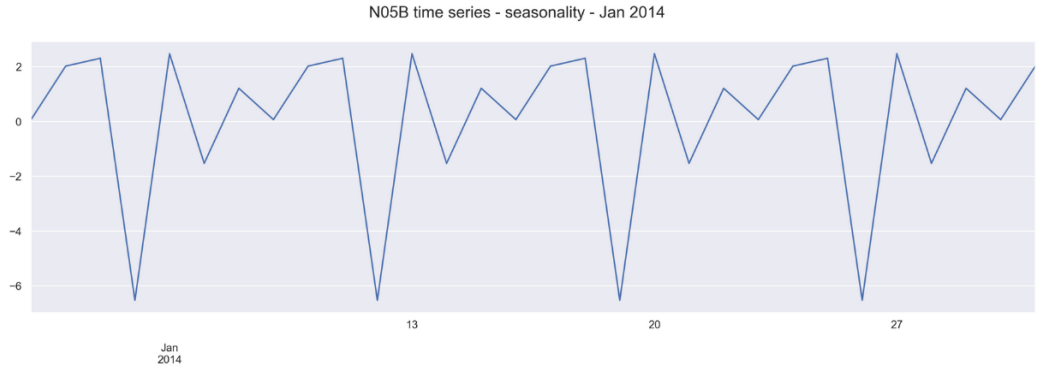


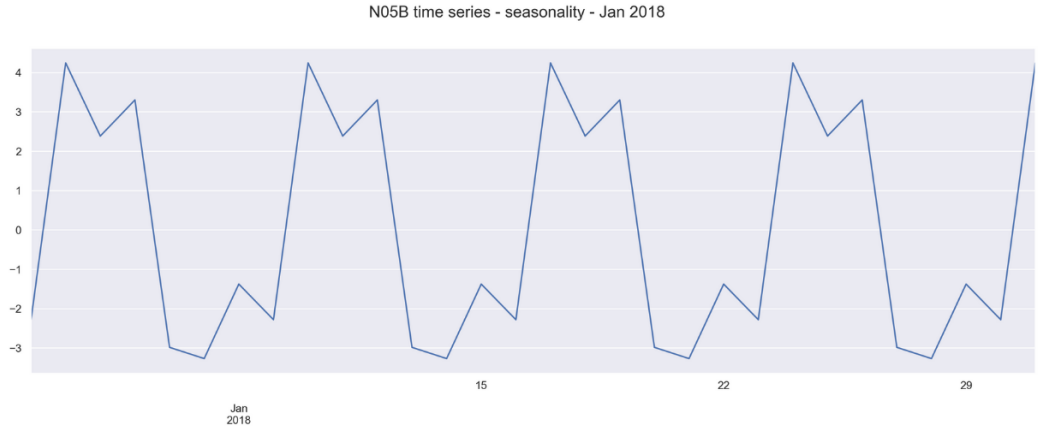

N02BA time series - seasonality - Jan 2018
1.5
1.0
0.5
0.0
−0.5
−1.0
15
22
29
Jan

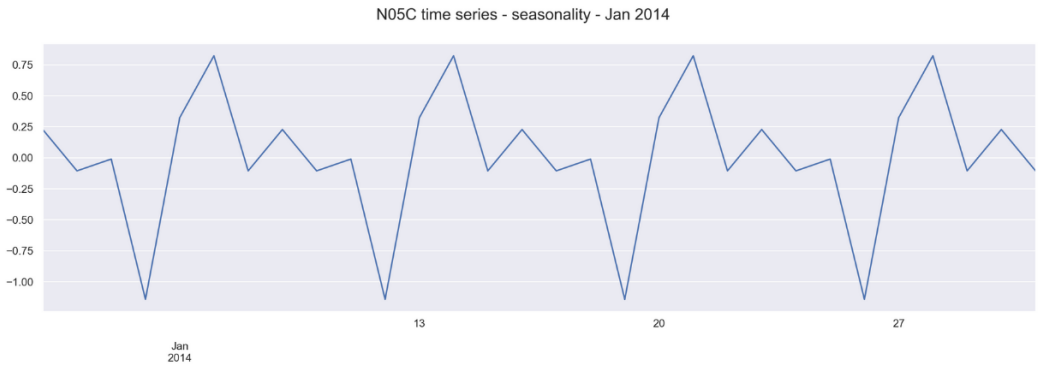

N02BA time series - seasonality - Jan 2014
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
13
20
27
Jan
2014

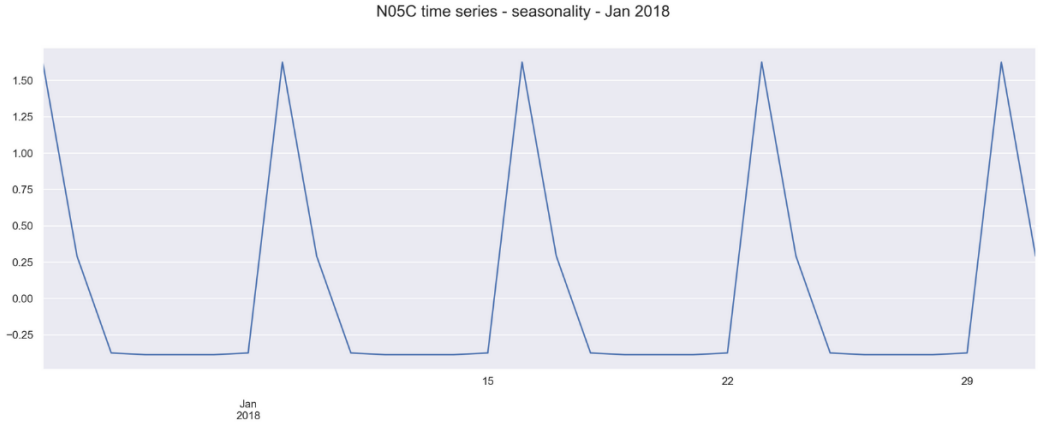

N02BE time series - seasonality - Jan 2014
15
10
5
0
−5
13
20
27
Jan
2014

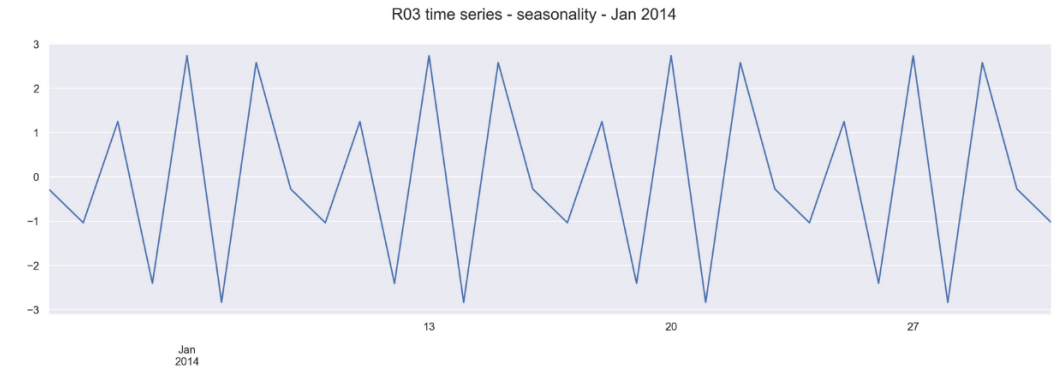

N02BE time series - seasonality - Jan 2018
10.0
7.5
5.0
2.5
0.0
−2.5
−5.0
15
22
29
Jan
2018

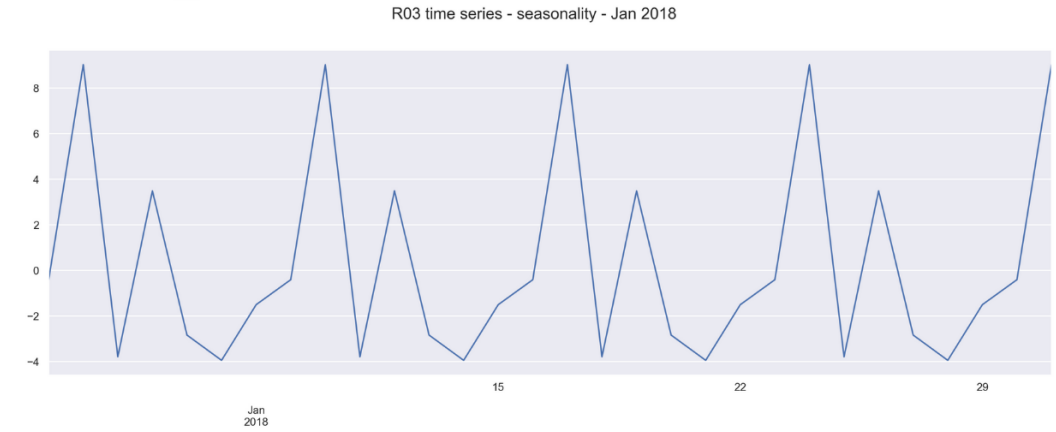

N05B time series - seasonality - Jan 2014
2
0
−2
−4
−6
13
20
27
Jan
2014

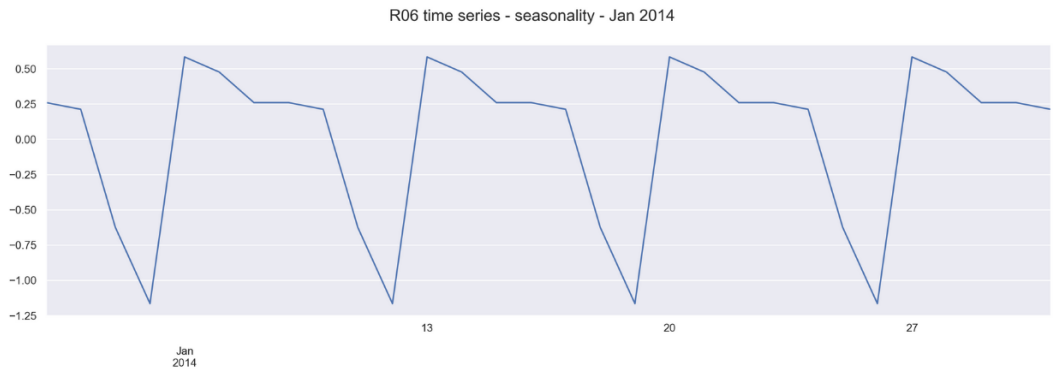

N05B time series - seasonality - Jan 2018
4
3
2
1
0
−1
−2
−3
15
22
29
Jan
2018


N05C time series - seasonality - Jan 2014
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
13
20
27
Jan
2014


N05C time series - seasonality - Jan 2018
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
15
22
29
Jan
2018


R03 time series - seasonality - Jan 2014
3
2
1
0
−1
−2
−3
13
20
27
Jan
2014


R03 time series - seasonality - Jan 2018
8
6
4
2
0
−2
−4
15
22
29
Jan
2018


R06 time series - seasonality - Jan 2014
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
13
20
27
Jan
2014

