UNIVERSIDADE DE COIMBRA

Pedro Moreira Costa

# Time-Series Analysis Framework

**VOLUME 1**

**Dissertação no âmbito do Mestrado em Engenharia Informática com especialização em Sistemas Inteligentes orientada pelo Professor Doutor Nuno Lourenço e apresentada ao Departamento de Engenharia Informática.**

Janeiro de 2019

Faculty of Sciences and Tecnology

Department of Informatics Engineering

# Time Series Analysis Framework

Pedro Moreira Costa

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof. Nuno Lourenço and Eng. Miguel Oliveira presented to the Faculty of Sciences and Technology / Department of Informatics Engineering

January 2019

UNIVERSIDADE Ð COIMBRA

This page is intentionally left blank.

# Abstract

A time series consists of an interesting data structure that relates an event observation with a time instance. It is the type of data that occurs in a variety of fields and whose analysis provides a more sensible understanding regarding events' behaviours. Usually, time series analysis is carried out through the use of plots and classical models. However, Machine Learning (ML) approaches have seen a rise in the state of art of popular time series analysis areas: Forecasting and Anomaly Detection. These methods provide acceptable results at appropriate time and data constraints. One of the major drawbacks in ML approaches is how the data preparation, model selection and parametrization negatively affect the results.

With this in mind, this work's main goal is to overcome the complexity involved with data processing, model selection and model tuning, through the support of an autonomous approach selector present in both forecasting and anomaly detection frameworks. The framework should handle time series of different categories and bearing different attributes and deliver the approach with the most acceptable results.

Thus far, resulting artifacts of the internship include this document, where the state of the art, regarding forecasting and anomaly detection techniques, has been explored. This includes background knowledge from the enumerated technologies as well as details regarding their use cases. The document also benefits from a collection of resources, tools, competitors, a proposed approach and the methodology in use and how it's planned.

**Keywords:** Time Series, Forecasting, Anomaly Detection, Machine Learning, Deep Learning

This page is intentionally left blank.

# Resumo

Uma série temporal é uma estrutura de dados interessante que relaciona uma observação de um evento com um instante de tempo. É o tipo de dados que é utilizado numa variedade de áreas e cuja análise providencia uma compreensão mais sensata em relação ao comportamento de um evento. É comum a análise de séries temporais ser desempenhada com recurso a vusializações gráficas e modelos clássicos. No entanto, os modelos de ML têm recebido atenção no estado da arte no que diz respeito a duas áreas populares: previsão e deteção de anomalias. Estes métodos apresentam resultados razoáveis, tendo em conta restrições de tempo e de dados. Uma das maiores desvantagens de métodos ML é o impacto que os processos de preparação de dados, seleção do modelo e a sua parametrização têm nos resultados.

Deste modo, o objectivo principal deste projeto trata por reduzir a complexidade associada à escolha do tipo de processamento de dados, seleção de modelo e a sua parametrização, através de um seletor automático da abordagem em uso, tanto para a framework de previsão, como a de deteção de anomalias. A framework deve gerir métodos de previsão e deteção de anomalias para séries temporais de diferentes categorias, constituídas por diferentes atributos, e fornecer o método com os melhores resultados.

Presentemente, os artefatos resultantes do estágio incluem este documento, onde o estado da arte, em relação a previsão e deteção de anomalias, foi explorado. Estão incluídos *background knowledge* das tecnologias enumeradas, bem como detalhes em relação aos seus casos de uso. O documento também possui uma coleção de recursos, ferramentas, competidores, uma proposta de abordagem e a metodologia em uso, bem como está planeada.

**Keywords:** Séries Temporais, Previsão, Deteção de Anomalia, Machine Learning, Deep Learning

This page is intentionally left blank.

# Contents

This page is intentionally left blank.