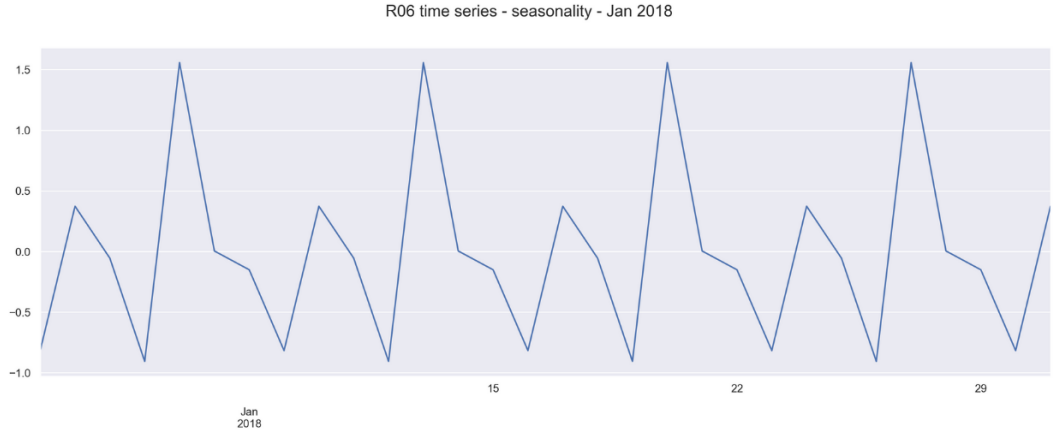
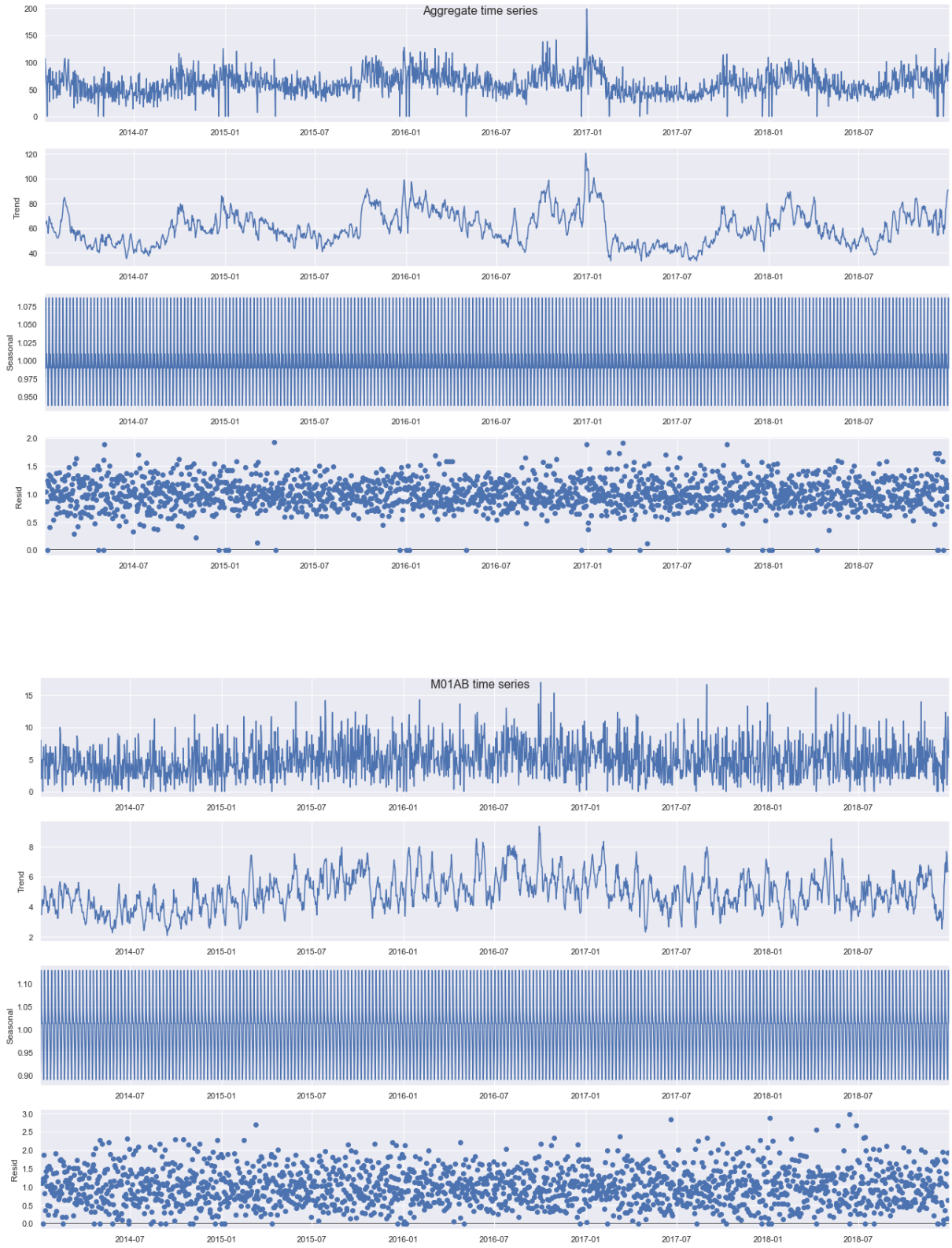
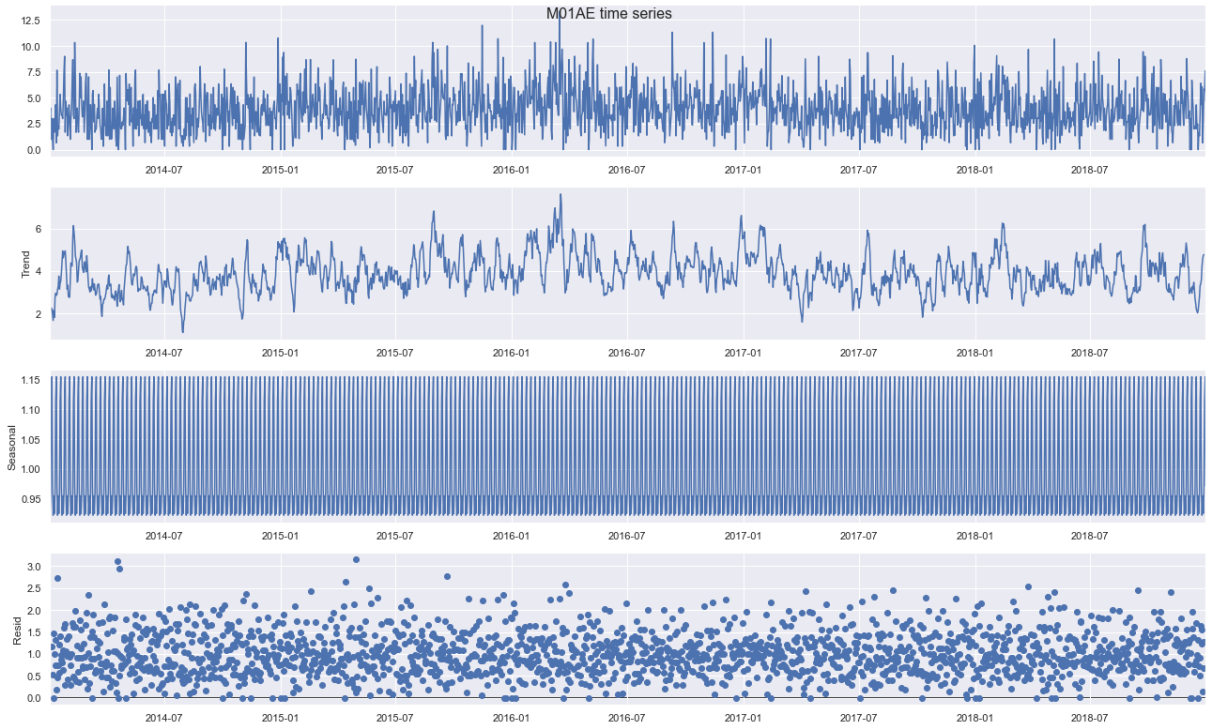
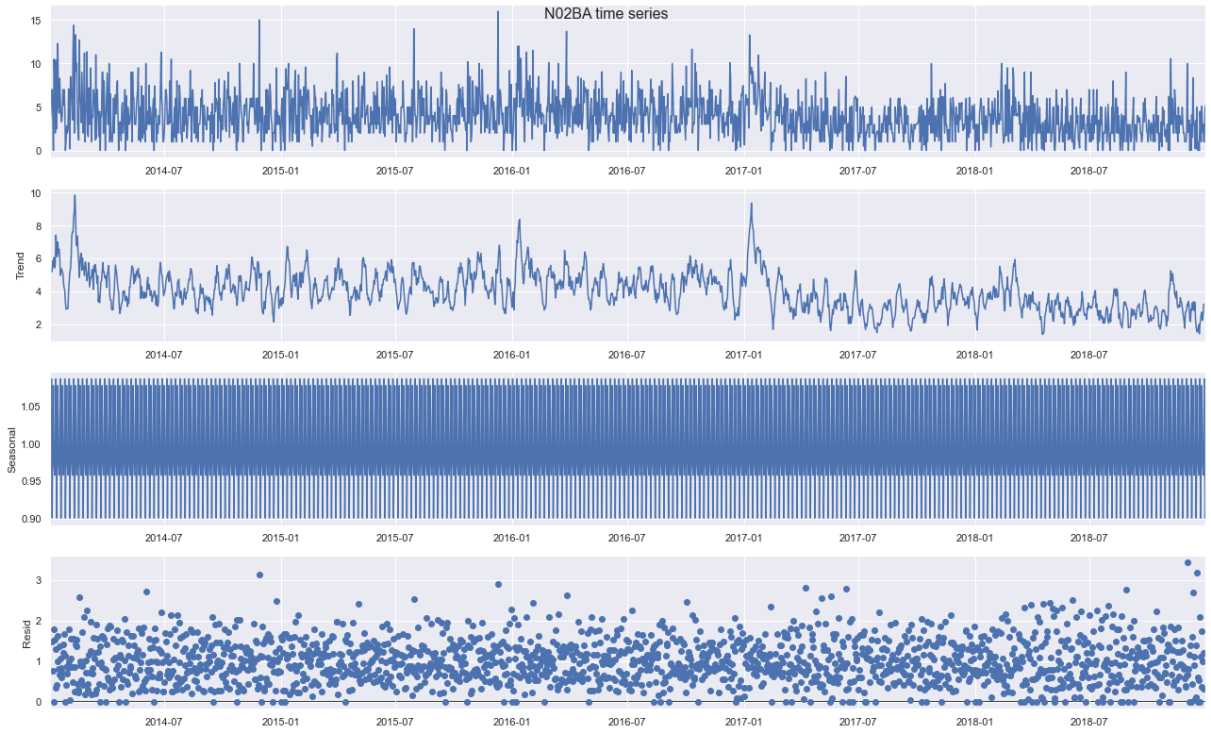
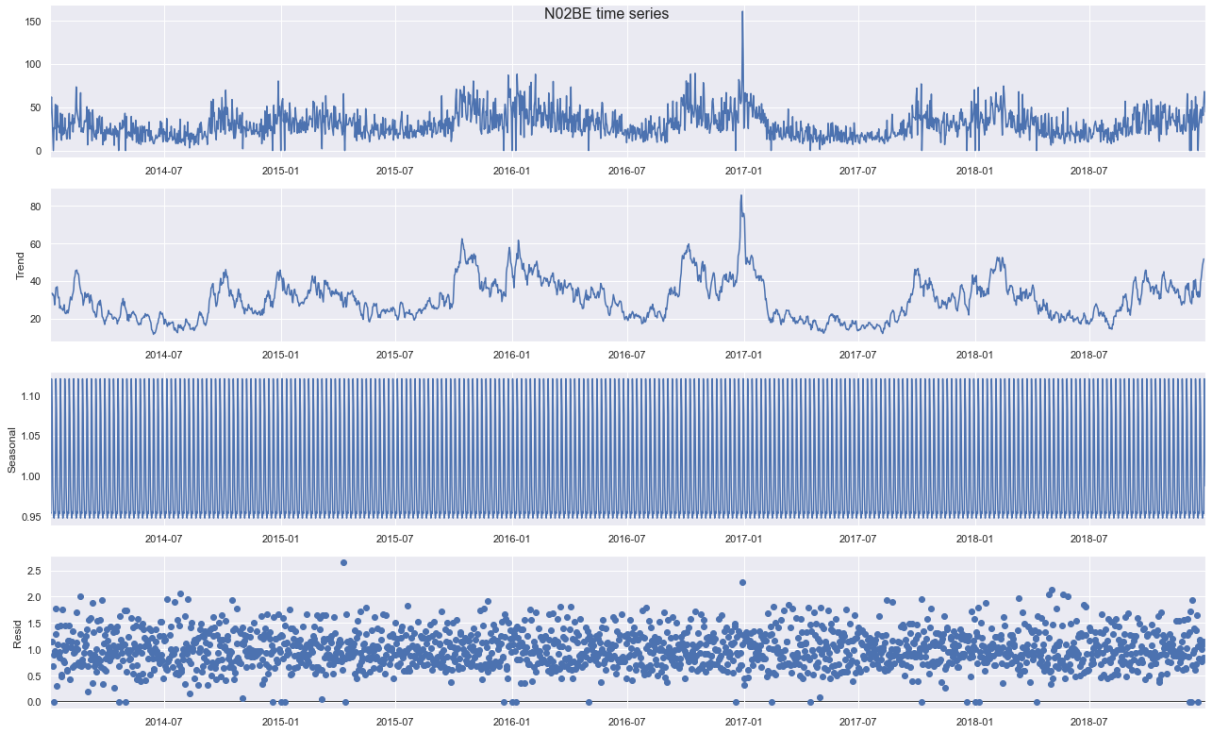

R06 time series - seasonality - Jan 2018
1.5
1.0
0.5
0.0
−0.5
−1.0
15
22
29
Jan
2018

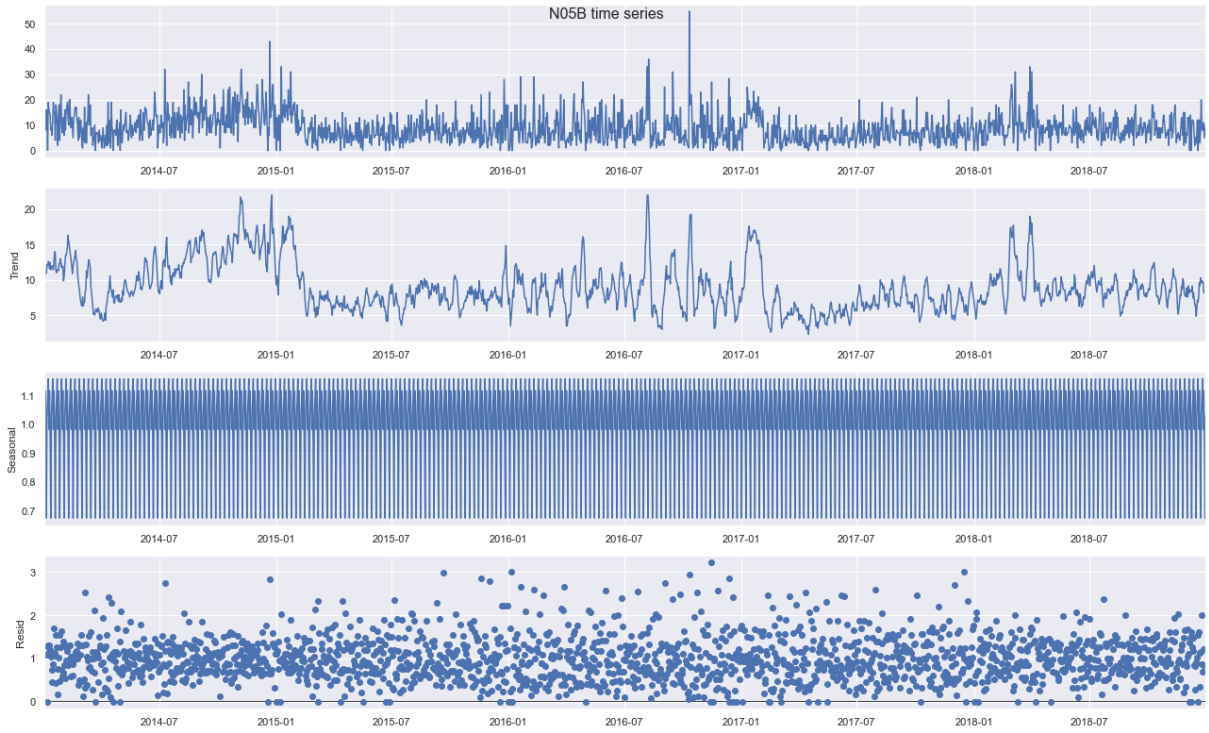
Figure C.2: Decomposition of the aggregate time series and all its sub series with a multiplicative model - Pharma dataset

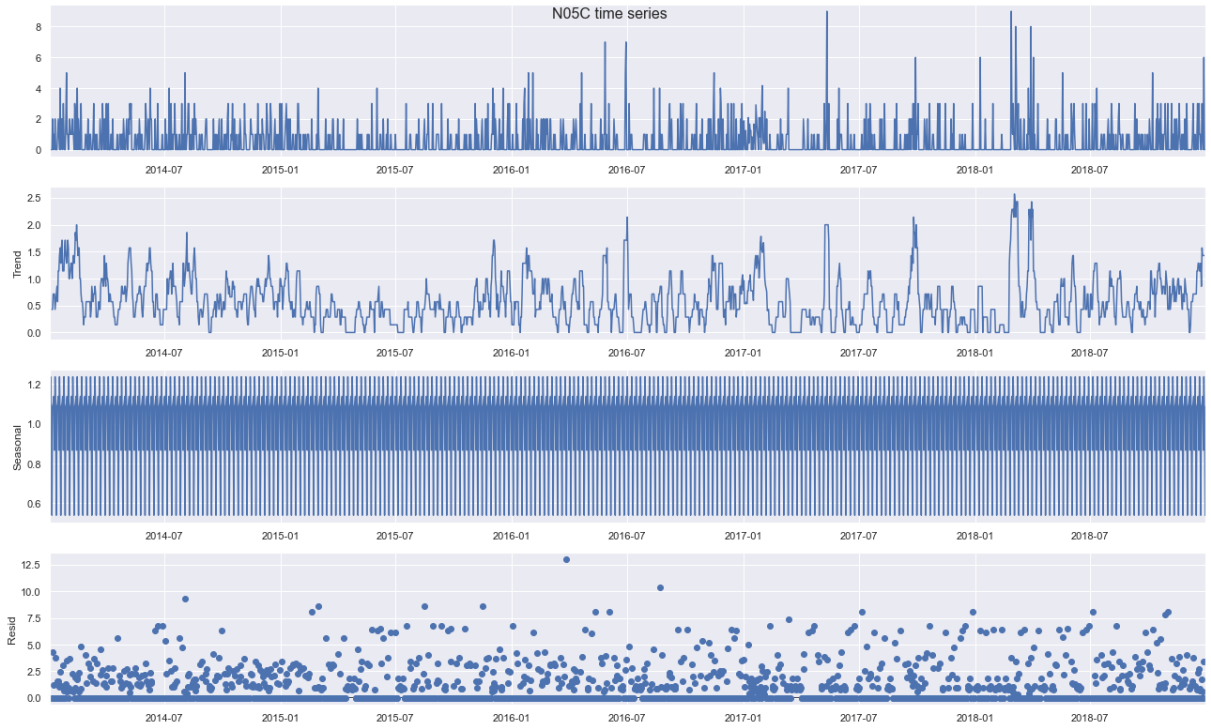


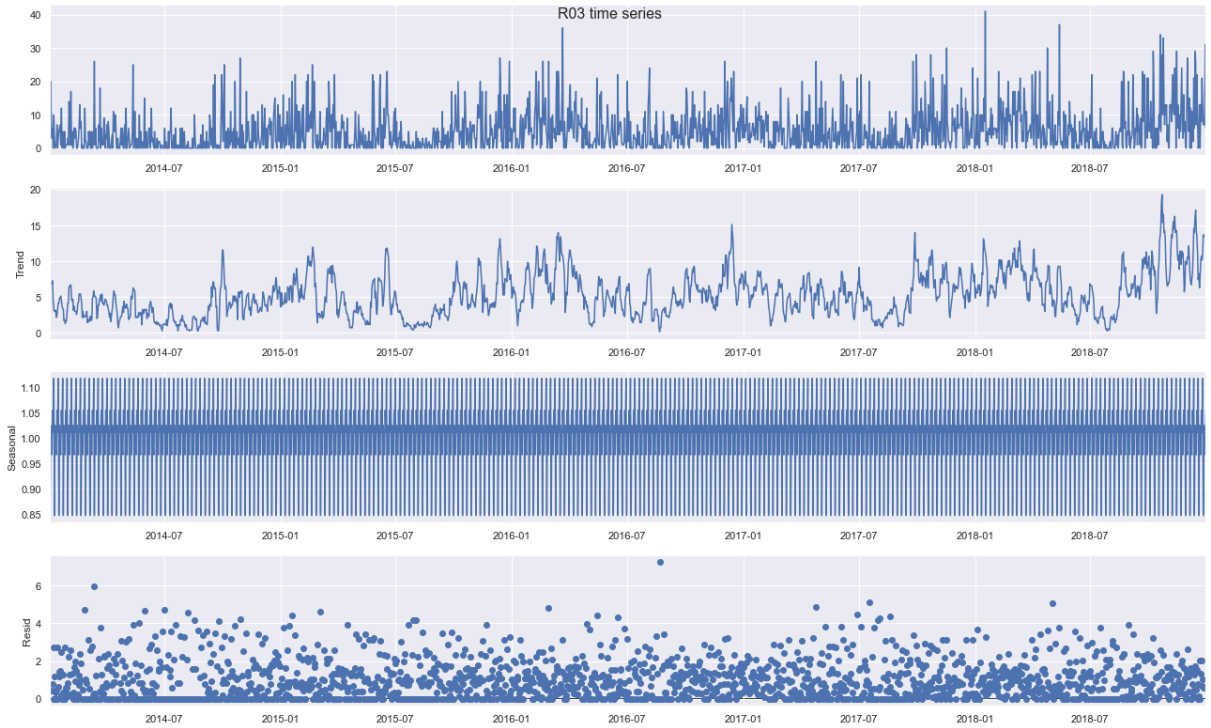

M01AE time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07


N02BA time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07


N02BE time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07


N05B time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07


N05C time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07


R03 time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07

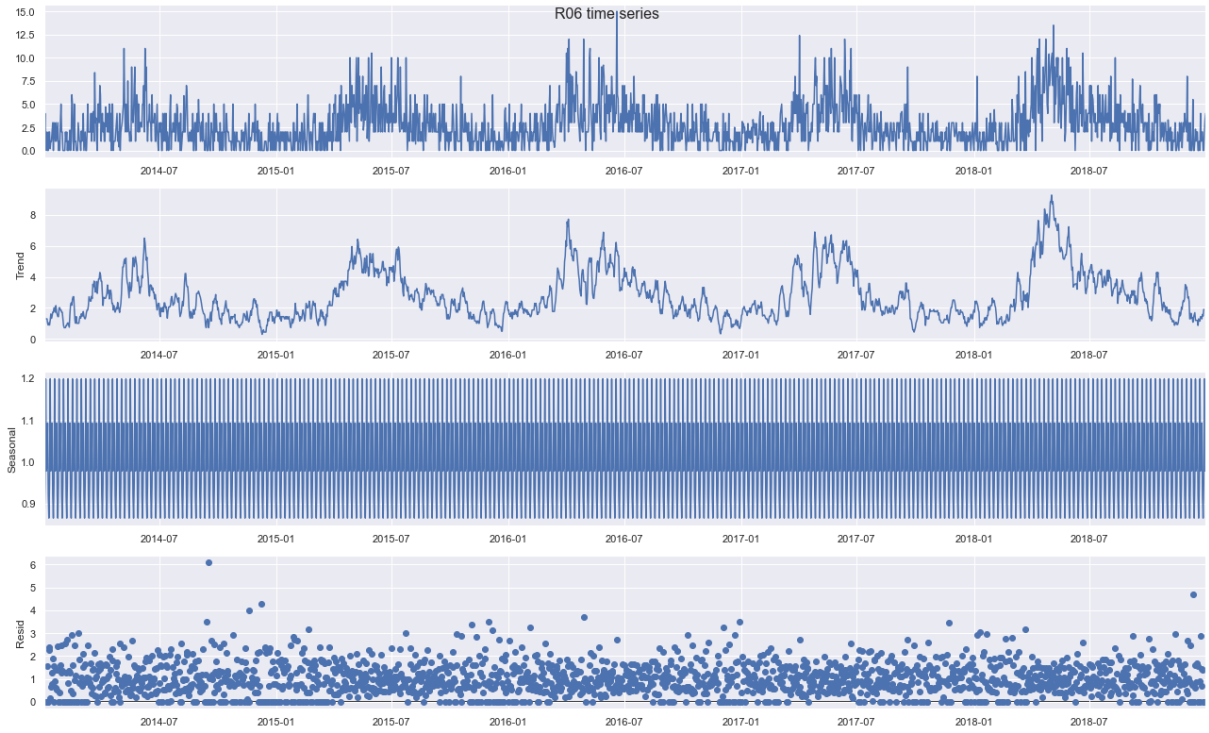
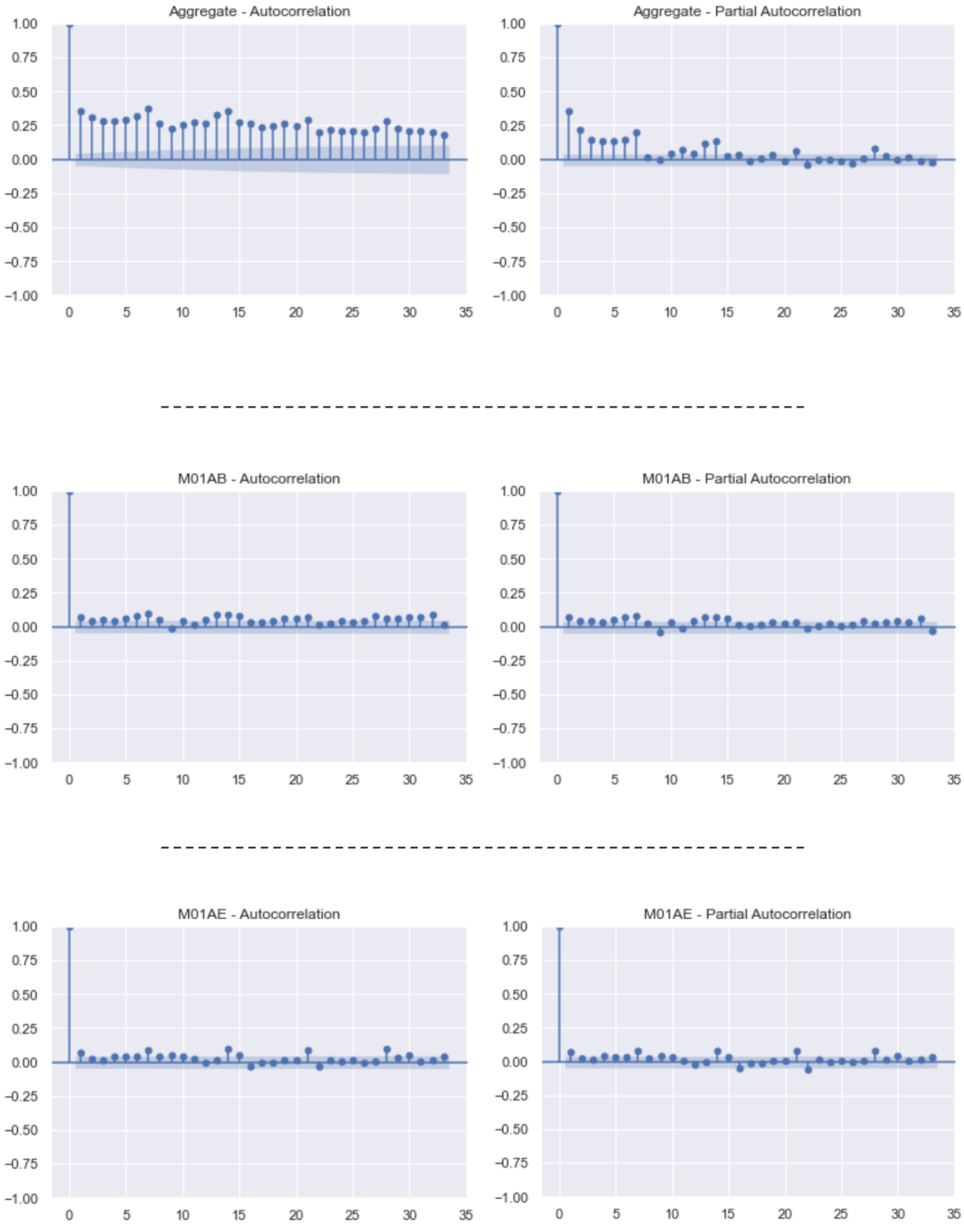
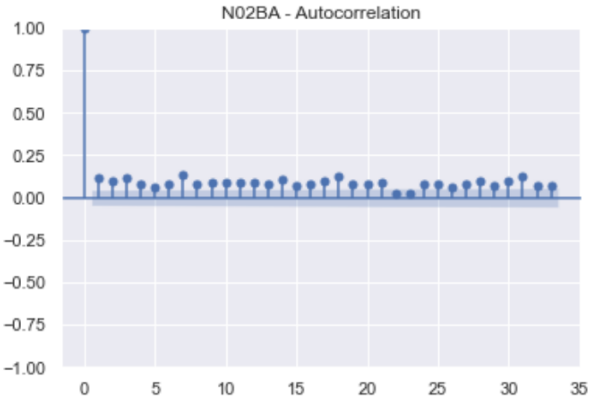
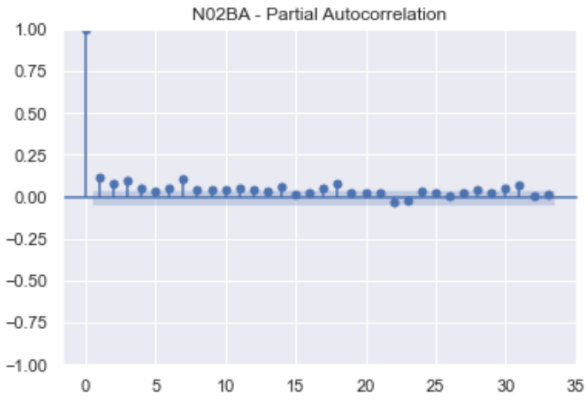
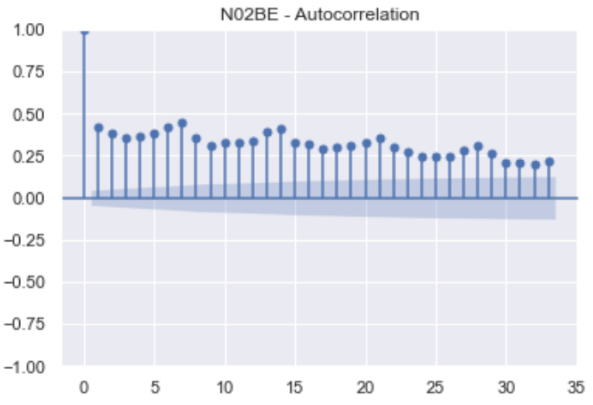

R06 time series
Trend
Seasonal
Resid
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01
2018-07

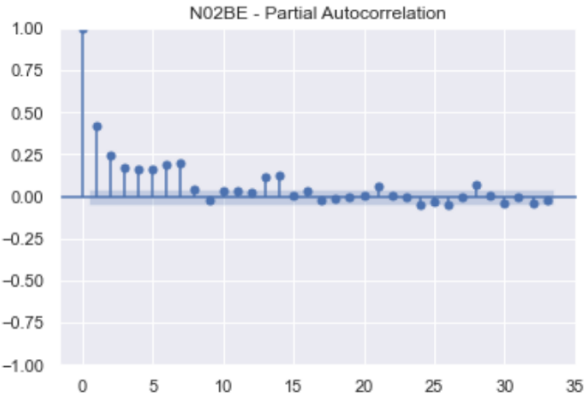
Figure C.3: Autocorrelation and Partial Autocorrelation - Pharma dataset