# Acronyms

**AIC** Akaike information criterion. 11

**API** Application Programming Interface. 34

**AR** Autoregressive. 6, 7, 8

**ARIMA** Autoregressive Integrated Moving Average. 5, 7, 8, 9, 11

**ARMA** Autoregressive Moving Average. 7

**ARMQE** Average Relative Mean Quadratic Error. 9

**AWS** Amazon Web Services. 34

**BN** Bayesian Network. 25

**BNN** Bayesian Neural Network. 11, 12, 16, 18, 22

**BSD** Berkeley Software Distribution. 33

**CART** Clasification and Regression Trees. 15, 18

**CNN** Convolutional Neural Network. 33, 34

**CNTK** Microsoft Cognitive Toolkit. 34

**CPU** Central Processing Unit. 34

**DA** Direction Accuracy. 11, 17

**DEI** Department of Informatics Engineering. 1, 47

**EML** Extreme Learning Machine. 15, 16

**FCTUC** Faculty of Sciences and Technology of the University of Coimbra. 1

**FFNN** Feedforward Neural Network. 9, 19

**GP** Gaussian Process. 14, 18

**GPU** Graphics Processing Unit. 34

**GRNN** Generalized Regression Neural Network. xiii, 17, 18

**HW** Holt-Winters Exponential Smoothing. 5, 8, 9, 11, 16, 41

**IDS** Intrusion Detection System. 24, 30

**k-NN** k-Nearest Neighbours. 14, 15, 16, 18, 33

**LSTM** Long Short Term Memory. xiii, 14, 19, 20, 21, 22, 24, 34

**MA** Moving Average. 6, 7

**MAE** Mean Absolute Error. 11, 17

**MAPE** Mean Absolute Percentage Error. 8, 9, 11, 17

**M-Competition** Makridakis Competition. vii, 1, 5, 32

**MIT** Massachusetts Institute of Technology. 34

**ML** Machine Learning. iii, v, 1, 2, 11, 12, 14, 16, 22, 32, 36, 37, 38, 42, 51

**MLP** Multi Layer Perceptron. xiii, 10, 11, 12, 14, 16, 19, 21, 22

**MSE** Mean Squared Error. 9, 17, 22

**NDM** Network Data Mining. 30

**NN** Neural Network. 11, 14, 19, 22

**OCSVM** One-Class Support Vector Machine. 24

**PCA** Principal Component Analysis. xiii, 26, 27

**RBFNN** Radial Basis Function Neural Network. xiii, 12, 13, 14, 17, 18

**RMSE** Root Mean Square Error. 11, 14, 17

**RNN** Recurrent Neural Network. xiii, 18, 19, 20, 21, 22, 24, 33, 34

**RSVM** Robust Support Vector Machine. 24

**sMAPE** Symmetric Mean Absolute Percentage Error. 11, 12, 14, 15, 16, 18

**SMO** Sequential Minimal Optimization. 11, 17, 24

**SVM** Support Vector Machine. 15, 16, 23, 24, 30, 33, 38

**SVR** Support Vector Regression. 15, 16, 22

**UCI** University of California Irvine. 33, 43

**WEKA** Waikato Environment for Knowledge Analysis. 33

This page is intentionally left blank.

# List of Figures

This page is intentionally left blank.

# List of Tables

This page is intentionally left blank.

# Chapter 1

# Introduction

This document describes the work carried out throughout the first semester, as well as the work that is expected to be fulfilled by the end of the second semester in the context of an academic internship at the Department of Informatics Engineering (DEI) from the Faculty of Sciences and Technology of the University of Coimbra (FCTUC) during the academic year of 2018/2019. The internship is taking place at Novabase[1], which is a company created in 1989 that specializes in a set of business areas (Financial Services, Energy, Transportation and Government). Recently, they've opened a new division at Instituto Pedro Nunes, named Cognitive Computing where software solutions are sought through artificial intelligence approaches. This division is where the internship is specifically taking place. Internship supervision is handled by **Eng. Miguel Oliveira** (Novabase, Cognitive Computing) and **Dr. Nuno Lourenço** (DEI, FCTUC).

As found in the literature review, time series analysis has been carried out mostly by classical approaches, but recently also by machine learning ones. In either case, analysis relies heavily on the model choice and corresponding parameter tuning in order to adapt to different time series.

Although classical approaches are well established throughout the sectors where they're applied, machine learning approaches show promise, as:

- the results are good (almost comparable to pure statistical approaches), as shown through popular time series forecasting competition, Makridakis Competition (M-Competition) [46, 47];
- some approaches are computationally adequate;
- they're more robust than classical approaches;
- they enable learning of multiple time series, for the same problem.

Given these attributes, it makes sense for Machine Learning (ML) approaches to participate in reliable time series analysis. Going back to the topic where ML approaches are more robust, one has to highlight that there is no single method which performs universally better, for whichever time series and whichever features it embeds. In most cases, the data to be fed to a model has to undergo specific data preparation techniques in order to be correctly processed for the models it is applied to.