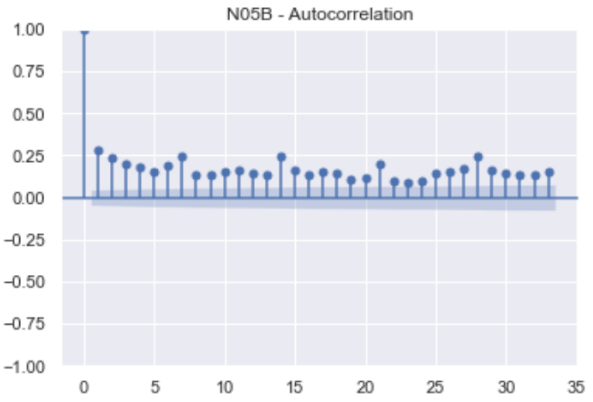


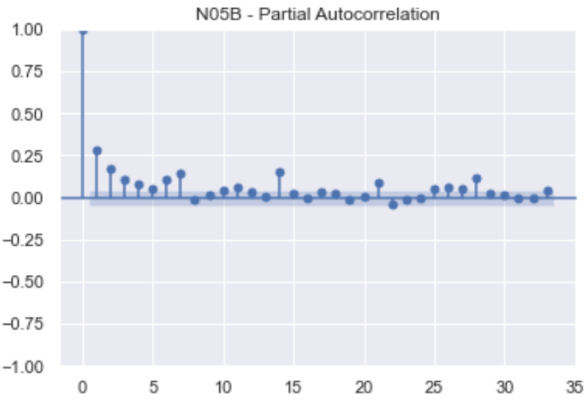

N02BA - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

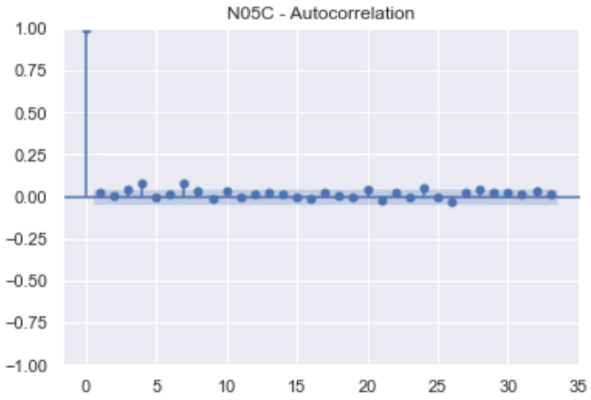

N02BA - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

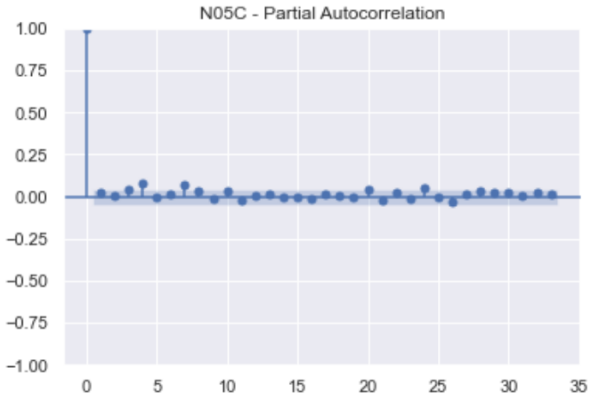

N02BE - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

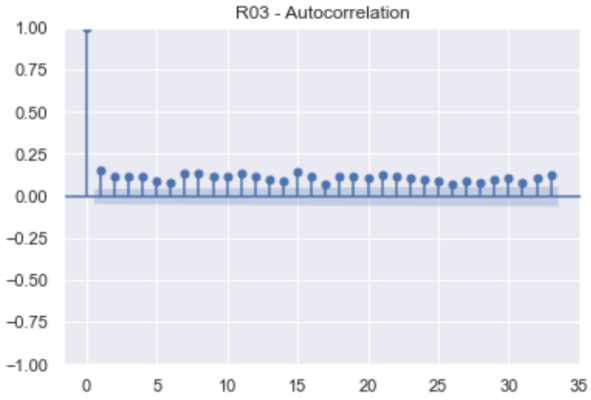

N02BE - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

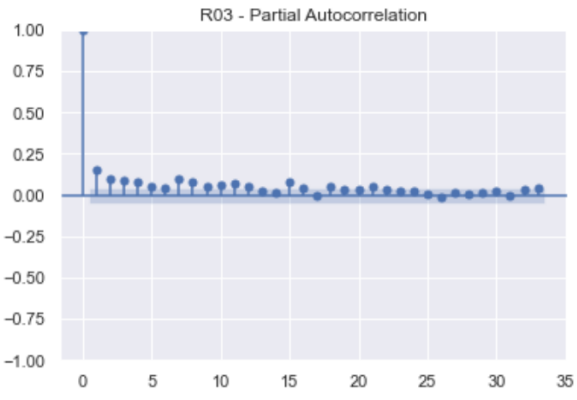

N05B - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

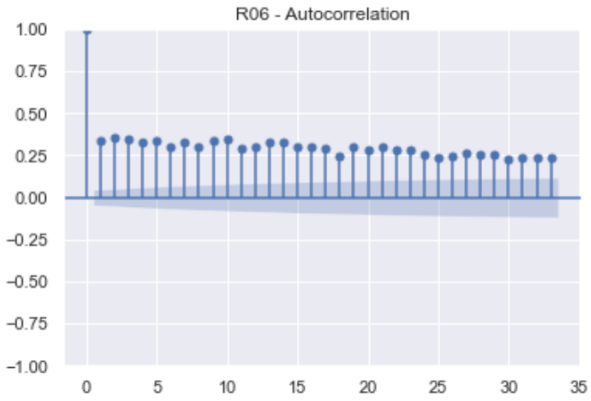

N05B - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35


N05C - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35


N05C - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35


R03 - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35


R03 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35


R06 - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

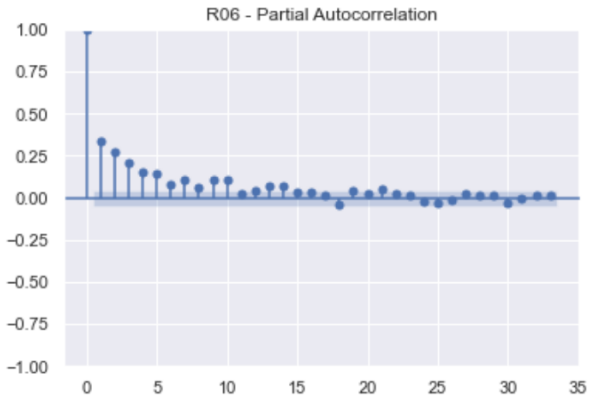
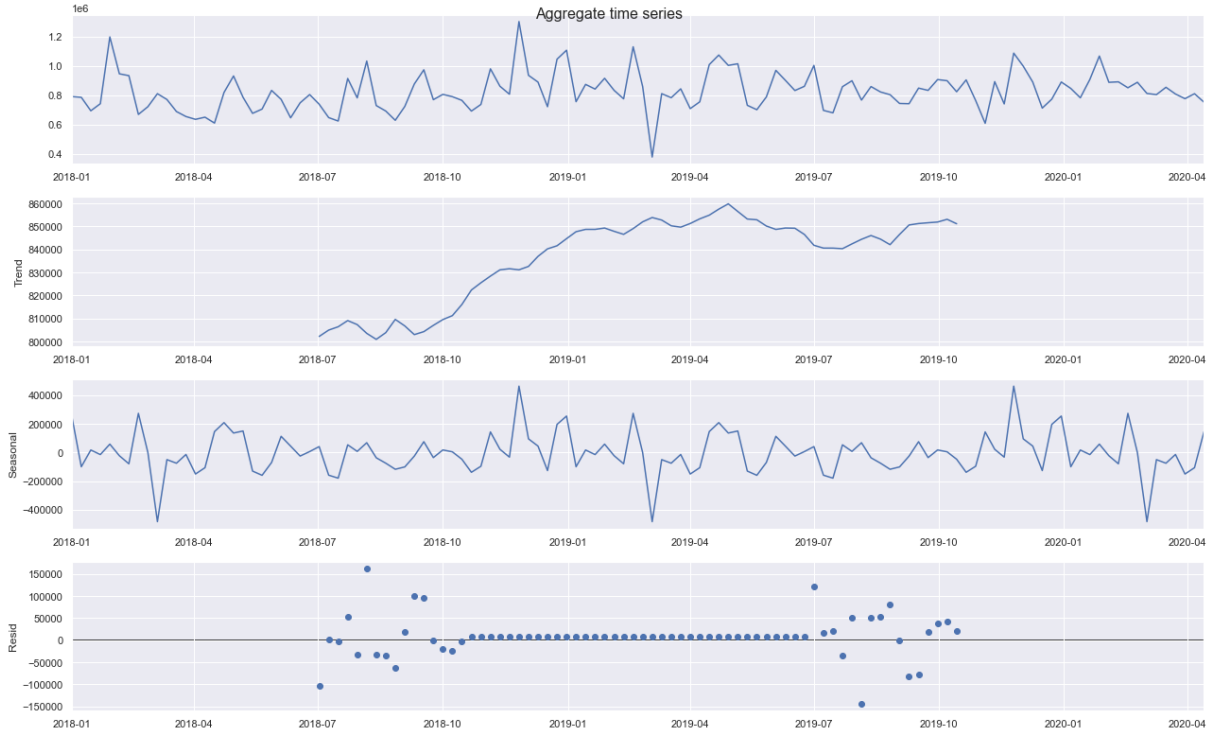
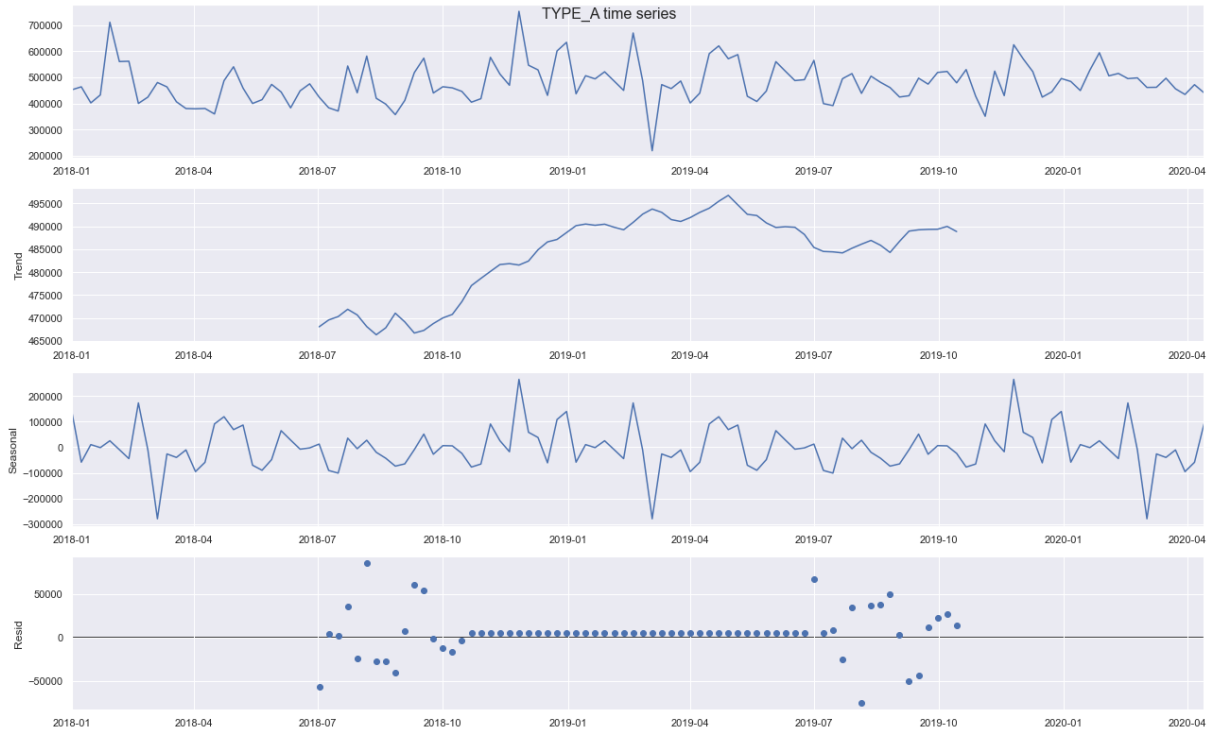
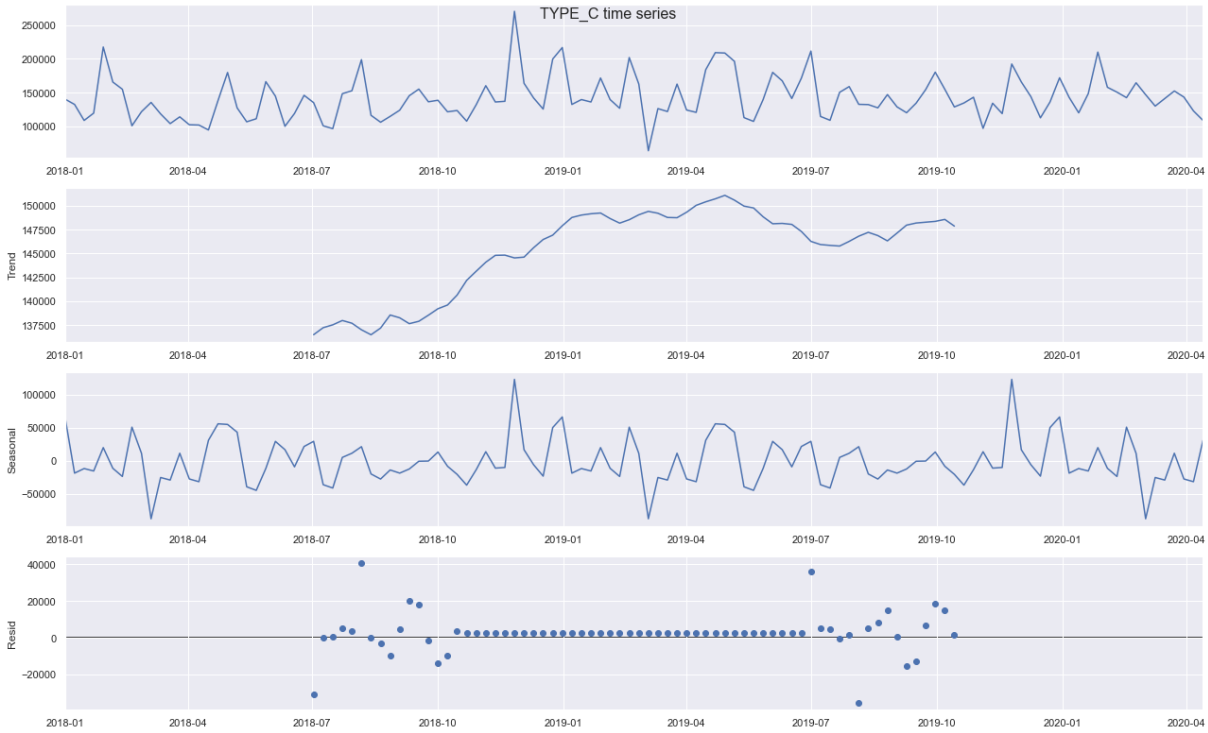
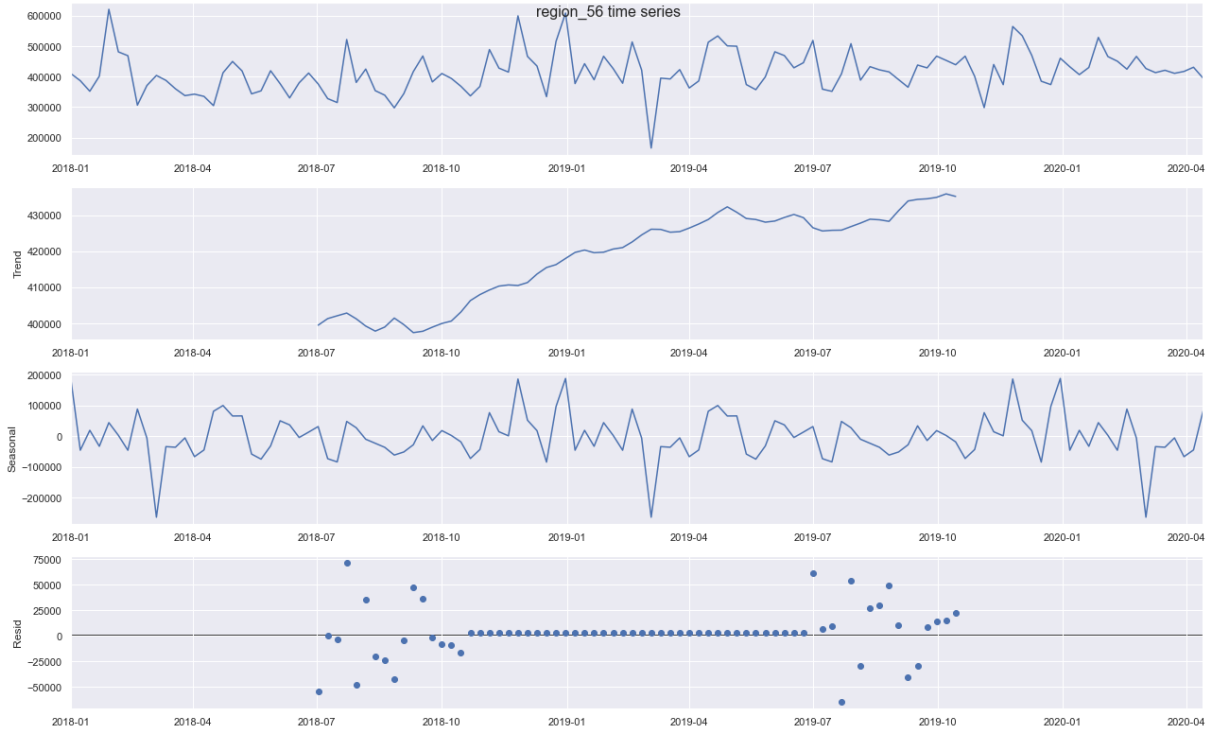

R06 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25
30
35

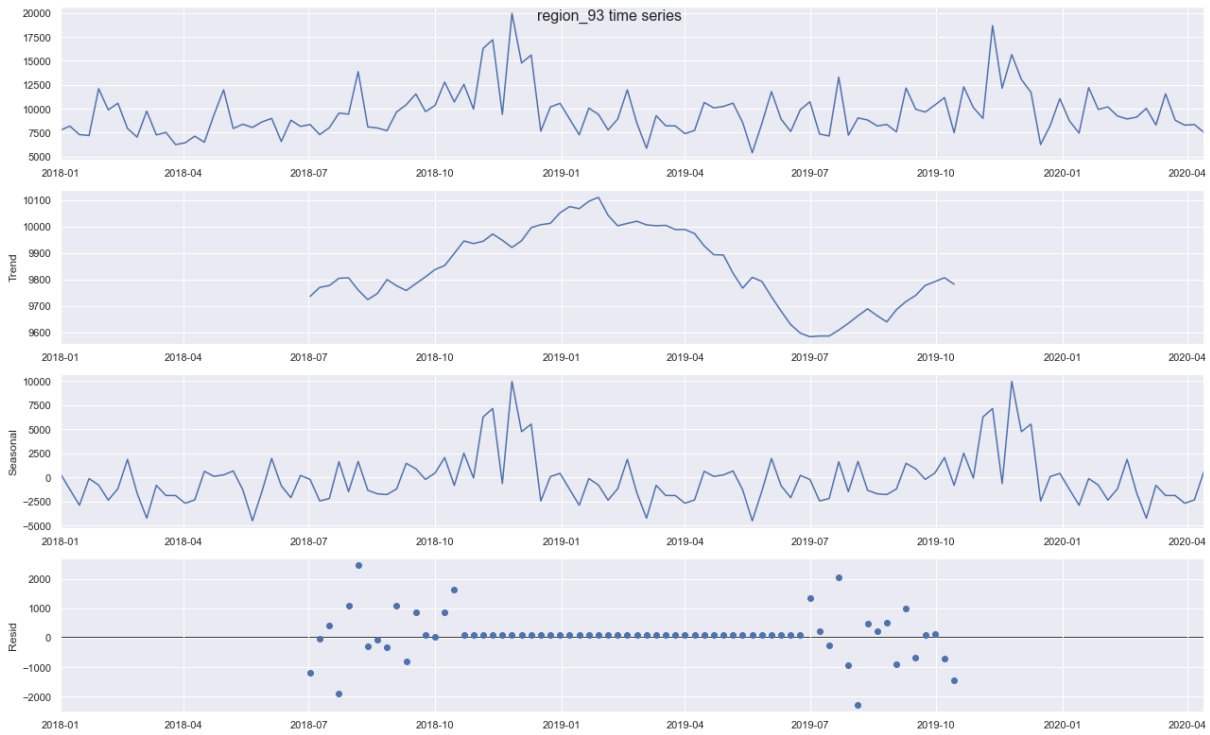
Figure C.4: Decomposition of the aggregate time series and all its sub series with an additive model - Food Demand dataset

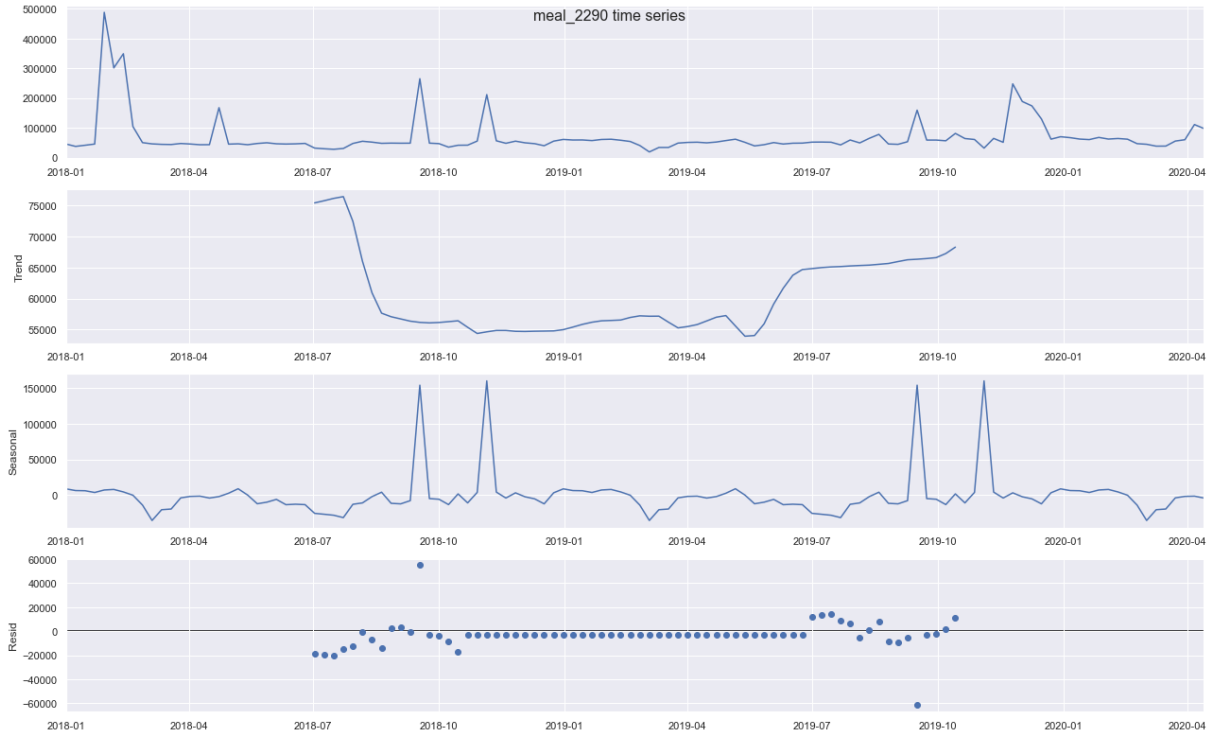


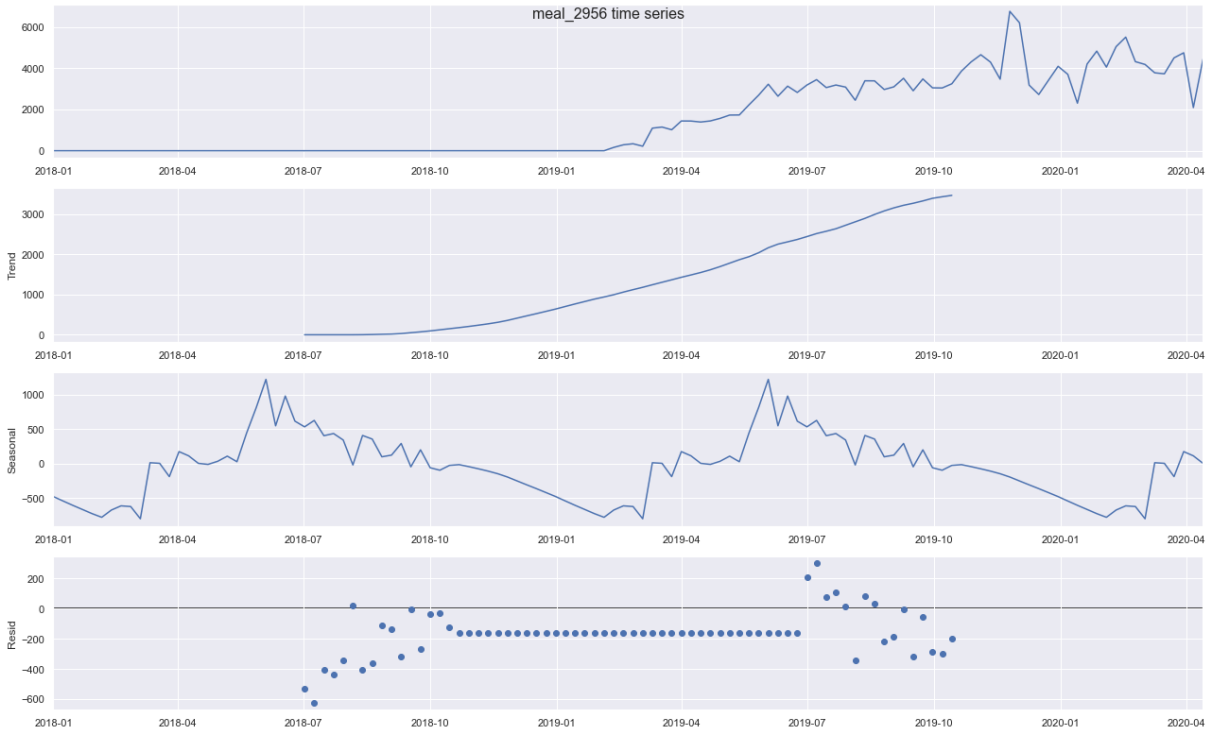


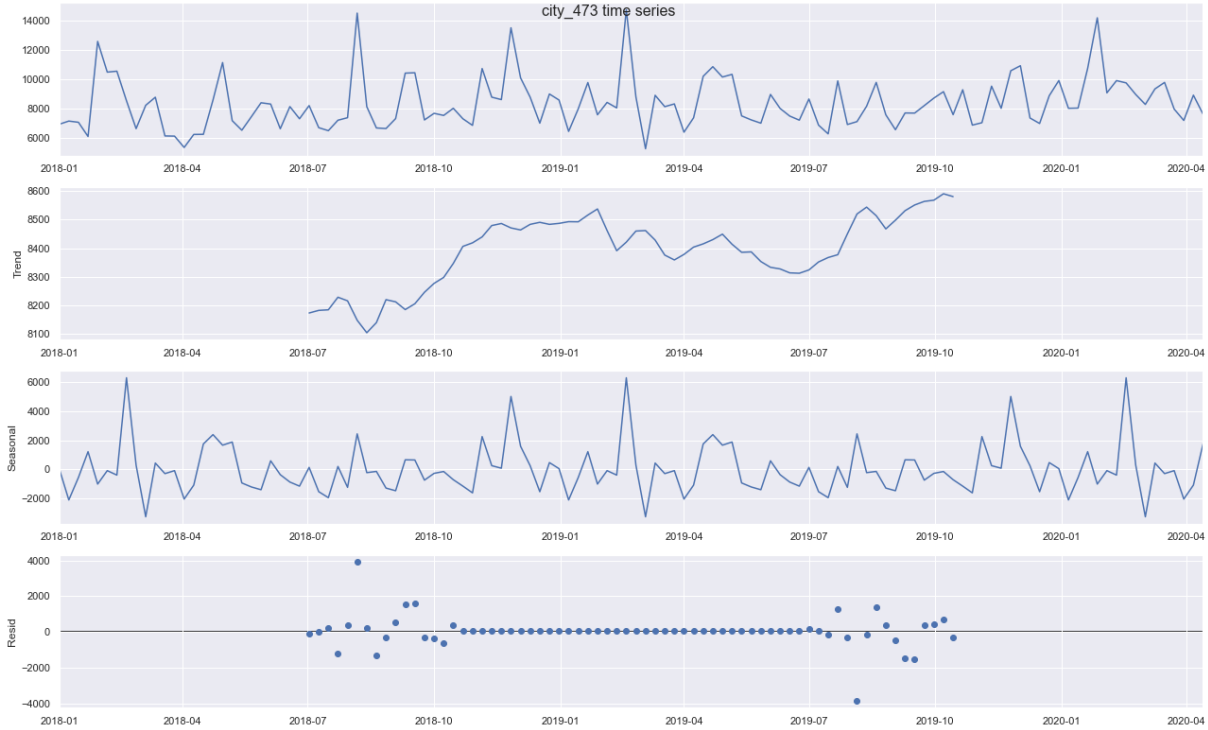

TYPE_C time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04

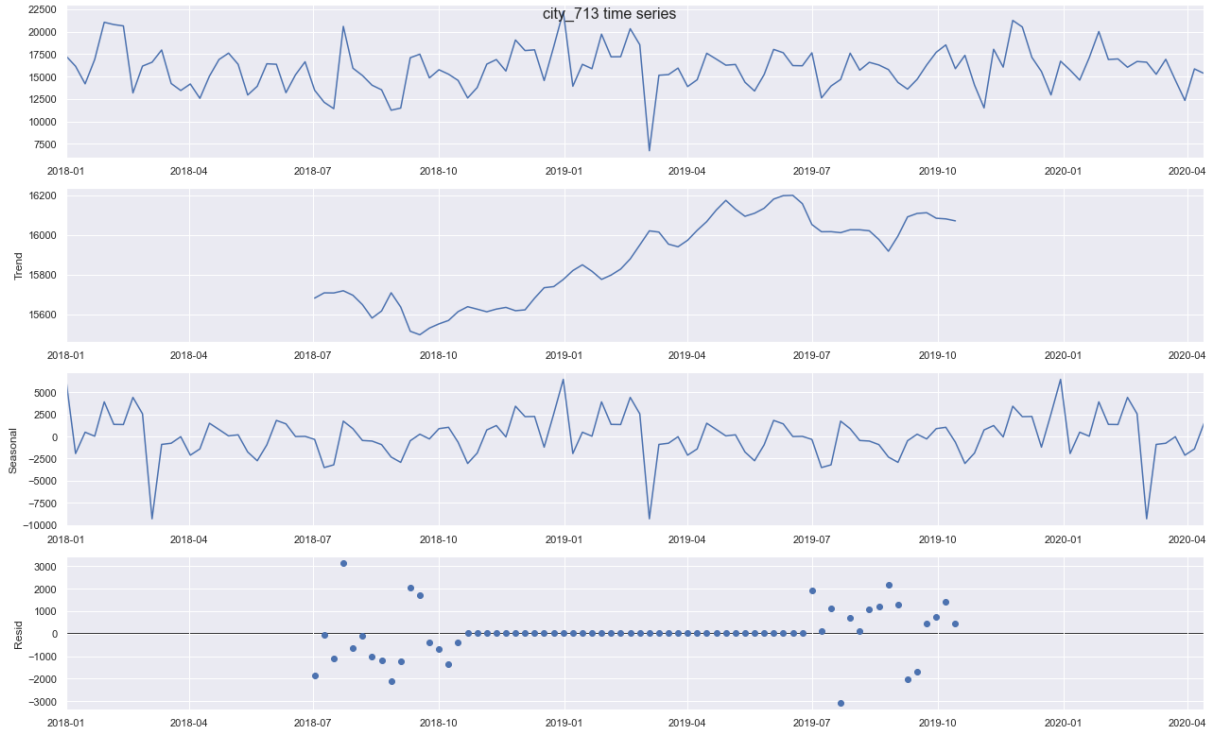

region_56 time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04

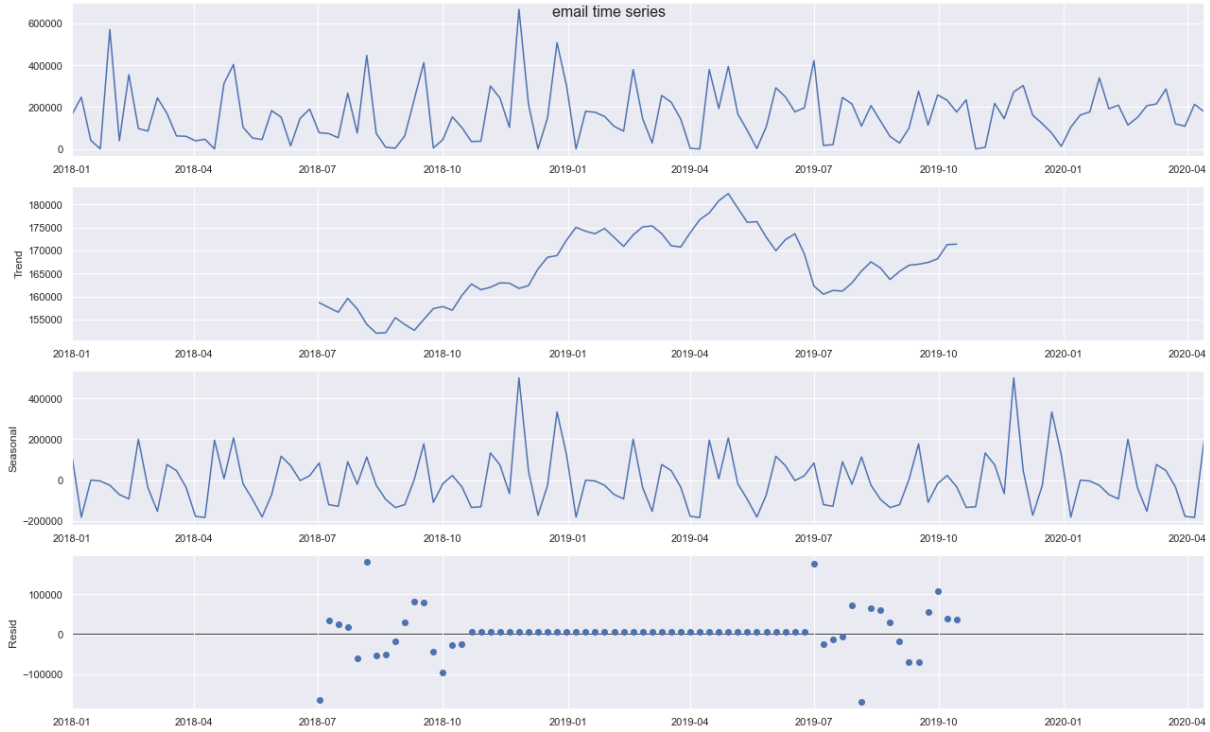

region_93 time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04


meal_2290 time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04


meal_2956 time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04


city_473 time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04


city_713 time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04


email time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04

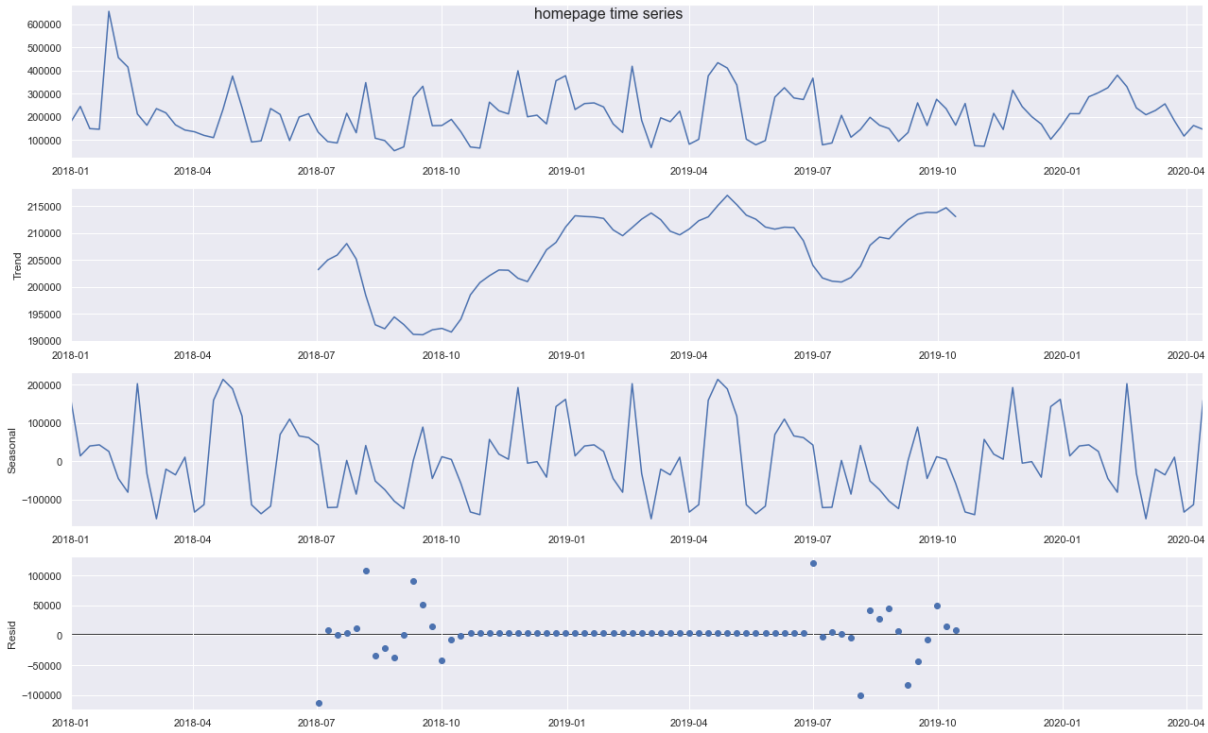
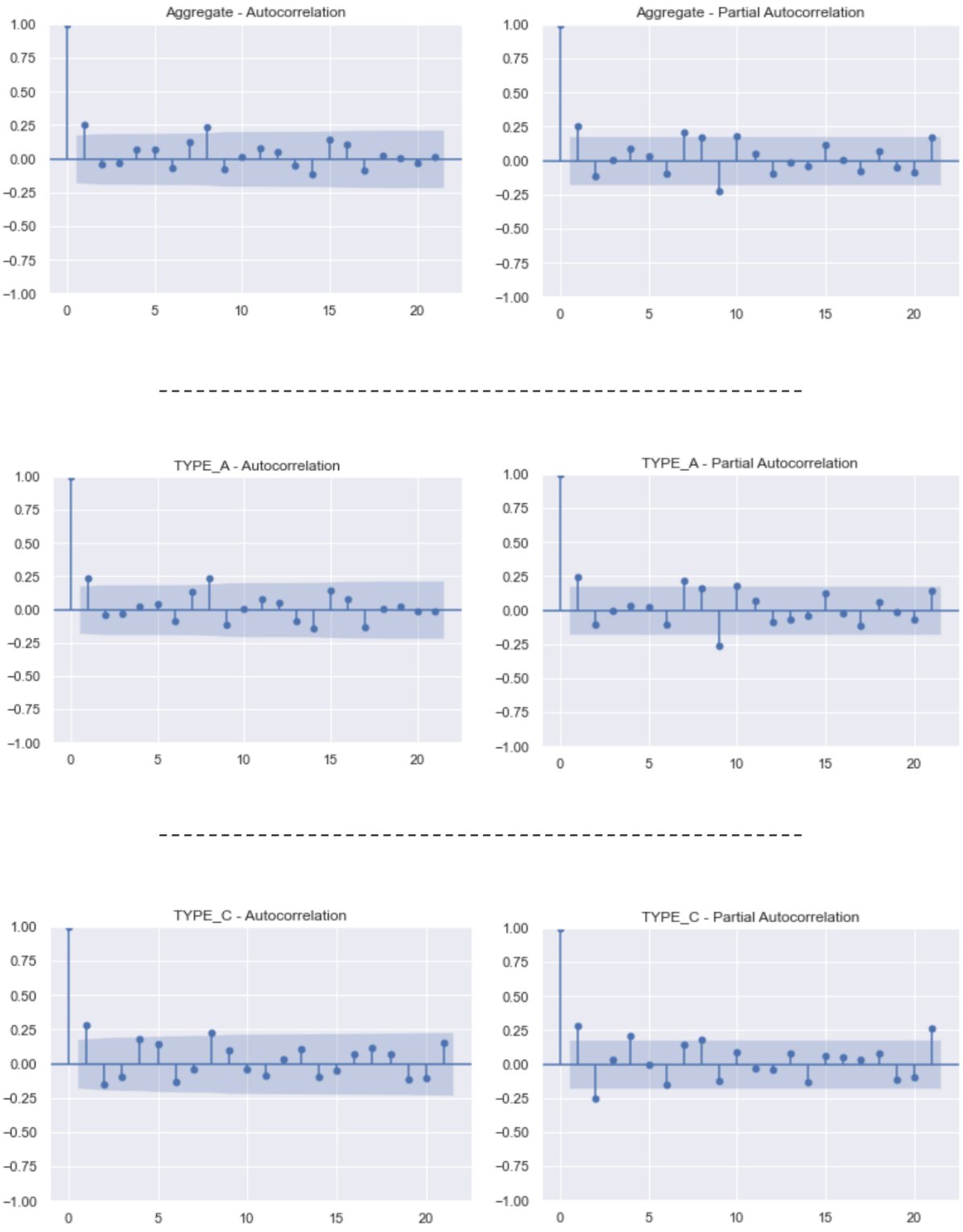
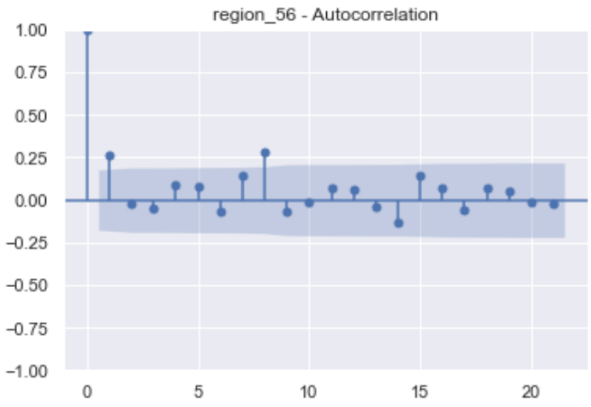
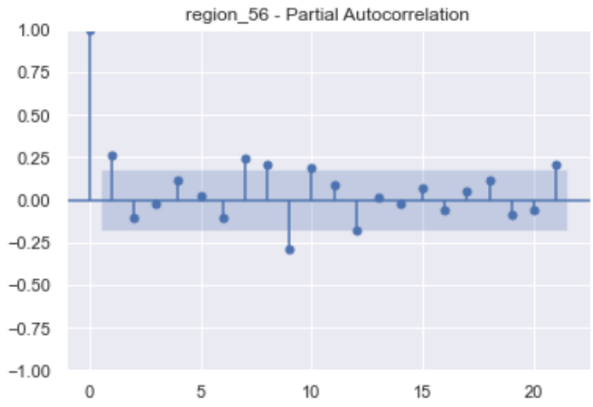
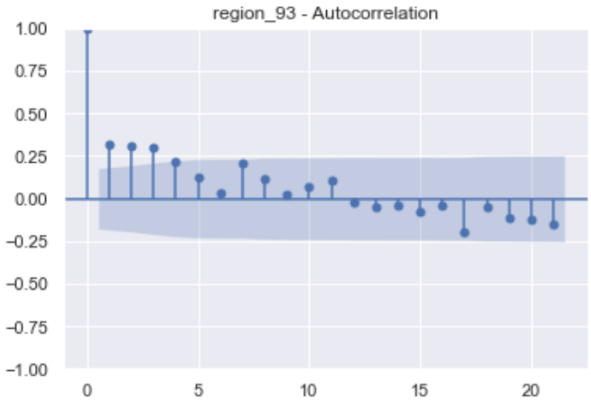