The main objective in this project is to develop a framework which performs that kind of selection autonomously, leaving as little data preparation, model selection and parametrization decisions to the user as possible, as this process itself is rather extensive. With this in mind, the framework performs these decisions based on the given time series' features. Support for this component should be provided for two distinct analysis areas denoted in this document, which are popular when considering time series analysis: forecasting and anomaly detection.

[1] `http://www.novabase.pt/pt`

The main goal of the internship is to build an automated time series analysis framework for forecasting and anomaly detection, mainly through state of the art machine learning approaches as well as hybrid ones. During the first semester, work results include this document, where the state of the art, resources and tools are explored and competitors are enumerated and compared. With this in mind, a work plan is proposed including the requirements specification of the solution as well as an architecture and risk analysis. The methodology in use and the course of the project have also been estimated.

## 1.1 Outline

**Chapter 2** : The State of the Art is a collection of the most relevant and recent concepts that help understand how time series analysis has been analyzed throughout the years. The enumerated methods not only detail background knowledge necessary to better understand them, but are also provided with significant use cases which help reassure their strength in the literature. Regarding Forecasting, both classical and machine learning models are provided. Regarding Anomaly Detection, most of the presented approaches are ML approaches.

**Chapter 3** : This chapter highlights the resources and tools that are available and may be considered for the development of the proposed framework.

**Chapter 4** : An enumeration of market solutions entailing time series forecasting and anomaly detection is presented and discussed, in regards to the state of the art of available solutions.

**Chapter 5** : Includes the motivation that drives the internship proposal as well as the methodology that will be used. Details on requirements specification are also present.

**Chapter 6** : Presents an overview on the entire project in the context of time estimation for each of the tasks that compose this curricular unit. This chapter also details actual costs registered so far, during the course of the internship.

**Chapter 7** : Finalizes this document with an overview on the proposal as well as the main points to address during the second semester.

# Chapter 2

# State of the Art

In [20] time series are defined as "*a collection of observations made sequentially through time.*" Examples occur in a variety of fields, from economics to engineering. Methods of analyzing time series constitute an important area of statistics.

As seen in [20], Chatfield differentiates time series categories into 6 different types:

- Economic and Financial Time Series
- Physical Time Series
- Marketing Time Series
- Process Control Data
- Binary Processes
- Point Processes

Categorization of time series becomes a necessity as there is no approach with the required generalization capabilities to address each and every type of time series. To better classify time series, one should consider the following features, as seen in [20]:

**Seasonality:** Whether the data presents a pattern that repeats itself over a time interval. For example, retail sales tend to peak during Christmas season and then decline after the holidays. Thus, time series of retail sales will present increasing sales from September to December and decreasing sales in the following months. Seasonality is quite common in economic time series, but not as much in engineering and scientific data.

**Trend:** A general systematic linear or (most often) nonlinear component that changes over time but never repeats, or at least doesn't repeat for the recorded time range of the series.

**Outliers:** Observation points that are distant from other observations. Usually, the presence of outliers indicate measurement error or that the population has a heavy-tailed distribution. In the context of time series, for most cases, outliers represent anomalies in the captured data that closely relate to the nature of the time series. Assuming a time series that holds frequency data on an EEG scan, events such as an epileptic seizure set off a spike. If the series' density is high enough, a seizure may represent a fraction of the data. Thus, seizures are regarded as outliers in the distribution.

In [18], the authors reason there is a multitude of time series encountered in the fields of engineering, science, sociology and economics. The purpose of time series analysis is to draw inference from such structures.

After an appropriate collection of models is made, it is possible to estimate parameters, check for goodness of fit to the data, and use the fitted models to deepen our understanding of series generation. These models may be used to provide description of the data (through time plots), separate (filter) the noise from signals and test hypotheses such as global warming, using recorded temperature data.

There are several possible objectives in analyzing a time series [20]. These objectives may be classified as description, explanation, prediction and control.

**Description:** Usually, the first step in time series analysis is to plot the observations against time to obtain a time plot. The power of the time plot as a descriptive tool is present in holiday sales data where a regular seasonal effect, with sales 'high' in winter and 'low' in summer is frequent. Additionally, in some cases, annual sales increase (there is an upward trend). For data originated from EEG scans, time plots will often present epileptic seizures as spikes in the variable (frequency) value, suggesting that the data composing the time series is inconsistent, and these observations represent outliers.