homepage time series
Trend
Seasonal
Resid
2018-01
2018-04
2018-07
2018-10
2019-01
2019-04
2019-07
2019-10
2020-01
2020-04

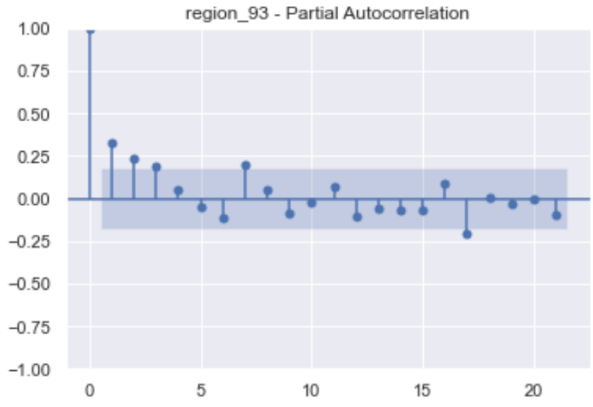
Figure C.5: Autocorrelation and Partial Autocorrelation - Food Demand dataset

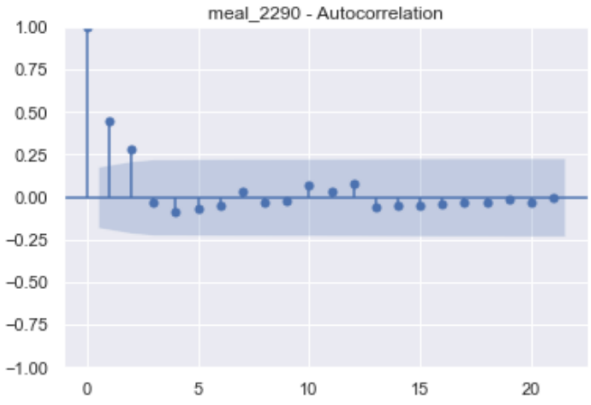


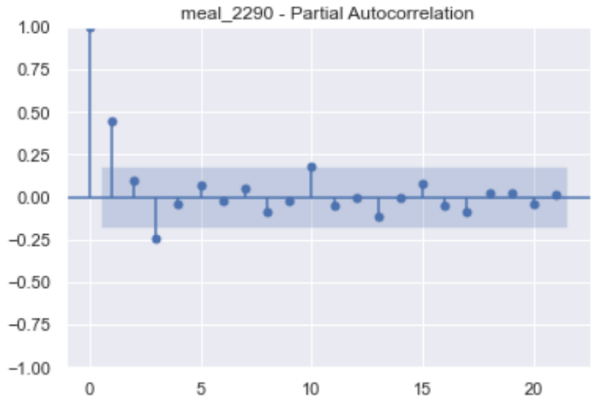

region_56 - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

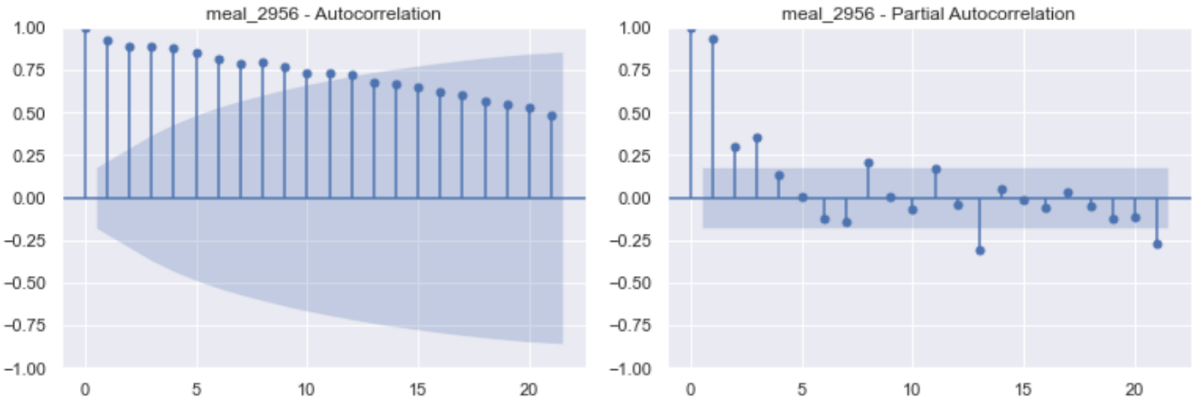

region_56 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

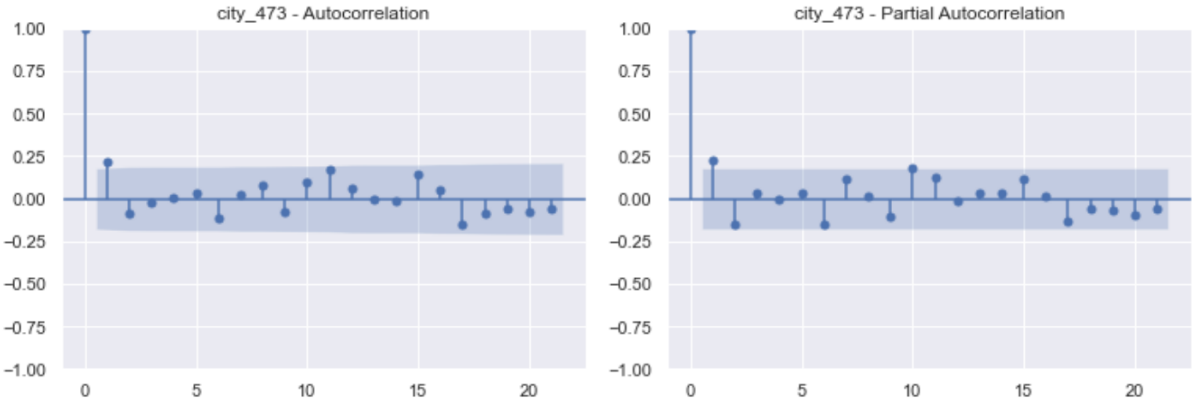

region_93 - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

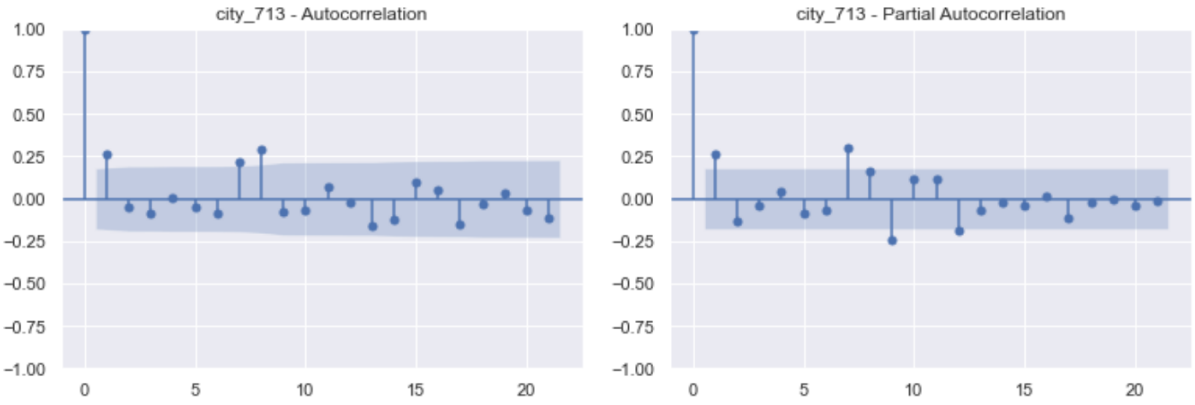

region_93 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

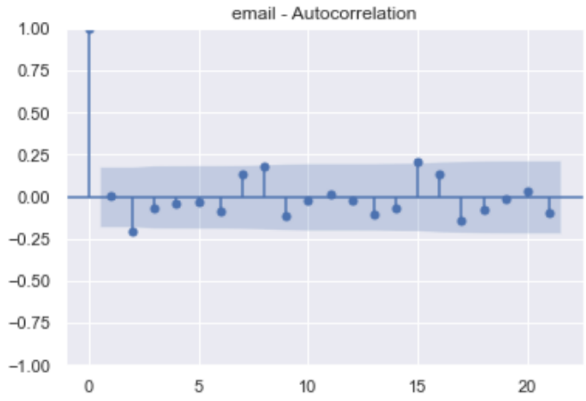

meal_2290 - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

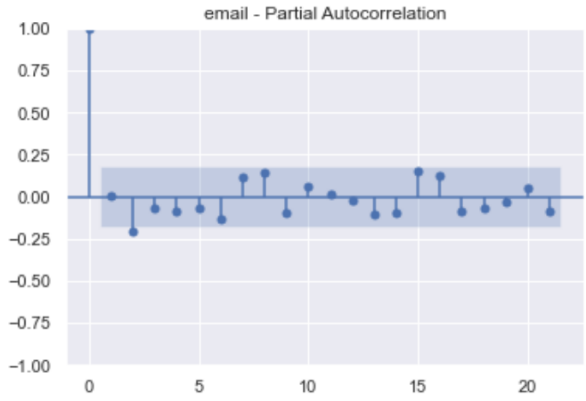

meal_2290 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

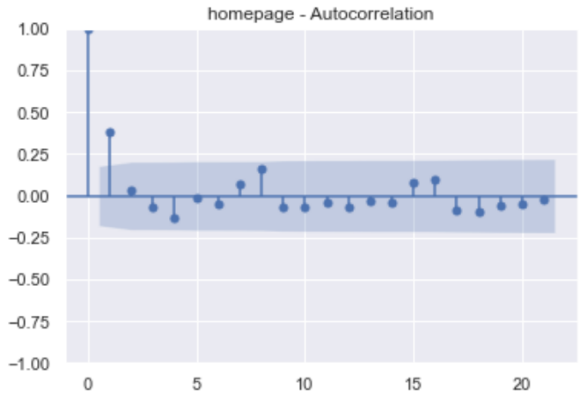

meal_2956 - Autocorrelation
meal_2956 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

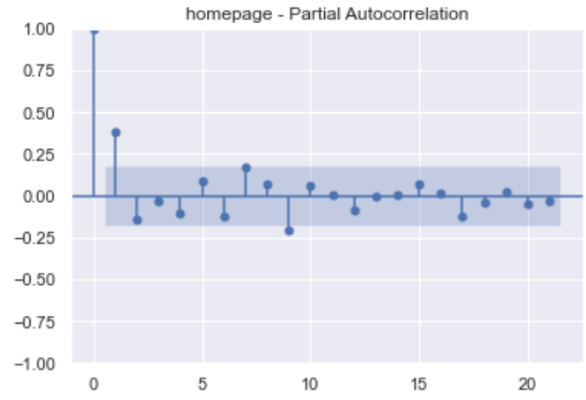

city_473 - Autocorrelation
city_473 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


city_713 - Autocorrelation
city_713 - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


email - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


email - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


homepage - Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20


homepage - Partial Autocorrelation
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20

D | Experiments' Complete Results

Figure D.1: Results of the experiments of all the sub series - Pharma dataset

dataset	model	mae	mape	mape_adj	msd	mse	rmse	parameters
M01AB	Exponential Smoothing	2.1882	inf	68.18	0.34	8.27	2.88	[{'optimized': True, 'remove_bias': True}, {'trend': 'add', 'damped_trend': False, 'seasonal': 'mul', 'seasonal_periods': 7}, {'smoothing_level': '0.011512699873772047', 'smoothing_trend': '4.202738091218606e-12', 'smoothing_seasonal': '0.01629554831626092', 'damping_trend': 'nan'}]
M01AB	auto_arima	2.2134	inf	68.33	0.33	8.29	2.88	[[1, 1, 2], [0, 0, 0, 0]]
M01AB	prophet	2.1889	inf	69.33	0.27	8.20	2.86	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 1.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True, 'daily_seasonality': True}
M01AB	LinearRegression	2.3079	inf	72.59	0.20	8.81	2.97	{params":{"0":{"fit_intercept":true}}}"
M01AB	SVR	2.2096	inf	68.55	0.31	8.25	2.87	{params":{"231":{"C":5,cache_size:200,coef0:0.0,degree:2,epsilon:0.01,gamma:1,kernel:"rbf",max_iter:-1,shrinking:false,tol:0.001,verbose:false}}}"
M01AB	Ridge	2.2099	inf	69.56	0.23	8.21	2.87	{params":{"14":{"alpha":10000,normalize:true}}}"
M01AB	Lasso	2.2099	inf	69.56	0.23	8.21	2.87	{params":{"4":{"alpha":0.1,normalize:true}}}"
M01AB	KNeighborsRegressor	2.3867	inf	59.36	1.19	9.77	3.13	{params":{"12":{"n_neighbors":40,p:1}}}"
M01AB	GradientBoostingRegressor	2.2072	inf	69.15	0.24	8.10	2.85	{params":{"4":{"learning_rate":0.01,max_depth:6}}}"

M01AE	Exponential Smoothing	1.7375	inf	108.03	0.07	5.64	2.37	[{'optimized': True, 'remove_bias': False}, {'trend': 'mul', 'damped_trend': False, 'seasonal': 'mul', 'seasonal_periods': 7}, {'smoothing_level': '0.007126372317948543', 'smoothing_trend': '1.0220816835322314e-11', 'smoothing_seasonal': '0.03140329221973143', 'damping_trend': 'nan'}]
M01AE	auto_arima	1.7778	inf	107.73	0.08	5.64	2.37	[[1, 1, 3], [0, 0, 0, 0]]
M01AE	prophet	1.6806	inf	114.57	-0.19	5.27	2.30	{'changepoint_prior_scale': 0.1, 'seasonality_prior_scale': 0.1, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': True, 'daily_seasonality': False}
M01AE	LinearRegression	1.8426	inf	104.34	0.02	5.86	2.42	{params":{"1":{"fit_intercept":false}}}"
M01AE	SVR	1.7947	inf	113.49	-0.11	5.64	2.38	{params":{"310":{"C":5,cache_size:200,coef0:0.0,degree:0,epsilon:0.01,gamma:1,kernel:"rbf",max_iter:-1,shrinking:true,tol:0.001,verbose:false}}}"
M01AE	Ridge	1.7900	inf	113.62	-0.12	5.57	2.36	{params":{"8":{"alpha":10,normalize:true}}}"
M01AE	Lasso	1.7972	inf	114.25	-0.13	5.65	2.38	{params":{"2":{"alpha":0.01,normalize:true}}}"
M01AE	KNeighborsRegressor	1.7699	inf	105.30	0.08	5.48	2.34	{params":{"19":{"n_neighbors":55,p:2}}}"
M01AE	GradientBoostingRegressor	1.7864	inf	111.52	-0.07	5.53	2.35	{params":{"4":{"learning_rate":0.01,max_depth:6}}}"
N02BA	Exponential Smoothing	1.4639	inf	88.29	0.31	3.76	1.94	[{'optimized': True, 'remove_bias': False}, {'trend': 'mul', 'damped_trend': False, 'seasonal': 'add', 'seasonal_periods': 7}, {'smoothing_level': '0.021397422709691458', 'smoothing_trend': '5.7318297519468235e-06', 'smoothing_seasonal': '0.019935029365793153', 'damping_trend': 'nan'}]
N02BA	auto_arima	1.4847	inf	86.66	0.37	3.81	1.95	[[0, 1, 1], [0, 0, 0, 0]]
N02BA	prophet	1.4563	inf	86.67	0.27	3.75	1.94	{'changepoint_prior_scale': 0.001, 'seasonality_prior_scale': 0.1, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True, 'daily_seasonality': False}
N02BA	LinearRegression	1.6064	inf	86.00	0.44	4.35	2.09	{params":{"1":{"fit_intercept":false}}}"
N02BA	SVR	1.6610	inf	124.46	-0.71	4.18	2.04	{params":{"308":{"C":5,cache_size:200,coef0:0.0,degree:0,epsilon:0.01,gamma:0.5,kernel:"rbf",max_iter:1000,shrinking:true,tol:0.001,verbose:false}}}"
N02BA	Ridge	1.5029	inf	91.06	0.21	3.84	1.96	{params":{"6":{"alpha":1,normalize:true}}}"
N02BA	Lasso	1.4979	inf	88.82	0.28	3.85	1.96	{params":{"0":{"alpha":0.001,normalize:true}}}"
N02BA	KNeighborsRegressor	1.6347	inf	118.37	-0.54	4.10	2.02	{params":{"20":{"n_neighbors":60,p:1}}}"
N02BA	GradientBoostingRegressor	1.5638	inf	101.56	-0.10	3.84	1.96	{params":{"47":{"learning_rate":0.06,max_depth:9}}}"
N02BE	Exponential Smoothing	9.7794	inf	111.59	-17.07	558.28	23.63	[{'optimized': True, 'remove_bias': True}, {'trend': 'mul', 'damped_trend': True, 'seasonal': 'add', 'seasonal_periods': 365}, {'smoothing_level': '0.09937813729017655', 'smoothing_trend': '0.00024595391551714537', 'smoothing_seasonal': '0.017866403588379997', 'damping_trend': '0.9878328643585678'}]
N02BE	auto_arima	21.1210	inf	111.73	-17.11	559.52	23.65	[[0, 1, 1], [0, 0, 0, 0]]

N02BE	prophet	9.1179	inf	44.38	-3.00	181.67	13.48	{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 0.1, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': False, 'daily_seasonality': False}
N02BE	LinearRegression	10.5329	inf	47.38	-2.11	184.85	13.60	{params":{"0":{"fit_intercept":true}}}"
N02BE	SVR	12.3548	inf	52.38	-0.17	266.85	16.34	{params":{"231":{"C":5,cache_size:200,coef0:0.0,degree:2,epsilon:0.01,gamma:1,kernel:"rbf",max_iter:-1,shrinking:false,tol:0.001,verbose:false}}}"
N02BE	Ridge	11.7063	inf	52.85	-2.18	223.51	14.95	{params":{"8":{"alpha":10,normalize:true}}}"
N02BE	Lasso	13.0760	inf	59.82	-2.98	267.84	16.37	{params":{"4":{"alpha":0.1,normalize:true}}}"
N02BE	KNeighbors Regressor	11.8431	inf	55.79	-4.34	225.49	15.02	{params":{"14":{"n_neighbors":45,p:1}}}"
N02BE	GradientBoostingRegressor	8.8521	inf	33.87	1.84	150.25	12.26	{params":{"77":{"learning_rate":0.1,max_depth:7}}}"
N05B	Exponential Smoothing	3.2192	inf	63.09	0.25	18.23	4.27	[{'optimized': True, 'remove_bias': False}, {'trend': 'add', 'damped_trend': True, 'seasonal': 'mul', 'seasonal_periods': 7}, {'smoothing_level': '0.08202337828409437', 'smoothing_trend': '1.1228874665703582e-11', 'smoothing_seasonal': '0.00684182181772582', 'damping_trend': '0.9795185684306438'}]
N05B	auto_arima	3.3920	inf	63.08	0.25	18.25	4.27	[[0, 1, 3], [0, 0, 0, 0]]
N05B	prophet	3.3364	inf	73.16	-0.92	20.07	4.48	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True, 'daily_seasonality': False}
N05B	LinearRegression	3.4999	inf	63.40	0.31	19.48	4.41	{params":{"0":{"fit_intercept":true}}}"
N05B	SVR	3.4360	inf	59.54	0.80	18.82	4.34	{params":{"181":{"C":5,cache_size:200,coef0:0.0,degree:3,epsilon:0.01,gamma:0.5,kernel:"rbf",max_iter:-1,shrinking:false,tol:0.001,verbose:false}}}"
N05B	Ridge	3.3865	inf	64.07	0.08	18.13	4.26	{params":{"8":{"alpha":10,normalize:true}}}"
N05B	Lasso	3.4097	inf	64.29	0.10	18.41	4.29	{params":{"2":{"alpha":0.01,normalize:true}}}"
N05B	KNeighbors Regressor	3.3538	inf	61.88	0.33	17.88	4.23	{params":{"20":{"n_neighbors":60,p:1}}}"
N05B	GradientBoostingRegressor	3.3979	inf	61.89	0.34	18.33	4.28	{params":{"26":{"learning_rate":0.04,max_depth:4}}}"
N05C	Exponential Smoothing	0.6975	inf	49.06	-0.02	1.21	1.10	[{'optimized': True, 'remove_bias': False}, {'trend': 'mul', 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 365}, {'smoothing_level': '0.995', 'smoothing_trend': '0.995', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
N05C	auto_arima	0.8352	inf	58.90	0.12	1.22	1.11	[[0, 0, 0], [0, 0, 0, 0]]
N05C	prophet	0.8216	inf	41.45	-0.17	1.28	1.13	{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 1.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True, 'daily_seasonality': True}
N05C	LinearRegression	0.8945	inf	58.77	0.04	1.40	1.18	{params":{"1":{"fit_intercept":false}}}"
N05C	SVR	0.8257	inf	61.74	0.16	1.23	1.11	{params":{"220":{"C":5,cache_size:200,coef0:0.0,degree:2,epsilon:0.01,gamma:0.5,kernel:"rbf",max_iter:-1,shrinking:true,tol:0.001,verbose:false}}}"

N05C	Ridge	0.8272	inf	61.28	0.15	1.23	1.11	{params":{"14":{"alpha":10000,normalize:true}}}"
N05C	Lasso	0.8272	inf	61.28	0.15	1.23	1.11	{params":{"5":{"alpha":0.1,normalize:false}}}"
N05C	KNeighbors Regressor	0.8183	inf	62.91	0.18	1.28	1.13	{params":{"5":{"n_neighbors":20,p:2}}}"
N05C	GradientBoostingRegressor	0.8121	inf	65.95	0.22	1.9	1.13	{params":{"15":{"learning_rate":0.02,max_depth:9}}}"

R03	Exponential Smoothing	5.8953	inf	255.19	-4.20	80.68	8.98	[{'optimized': True, 'remove_bias': False}, {'trend': 'mul', 'damped_trend': False, 'seasonal': 'add', 'seasonal_periods': 7}, {'smoothing_level': '0.07571428571428572', 'smoothing_trend': '0.0001', 'smoothing_seasonal': '0.03423280423280423', 'damping_trend': 'nan'}]
R03	auto_arima	7.7325	inf	255.88	-4.22	80.88	8.99	[[0, 1, 1], [0, 0, 0, 0]]
R03	prophet	5.5967	inf	193.53	-2.24	69.82	8.36	{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 1.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': False, 'daily_seasonality': False}
R03	LinearRegression	6.3912	inf	152.33	0.83	75.97	8.72	{params":{"0":{"fit_intercept":true}}}"
R03	SVR	5.4220	inf	114.58	1.42	65.05	8.07	{params":{"340":{"C":15,cache_size:200,coef0:0.0,degree:3,epsilon:0.01,gamma:0.5,kernel:"rbf",max_iter:-1,shrinking:true,tol:0.001,verbose:false}}}"
R03	Ridge	5.5299	inf	121.89	1.23	65.19	8.07	{params":{"8":{"alpha":10,normalize:true}}}"
R03	Lasso	5.4434	inf	118.21	1.24	64.57	8.04	{params":{"4":{"alpha":0.1,normalize:true}}}"
R03	KNeighbors Regressor	5.4179	inf	110.17	1.79	65.98	8.12	{params":{"21":{"n_neighbors":60,p:2}}}"
R03	GradientBoostingRegressor	5.7803	inf	146.28	-0.61	63.15	7.95	{params":{"21":{"learning_rate":0.03,max_depth:7}}}"

R06	Exponential Smoothing	1.9559	inf	55.29	2.05	11.39	3.37	[{'optimized': True, 'remove_bias': False}, {'trend': None, 'damped_trend': False, 'seasonal': 'add', 'seasonal_periods': 365}, {'smoothing_level': '0.0563232331469055', 'smoothing_trend': 'nan', 'smoothing_seasonal': '1.6315533436360873e-08', 'damping_trend': 'nan'}]
R06	auto_arima	2.4665	inf	55.27	2.04	11.39	3.37	[[0, 1, 1], [0, 0, 0, 0]]
R06	prophet	1.8455	inf	76.64	-0.75	6.97	2.64	{'changepoint_prior_scale': 0.001, 'seasonality_prior_scale': 10.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True, 'daily_seasonality': False}
R06	LinearRegression	1.9807	inf	71.11	-0.14	6.57	2.56	{params":{"1":{"fit_intercept":false}}}"
R06	SVR	2.1326	inf	57.49	1.14	8.51	2.92	{params":{"319":{"C":5,cache_size:200,coef0:0.0,degree:0,epsilon:0.01,gamma:1,kernel:"rbf",max_iter:1000,shrinking:false,tol:0.001,verbose:false}}}"
R06	Ridge	1.9801	inf	55.20	0.92	7.45	2.73	{params":{"8":{"alpha":10,normalize:true}}}"
R06	Lasso	1.9762	inf	56.10	0.83	7.32	2.71	{params":{"5":{"alpha":0.1,normalize:false}}}"

R06	KNeighbors Regressor	1.9372	inf	57.42	0.57	6.91	2.63	{params":{"20":{"n_neighbors":60,p:1}}}"
R06	GradientBoostingRegressor	2.3867	inf	68.86	0.92	9.89	3.14	{params":{"118":{"learning_rate":0.6,max_depth:8}}}"

Aggregate	Exponential Smoothing	15.3140	inf	51.55	-19.09	913.56	30.23	[{'optimized': True, 'remove_bias': False}, {'trend': 'add', 'damped_trend': True, 'seasonal': 'mul', 'seasonal_periods': 365}, {'smoothing_level': '0.0405373092013567', 'smoothing_trend': '0.00010923599504998384', 'smoothing_seasonal': '0.00044535807074286797', 'damping_trend': '0.9899275466881723'}]
Aggregate	auto_arima	24.9194	inf	49.64	-17.76	864.77	29.41	[[0, 1, 2], [0, 0, 0, 0]]
Aggregate	prophet	13.7067	inf	28.83	-6.52	324.77	18.02	{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 10.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': False, 'daily_seasonality': False}
Aggregate	LinearRegression	15.7027	inf	26.79	-0.75	432.98	20.81	{params":{"0":{"fit_intercept":true}}}"
Aggregate	SVR	17.4171	inf	29.05	1.12	550.56	23.46	{params":{"231":{"C":5,cache_size:200,coef0:0.0,degree:2,epsilon:0.01,gamma:1,kernel:"rbf",max_iter:-1,shrinking:false,tol:0.001,verbose:false}}}"
Aggregate	Ridge	16.2753	inf	27.87	-0.55	475.75	21.81	{params":{"8":{"alpha":10,normalize:true}}}"
Aggregate	Lasso	15.2902	inf	26.40	-1.00	422.65	20.56	{params":{"9":{"alpha":10,normalize:false}}}"
Aggregate	KNeighbors Regressor	14.8047	inf	25.27	-1.25	403.85	20.10	{params":{"14":{"n_neighbors":45,p:1}}}"
Aggregate	GradientBoostingRegressor	15.9772	inf	27.29	-0.11	459.69	21.44	{params":{"17":{"learning_rate":0.03,max_depth:3}}}"