**Explanation:** When observations are taken on two or more variables, it is possible to use the variation in one series to explain the variation in another series. This may lead to a deeper understanding of the series generation mechanism.

**Prediction:** Given an observed time series, one may want to forecast the future values of the series. This is an important task in sales forecasting, and in the analysis of economic and industrial time series.

**Control:** Time series are sometimes used to improve control over some physical or economic system. When a time series that measures the 'quality' of a manufacturing process is generated, the aim of the analysis is to keep the process operating at a highly efficient level. Control problems are closely related to prediction in many situations. If one can predict that a manufacturing process is going to move off target, then appropriate corrections can be applied.

In that sense, the following chapter sheds information on the available and state of the art approaches for time series analysis regarding forecasting and novelty detection. Both a description on the available approaches and related work are provided.

## 2.1 Forecasting

Forecasting consists in predicting unseen values within an observed time series. For instance, if we have an observed series $x_1, x_2, ..., x_n$, the goal would be to estimate values , $x_{n+h}$, where $h$ represents the *forecasting horizon.* This constitutes an important problem in areas such as economics, stock control, marketing and production planning.

While there is a wide variety of forecasting procedures, the generalization capability of the available methods has yet to reach an appropriate value [47]. Thus, there is a need to choose the most appropriate approach given a set of pre-conditions (such as those presented in the introduction of this chapter). It is also worth mentioning that forecasting is a form of extrapolation, therefore bearing all the risks of such an activity. The larger the considered *horizon*, the more prone the approach is to increase the associated error. Thus, the analyst should always be prepared to modify them as necessary in light of any incoming information.

As a side note, it is imperative to refer the Makridakis Competition (M-Competition) [44, 45, 46, 47], which is referenced throughout this section of the document. The M-Competition has become a benchmark in time series forecasting. The competitions are led by Spyros Makridakis and intend to evaluate and compare different approaches for forecasting time series.

The following subsections present state of the art approaches on forecasting. A brief description and application are provided for each of them.

### 2.1.1 Classical Models

The classical models correspond to the first, more statistical, approaches to forecasting. In this subsection, popular classical time series forecasting approaches are presented. Both a description and usage of the most prominent approaches (in this case, Autoregressive Integrated Moving Average (ARIMA) and Holt-Winters Exponential Smoothing (HW)) are highlighted.

#### Regression

**Regression** is a basic and commonly used type of predictive analysis.

There are a variety of techniques for modeling and analyzing multiple variables(from simple linear regression to discriminant analysis), where the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of Machine Learning. The simplest form of the regression equation, with one dependent variable and one independent variable, is defined as follows [39].

$$y = c + bx \tag{2.1}$$

where $y$ is the estimated dependent variable score, $c$ is a constant, $b$ is the regression coefficient and $x$ is the score on the independent variable. An graphical example of a regression is visible in figure 2.1.

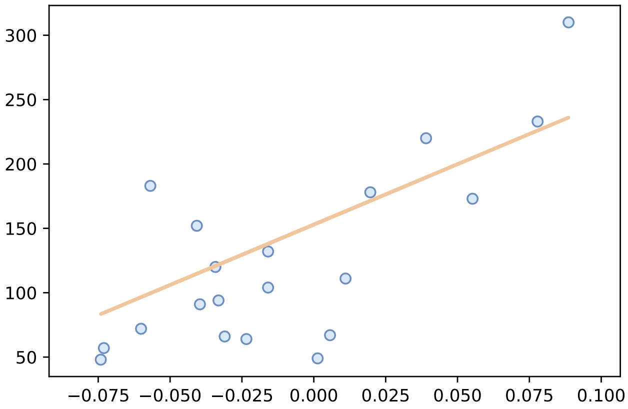


Figure 2.1: Regression applied on the Scikit-learn diabetes dataset.

### Autoregressive Model

According to [20], the idea behind Autoregressive (AR) models is to obtain the present value of the series, $X_t$, by a function of $p$ past values, $X_{t-1}$, $X_{t-2}$, ..., $X_{t-p}$.

An AR process of order $p$ is written as

$$X_t = \alpha_1 X_{t-1} + \alpha_2 X_{t-2} + ... + \alpha_p X_{t-p} + Z_t \tag{2.2}$$

where $Z_t$ is White Noise distribution, $\{Z_t\} \sim WN(0, \sigma^2)$, and $Z_t$ is uncorrelated with $X_s$ for each $s < t$.

White noise is also called a purely random process. A purely random process consists of a sequence of random variables, $Z_t$, which are mutually independent and identically distributed. Normally, it is further assumed that the random variables are normally distributed with mean zero and variance $\sigma_Z^2$.

This is rather like a multiple regression model, but $X_t$ is regressed on past values of $X_t$ rather than on separate predictor variables, which explains the prefix ‘auto’.

### Moving Average Model

In an Moving Average (MA) process, the current value is given by an expression of current and past values of the random disturbances that form a white noise series [20]. MA differs from AR because each random disturbance propagates to future values in the series. An MA process of order $q$ can be written in the form

$$X_t = \beta_0 Z_t + \beta_1 Z_{t-1} + ... + \beta_q Z_{t-q} \tag{2.3}$$

where $\{Z_t\} \sim WN(0, \sigma^2)$ and $\beta_i$ are constants. $\{Z\}$ is usually scaled so that $\beta_0 = 1$.

No restrictions on the $\beta_i$ are required for a (finite-order) MA process to be stationary,

but it is generally desirable to impose restrictions on the $\beta_i$ to ensure that the process satisfies a condition called **invertibility** [20].

### Autoregressive Moving Average

A useful class of models for time series is formed by combining AR and MA processes. An Autoregressive Moving Average (ARMA) model is suitable for univariate time series modeling. A mixed autoregressive/moving-average process containing $p$ AR terms and $q$ MA terms is said to be an ARMAprocess of order $(p, q)$ and is given by

$$X_t = \alpha_1 X_{t-1} + ... + \alpha_p X_{t-p} + Z_t + \beta_1 Z_{t-1} + ... + \beta_q Z_{t-q} \tag{2.4}$$

In [2], ARMA model manipulation is made using the lag operator notation.The lag or backshift operator is defined as

$$\phi(B)X_t = \theta(B)Z_t \tag{2.4a}$$

where $\phi(B)$ and $\theta(B)$ are polynomials of order $p$ and $q$, respectively, such that

$$\phi(B) = 1 - \alpha_1 B - ... - \alpha_p B^p$$

and

$$\theta(B) = 1 + \beta_1 B - ... - \beta_q B^q$$

The conditions on the model parameters to make the process stationary and invertible are the same as for AR or MA processes, namely, that the values of $\alpha_i$, which make the process stationary, are such that the roots of

$$\phi(B) = 0$$

lie outside the unit circle, while the values of $\beta_i$, which make the process invertible, are such that the roots of

$$\theta(B) = 0$$

lie outside the unit circle.

### Autoregressive Integrated Moving Average

In practice, many time series are non-stationary, and so the stationary models we have mentioned so far cannot be applied directly. In ARIMA modeling, the general approach is to differentiate an observed time series until it appears to come from a stationary process. Differenciating is widely used for econometric data. If $X_t$ is replaced by $\nabla^d X_t$ in Equation 2.4, then we have a model capable of describing certain types of non-stationary series. Such a model is called an 'integrated' model because the stationary model that is fitted to the differenciated data has to be summed or 'integrated' to provide a model for the original non-stationary data, as seen in [20].

Writing

$$W_t^d = \nabla^d X_t = (1 - B)^d X_t$$

the general ARIMAprocess is of the form

$$W_t = \alpha_1 W_{t-1} + ... + \alpha_p W_{t-p} + Z_t + ... + \beta_q Z_{t-q} \tag{2.5}$$

By analogy with equation 2.4a, we may write the ARIMAprocess (2.5) as

$$\phi(B)W_t = \theta(B)Z_t \tag{2.5a}$$

or

$$\phi(B)(1 - B)^d X_t = \theta(B)Z_t \tag{2.5b}$$

Thus, we have an ARMA$(p, q)$ process for $W_t$, while the model in equation 2.5b, describing the $d$th differences of $X_t$, contemplates an ARIMAprocess of order $(p, d, q)$. The model for $X_t$ is non-stationary, as the AR operator $\phi(B)(1-B)^d$ has $d$ roots on the unit circle. Usually, first differenciating is adequate to make a series stationary.