Figure D.2: Results of the experiments of all the sub series - Food Demand dataset

dataset	model	mae	mape	mape_adj	msd	mse	rmse	parameters
TYPE_A	Exponential Smoothing	47864	10.72	10.72	-15437	2965483101	54456	[{'optimized': True, 'remove_bias': True}, {'trend': None, 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.005', 'smoothing_trend': 'nan', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
TYPE_A	auto_arima	47747	10.53	10.53	-11720	2935784853	54183	[[0, 0, 1], [0, 0, 0, 0]]
TYPE_A	prophet	41978	9.62	9.62	3438	2967066084	54471	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': False}
TYPE_A	LinearRegression	48853	10.89	10.89	-17644	3240480163	56925	{params":{"0":{"fit_intercept":true}}}"
TYPE_A	SVR	47593	10.53	10.53	-13048	2897418504	53828	{params":{"443":{"C":15,cache_size:200,coef0:0.0,degree:0,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
TYPE_A	Ridge	48821	10.92	10.92	-19138	3064915258	55362	{params":{"8":{"alpha":10,normalize:true}}}"
TYPE_A	Lasso	49314	11.02	11.02	-19195	3095618757	55638	{params":{"14":{"alpha":10000,normalize:true}}}"
TYPE_A	KNeighbors Regressor	51569	11.51	11.51	-19943	3348648868	57868	{params":{"6":{"n_neighbors":25,p:1}}}"
TYPE_A	GradientBoostingRegressor	48964	10.97	10.97	-19504	3098209522	556612	{params":{"0":{"learning_rate":0.01, max_depth:2}}}"
TYPE_C	Exponential Smoothing	16593	12.74	12.74	-1425	458917465	21422	[{'optimized': True, 'remove_bias': False}, {'trend': 'add', 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.09928571428571428', 'smoothing_trend': '0.015274725274725275', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
TYPE_C	auto_arima	17241	12.61	12.61	876	484480205	22011	[[0, 0, 1], [0, 0, 0, 0]]
TYPE_C	prophet	16308	12.92	12.92	3228	460106230	21450	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True}
TYPE_C	LinearRegression	14995	10.99	10.99	-5225	326747567	18076	{params":{"0":{"fit_intercept":true}}}"
TYPE_C	SVR	17168	12.80	12.80	-1700	459775965	21442	{params":{"443":{"C":15,cache_size:200,coef0:0.0,degree:0,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
TYPE_C	Ridge	16956	12.86	12.86	-4012	438082823	20930	{params":{"8":{"alpha":10,normalize:true}}}"
TYPE_C	Lasso	17754	13.40	13.40	-3819	471468358	21713	{params":{"14":{"alpha":10000,normalize:true}}}"
TYPE_C	KNeighbors Regressor	16266	12.23	12.23	-3069	429787485	20731	{params":{"3":{"n_neighbors":10,p:2}}}"
TYPE_C	GradientBoostingRegressor	14620	10.95	10.95	-1815	351328302	18744	{params":{"0":{"learning_rate":0.01,max_depth:2}}}"
region_56	Exponential Smoothing	37314	9.19	9.19	-6398	1980421551	44502	[{'optimized': True, 'remove_bias': False}, {'trend': 'mul', 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.12285714285714286', 'smoothing_trend': '0.01638095238095238', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]

region_56	auto_arima	37462	9.23	9.23	-5318	2020633103	44951	[[0, 0, 1], [0, 0, 0, 0]]
region_56	prophet	36907	9.02	9.02	-3056	2168090787	46563	{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': False}
region_56	LinearRegression	40656	10.00	10.00	-18672	2423341988	49227	{params":{"1":{"fit_intercept":false}}}"
region_56	SVR	37072	8.73	8.73	2887	1947828441	44134	{params":{"0":{"C":10,cache_size:200,coef0:0.0,degree:3,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:-1,shrinking:true,tol:0.001,verbose:false}}}"
region_56	Ridge	36896	8.74	8.74	534	1898893835	43576	{params":{"8":{"alpha":10,normalize:true}}}"
region_56	Lasso	37123	8.77	8.77	1604	1942065068	44069	{params":{"14":{"alpha":10000,normalize:true}}}"
region_56	KNeighborsRegressor	43805	10.45	10.45	-9545	2551468542	50512	{params":{"0":{"n_neighbors":5,p:1}}}"
region_56	GradientBoostingRegressor	35357	8.58	8.58	-9031	1793744397	42353	{params":{"40":{"learning_rate":0.06,max_depth:2}}}"
region_93	Exponential Smoothing	1436	18.10	18.10	-59	3048019	1746	[{'optimized': True, 'remove_bias': True}, {'trend': 'add', 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.28750914436778874', 'smoothing_trend': '0.012832863305161101', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
region_93	auto_arima	1794	23.06	23.06	-833	4120651	2030	[[3, 0, 0], [0, 0, 0, 0]]
region_93	prophet	1373	27.42	27.42	-1467	6797064	2607	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True}
region_93	LinearRegression	2114	27.54	27.54	-1166	5908816	2431	{params":{"0":{"fit_intercept":true}}}"
region_93	SVR	1699	21.98	21.98	-867	3795649	1948	{params":{"0":{"C":10,cache_size:200,coef0:0.0,degree:3,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:-1,shrinking:true,tol:0.001,verbose:false}}}"
region_93	Ridge	1906	25.42	25.42	-1406	5020434	2241	{params":{"14":{"alpha":10000,normalize:true}}}"
region_93	Lasso	1906	25.42	25.42	-1406	5020397	2241	{params":{"10":{"alpha":100,normalize:true}}}"
region_93	KNeighborsRegressor	2085	27.93	27.93	-1619	6002649	2450	{params":{"15":{"n_neighbors":45,p:2}}}"
region_93	GradientBoostingRegressor	2370	29.52	29.52	-650	7541807	2746	{params":{"132":{"learning_rate":0.8,max_depth:6}}}"
meal_2290	Exponential Smoothing	30076	74.80	74.80	-30424	2071521265	45514	[{'optimized': True, 'remove_bias': False}, {'trend': 'mul', 'damped_trend': False, 'seasonal': 'mul', 'seasonal_periods': 52}, {'smoothing_level': '0.7121428571428572', 'smoothing_trend': '9.999999999999999e-05', 'smoothing_seasonal': '0.0001', 'damping_trend': 'nan'}]
meal_2290	auto_arima	22759	37.88	37.88	-10592	1267864939	35607	[[1, 0, 0], [0, 0, 0, 0]]
meal_2290	prophet	13230	46.08	46.08	20086	2321434495	48181	{'changepoint_prior_scale': 0.01, 'seasonality_prior_scale': 1.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True}

meal_2290	LinearRegression	22207	35.98	35.98	-7988	1415475696	37623	{params":{"1":{"fit_intercept":false}}}"
meal_2290	SVR	11741	14.36	14.36	5954	1181341353	34371	{params":{"443":{"C":15,cache_size:200,coef0:0.0,degree:0,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
meal_2290	Ridge	16196	24.86	24.86	-3474	1157960953	34029	{params":{"14":{"alpha":10000,normalize:true}}}"
meal_2290	Lasso	16358	25.17	25.17	-3646	1160402333	34065	{params":{"12":{"alpha":1000,normalize:true}}}"
meal_2290	KNeighborsRegressor	19088	31.20	31.20	-6701	1212366898	34819	{params":{"15":{"n_neighbors":45,p:2}}}"
meal_2290	GradientBoostingRegressor	22857	34.38	34.38	-3074	2224239251	47162	{params":{"131":{"learning_rate":0.8,max_depth:5}}}"

meal_2956	Exponential Smoothing	1486	25.11	25.11	804	5884559	2426	[{'optimized': True, 'remove_bias': False}, {'trend': None, 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.4200745557992307', 'smoothing_trend': 'nan', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
meal_2956	auto_arima	1574	30.67	30.67	522	5611690	2369	[[3, 0, 0], [0, 0, 0, 0]]
meal_2956	prophet	1704	78.57	78.57	-2118	9300032	3050	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 1.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': True}
meal_2956	LinearRegression	4576	113.82	113.82	-35	32108516	5666	{params":{"0":{"fit_intercept":true}}}"
meal_2956	SVR	2485	44.43	44.43	2485	11411988	3378	{params":{"773":{"C":50,cache_size:200,coef0:0.0,degree:0,epsilon:0.05,gamma:1,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
meal_2956	Ridge	1498	24.90	24.90	890	6105119	2471	{params":{"8":{"alpha":10,normalize:true}}}"
meal_2956	Lasso	1714	33.68	33.68	360	5418088	2328	{params":{"8":{"alpha":10,normalize:true}}}"
meal_2956	KNeighborsRegressor	1694	32.32	32.32	532	5728140	2393	{params":{"0":{"n_neighbors":5,p:1}}}"
meal_2956	GradientBoostingRegressor	2028	39.79	39.79	730	7394826	2719	{params":{"113":{"learning_rate":0.6,max_depth:3}}}"

city_473	Exponential Smoothing	861	10.54	10.54	-300	1139361	1067	[{'optimized': True, 'remove_bias': True}, {'trend': 'mul', 'damped_trend': True, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.16735834328141797', 'smoothing_trend': '0.044917335279672344', 'smoothing_seasonal': 'nan', 'damping_trend': '0.9853411553053989'}]
city_473	auto_arima	751	8.64	8.64	89	1090341	1044	[[0, 0, 1], [0, 0, 0, 0]]
city_473	prophet	953	14.64	14.64	-652	2360383	1536	{'changepoint_prior_scale': 0.1, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': True}
city_473	LinearRegression	936	10.74	10.74	394	1682424	1297	{params":{"0":{"fit_intercept":true}}}"
city_473	SVR	742	8.32	8.32	313	1147504	1071	{params":{"569":{"C":20,cache_size:200,coef0:0.0,degree:1,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:1000,shrinking:false,tol:0.001,verbose:false}}}"
city_473	Ridge	778	9.13	9.13	-51	1060376	1030	{params":{"8":{"alpha":10,normalize:true}}}"

city_473	Lasso	778	9.14	9.14	-74	1055080	1027	{params":{"10":{"alpha":100,normalize:true}}}"
city_473	KNeighbors Regressor	990	11.99	11.99	-298	1527653	1236	{params":{"5":{"n_neighbors":20,p:2}}}"
city_473	GradientBoostingRegressor	798	9.51	9.51	-124	1047827	1024	{params":{"0":{"learning_rate":0.01,max_depth:2}}}"

city_713	Exponential Smoothing	1448	9.87	9.87	-324	3095662	1759	[{'optimized': True, 'remove_bias': False}, {'trend': 'add', 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.0992810512262502', 'smoothing_trend': '9.999060700656736e-05', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
city_713	auto_arima	1499	9.85	9.85	-264	3113492	1765	[[0, 0, 1], [0, 0, 0, 0]]
city_713	prophet	1447	9.75	9.75	420	3712148	1927	{'changepoint_prior_scale': 0.001, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': True}
city_713	LinearRegression	1395	9.21	9.21	-149	2744232	1657	{params":{"1":{"fit_intercept":false}}}"
city_713	SVR	1503	9.93	9.93	-358	3118721	1766	{params":{"0":{"C":10,cache_size:200,coef0:0.0,degree:3,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:-1,shrinking:true,tol:0.001,verbose:false}}}"
city_713	Ridge	1509	9.93	9.93	-271	3081173	1755	{params":{"8":{"alpha":10,normalize:true}}}"
city_713	Lasso	1493	9.83	9.83	-297	3078859	1755	{params":{"10":{"alpha":100,normalize:true}}}"
city_713	KNeighbors Regressor	1811	11.79	11.79	-257	4215816	2053	{params":{"2":{"n_neighbors":10,p:1}}}"
city_713	GradientBoostingRegressor	1638	10.90	10.90	-568	3481851	1866	{params":{"0":{"learning_rate":0.01,max_depth:2}}}"

email	Exponential Smoothing	80942	inf	205.06	-8470	9416550977	97039	[{'optimized': True, 'remove_bias': True}, {'trend': 'add', 'damped_trend': True, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.02857142857142857', 'smoothing_trend': '0.02857142857142857', 'smoothing_seasonal': 'nan', 'damping_trend': '0.99'}]
email	auto_arima	82498	inf	206.53	-9036	9573766616	97846	[[0, 0, 0], [0, 0, 0, 0]]
email	prophet	77449	inf	187.34	-2939	8541508897	92420	{'changepoint_prior_scale': 0.001, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': True}
email	LinearRegression	82244	inf	206.93	-14156	10180550905	100899	{params":{"0":{"fit_intercept":true}}}"
email	SVR	81614	inf	195.56	-1199	9346239723	96676	{params":{"443":{"C":15,cache_size:200,coef0:0.0,degree:0,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
email	Ridge	81718	inf	211.68	-14641	9662528522	98298	{params":{"6":{"alpha":1,normalize:true}}}"
email	Lasso	83053	inf	212.81	-14338	9550387976	97726	{params":{"14":{"alpha":10000,normalize:true}}}"
email	KNeighbors Regressor	99901	inf	221.65	-19543	13560651559	116450	{params":{"0":{"n_neighbors":5,p:1}}}"

email	GradientBoostingRegressor	90691	inf	249.20	-31680	12047109829	109759	{params":{"0":{"learning_rate":0.01,max_depth:2}}}"

homepage	Exponential Smoothing	77886	66.05	66.05	-20440	8477848761	92075	[{'optimized': True, 'remove_bias': True}, {'trend': 'add', 'damped_trend': True, 'seasonal': 'mul', 'seasonal_periods': 52}, {'smoothing_level': '0.005', 'smoothing_trend': '0.005', 'smoothing_seasonal': '0.21321428571428575', 'damping_trend': '0.99'}]
homepage	auto_arima	82927	65.32	65.32	-17158	8575518355	92604	[[0, 0, 1], [0, 0, 0, 0]]
homepage	prophet	70128	48.45	48.45	-3146	8050996433	89727	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 1.0, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'multiplicative', 'weekly_seasonality': False}
homepage	LinearRegression	84516	64.79	64.79	-9423	8874498035	94205	{params":{"0":{"fit_intercept":true}}}"
homepage	SVR	81458	60.88	60.88	-6529	8102683815	90015	{params":{"443":{"C":15,cache_size:200,coef0:0.0,degree:0,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
homepage	Ridge	82012	63.70	63.70	-12162	8420041469	91761	{params":{"6":{"alpha":1,normalize:true}}}"
homepage	Lasso	82477	64.04	64.04	-15022	8285719458	91026	{params":{"14":{"alpha":10000,normalize:true}}}"
homepage	KNeighborsRegressor	80103	65.59	65.59	-21093	8404401720	91676	{params":{"4":{"n_neighbors":20, p:1}}}"
homepage	GradientBoostingRegressor	85616	63.05	63.05	-7200	9056517540	95166	{params":{"89":{"learning_rate":0.3,max_depth:3}}}"

Aggregate	Exponential Smoothing	71466	9.58	9.58	-21796	7734893337	87948	[{'optimized': True, 'remove_bias': False}, {'trend': 'add', 'damped_trend': False, 'seasonal': None, 'seasonal_periods': 52}, {'smoothing_level': '0.0757142857142857', 'smoothing_trend': '9.999999999999999e-05', 'smoothing_seasonal': 'nan', 'damping_trend': 'nan'}]
Aggregate	auto_arima	73524	9.10	9.10	-5586	7518364642	86709	[[0, 0, 1], [0, 0, 0, 0]]
Aggregate	prophet	68816	9.35	9.35	-651	8844232811	94044	{'changepoint_prior_scale': 0.5, 'seasonality_prior_scale': 0.01, 'holidays_prior_scale': 0.01, 'seasonality_mode': 'additive', 'weekly_seasonality': False}
Aggregate	LinearRegression	75437	9.52	9.52	-22880	8135227680	90195	{params":{"0":{"fit_intercept":true}}}"
Aggregate	SVR	72918	9.05	9.05	-8031	7324340075	85582	{params":{"443":{"C":15,cache_size:200,coef0:0.0, degree:0,epsilon:0.05,gamma:0.5,kernel:"rbf",max_iter:20,shrinking:false,tol:0.001,verbose:false}}}"
Aggregate	Ridge	75911	9.57	9.57	-21815	7722710816	87879	{params":{"10":{"alpha":100,normalize:true}}}"
Aggregate	Lasso	76006	9.58	9.58	-21773	7733909851	87943	{params":{"14":{"alpha":10000,normalize:true}}}"
Aggregate	KNeighborsRegressor	82542	10.41	10.41	-24853	8971105026	94716	{params":{"4":{"n_neighbors":20,p:1}}}"
Aggregate	GradientBoostingRegressor	79907	10.16	10.16	-30795	8483242183	92105	{params":{"0":{"learning_rate":0.01,max_depth:2}}}"

List of Figures

List of Tables

List of Symbols

Variable	Description
$\{y_t\}$	Time series
t	Discrete time instants
Y_t	Random variable of the target
$E[Y_t]$	First order moment
$E[Y_{t+h}Y_t]$	Second order moment
$M_t/g(t)$	Trend component
$\mathrm{m}_t(h)$	Moving average
$Q_t/s(t)$	Seasonality component
$\{\varepsilon_t\}$	Random noise
$h(t)$	Holiday component
s_t	Smoothed mean
$\alpha \in [0, 1]$	Regulates the importance of the recent value
$\beta \in [0, 1]$	Regulates the importance of the most recent value of the trend
$\gamma \in [0, 1]$	Regulates the importance of the most recent value of seasonality
e_t	Error at time t
SSE	Sum of Squared Errors
B	Backshift for autocorrelation
A_t	Function of the ARIMA model
f_{t+1}	Prediction for period t+1
(p, d, q)	Order of the non-seasonal component - ARIMA model
(P, D, Q)	Order of the seasonal component - ARIMA model
ACF_h	Autocorrelation coefficient
$PACF_h$	Partial autocorrelation coefficient
$y = f(X_1, X_2, ..., X_n)$	Regression models

Acknowledgements

Innanzitutto è doveroso ringraziare il professor Carlo Vercellis, relatore di questa tesi, per avermi dato la possibilità di approfondire un tema tanto interessante ed attuale. Grazie per avermi trasmesso la passione che mi ha permesso di indirizzare i primi passi della mia carriera professionale.

Un ringraziamento va al Politecnico di Milano, per avermi fatta crescere e diventare la persona che sono, professionalmente e personalmente.

Ringrazio tutti gli amici che mi hanno trasmesso l'entusiasmo per festeggiare i momenti più belli e il supporto per affrontare quelli meno belli. Grazie agli amici di una vita e a quelli che sono entrati di sorpresa nella mia vita.
Un grazie particolare va a Gisella, brillante collega e grande amica.

Grazie a Lorenzo, che mi ha supportata - non solo moralmente - nella stesura di questa tesi, con l'augurio che questo sia solo il primo passo *verso cose più grandi*.

Grazie ai miei fratelli, per i quali spero di essere sempre la sorella maggiore su cui contare.
A Silvia, a cui auguro di provare per sé stessa il bene che sa provare per gli altri.
A Daniele, a cui auguro di percorrere con entusiasmo la sua strada.

Grazie ai miei genitori, soci al 50%, a cui spero di dimostrare la mia immensa gratitudine. Grazie a loro sono diventata la persona che sono oggi.