ARIMA was first published in [16] and quickly found its use for non-stationary data, like economic and stock price series. The ARIMAmodel and its different variations are based on the famous Box-Jenkins principle [7, 9, 31, 48, 56, 61] and so these are also broadly known as the Box-Jenkins models.

In [27], the airline data is used as the main example in a case study of forecasting results from a variety of neural network models against the Box-Jenkins seasonal ARIMAmodel referred to as the airline model.

In [9], findings show that modeling crude palm oil price through an ARIMAprocess is appropriate for short term predictions.

In [61], a comparison between an automatic Box-Jenkins modeling expert system, AUTOBOX, and a Neural Network was carried out, using a portion of time series stemming from the famous M-Competition [44]. Results showed that there was promise in using neural network models as these appeared to present similar error (Mean Absolute Percentage Error (MAPE)) in the generality of the used time series.

Several articles [7, 31, 56] also sought out to estimate machine learning methods' performance in relation to traditional methods, always including ARIMA in their experiments. Conclusions in this study wager traditional methods' appropriation in time series on annual data, while the neural networks in use are more suitable for monthly and quarterly data, as well as for unstable data (namely discontinuities).

### Holt-Winters Exponential Smoothing

Exponential smoothing's application in forecasting was first introduced by Robert Brown in [28]. In 1957, Professor Charles C. Holt (1921-2010), MIT and University of Chicago graduate, was working at the Carnegie Mellon University on forecasting trends in production, inventories and labor force.

Holt published "*Forecasting trends and seasonals by exponentially weighted moving averages*" describing double exponential smoothing. Three years later, Peter R. Winters, his student, improved the algorithm by including seasonality and published "*Forecasting sales by exponentially weighted moving averages*", citing Holt's 1957 paper. Thus, this algorithm became known as triple exponential smoothing or HW.

The following paragraphs provide insight on the process from Single Exponential Smoothing to Triple Exponential Smoothing.

**Single Exponential Smoothing** Also known as simple exponential smoothing, it is used for short-range forecasting. This model assumes that there is no trend associated to the data. The formula is defined as follows, according to [1].

$$S_t = \alpha * y_t + (1-\alpha) * S_{t-1} \tag{2.6}$$

where $S_t$ stands for the smoothed observation, $y_t$ is the original observation and $\alpha$ represents the smoothing constant. In this model, the new predicted value is based on the old one in addition to an adjustment for the error associated with the last forecast.

**Double Exponential Smoothing** The Single Exponential Smoothing does not prove to be an appropriate method when the data follows a trend. This case can be corrected by introducing a new equation, with a second constant, $\gamma$. In Double Exponential Smoothing, both *level* and *trend* are updated for each period $t$.The level is an estimate of the data value at the end of each period, while the trend is an estimate of the average growth at the end of each period. The resulting equations are as follows, according to [1].

$$S_t = \alpha * y_t + (1 - \alpha) * (S_{t-1} + b_{t-1}) \qquad 0<\alpha<1 \tag{2.7}$$

$$b_t = \gamma * (S_t - S_{t-1}) + (1 - \gamma) * b_{t-1} \qquad 0<\gamma<1 \tag{2.8}$$

Note that the current value of the series is used to compute its smoothed value replacement in double exponential smoothing.

The initial value for $S_t$ is generally $y_1$, while, for $b_1$, in [35] the following suggestions are presented:

- $b_1 = y_2 - y_1$
- $b_1 = [(y_2 - y_1) + (y_3 + y_2) + (y_4 - y_3)]/3$
- $b_1 = (y_n - y_1)/(n - 1)$

**Triple Exponential Smoothing** When the data shows both trend and seasonality, double smoothing is not enough. Thus, a third equation that takes care of seasonality is introduced. The resulting set of equations is named the "Holt-Winters" method, named after the inventors. The basic equations for this method are given by

$$S_t = \alpha \frac{y_t}{I_{t-L}} + (1 - \alpha)(S_{t-1} + b_{t-1}) \tag{2.9}$$

$$b_t = \gamma(S_t - S_{t-1}) + (1 - \gamma)b_{t-1} \tag{2.10}$$

$$I_t = \beta \frac{y_t}{S_t} + (1 - \beta)I_{t-L} \tag{2.11}$$

$$F_{t+m} = (S_t + mb_t)I_{t-L+m} \tag{2.12}$$

where $y$ is the original observation, $S$ is the smoothed observation, $b$ is the trend factor, $I$ is the seasonal index, $F$ is the forecast at $m$ periods, $t$ is an index denoting a time period and $\alpha$, $\beta$ and $\gamma$ are constants that must be estimated with the goal of minimizing the Mean Squared Error (MSE).

To initialize Holt-Winters we need at least one complete season's data to determine initial estimates of the seasonal indices $I_{t-L}$. A complete season's data consists of $L$ periods. To estimate the trend factor, is it advisable to use two complete seasons, that is, $2L$ periods.

Holt-Winters Exponential Smoothing is a prominent traditional method with application in time series forecasting. In [35], a detailed analysis on the weight of both multiplicative and additive models of HW when confronted with multiplicative and additive seasonality is made, obtaining an MAPE as low as 17.48%. Authors [7, 31] draw comparisons not only between machine learning methods and ARIMA, but also HW. In [37], a modified HW method where the forecasting horizon, $m$, is square rooted, leads to a more conservative trend extrapolation, and is pitted against Feedforward Neural Network (FFNN). Results showed that the best configured FFNNs obtained an Average Relative Mean Quadratic Error (ARMQE) comparable with that of a commonly configured HW model.

### 2.1.2 Machine Learning Models

While Machine Learning approaches have been theorized long ago, their application in the domain of forecasting is relatively recent. However, one can assume that considering the state of the art on time series analysis, Machine Learning is present and highlighted in the methods that follow in this subsection.

#### Multilayer Perceptron

The Multi Layer Perceptron (MLP) finds its base on the Perceptron algorithm. The **Perceptron** is a linear binary classifier for supervised learning. In that sense, classification is carried out through a linear prediction function which combines a set of weights and bias with a feature vector.

In [52], the authors define that "a perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise." Given inputs $x_1$ through $x_n$, the output $o(x_1, ..., x_n)$ computed by the perceptron is

$$o(x_1, ..., x_n) = \begin{cases} 1, & if\ w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n > 0 \\ -1, & otherwise \end{cases}$$

where each $w_i$ is a real-valued *weight*, that determines the contribution of input $x_i$ to the perceptron output. The quantity $(-w_0)$ is a threshold that the weighted combination of inputs $w_1x_1 + ... + w_nx_n$ must surpass in order for the perceptron to output 1. To simplify notation, an additional constant $x_0 = 1$ is introduced, such that the above inequation is rewritten in the form $\sum_{i=0}^{n} w_ix_i > 0$, or in vector form, $\vec{w} \cdot \vec{x} > 0$.

Single perceptrons can only express linear decision surfaces. However, multilayer networks learned by the backpropagation algorithm are capable of expressing a rich variety of non-linear decision surfaces. The MLP , at its most bare form, is a set of perceptrons with one additional layer of nodes (referred to as hidden layer). More complex variations include more than one hidden layer, as the MLP is a heavily parametrized model. The general representation for the MLP is show below:

$$\hat{y} = \upsilon_0 + \sum_{j=1}^{NH} \upsilon_j g(w_j^T x') \tag{2.13}$$

where $x'$ is the input vector $x$, augmented with 1, i.e., $x' = (1, x^T)^T$, $w_j$ is the weight vector for $j$th node, $\upsilon_0, \upsilon_1, ..., \upsilon_{NH}$ are the weights for the output node, and $\hat{y}$ is the network output. Function $g$ represents the hidden node output, and is given in terms of a squashing function.

The MLP uses a supervised learning technique called backpropagation for training. The backpropagation algorithm learns the weights, given a network with a fixed set of units and interconnections. In order to minimize the squared error between the network output and target values, it employs gradient descent.

The error in output node $j$, for the $n^{th}$ training example, is given by

$$e_j(n) = t_j(n) - o_j(n)$$

where $t$, $o$ are the target and output values. The node weights are adjusted based on corrections that minimize the error in the entire output:

$$\varepsilon(n) = \frac{1}{2} \sum_j e_j^2(n)$$

Using gradient descent, the change in the weights is given by

$$\Delta w_{ji}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial \upsilon_j(n)} y_i(n)$$

where $y_i$ is the output of the previous neuron and $\eta$ is the learning rate.

The MLP has a strong presence in the state of the art of time series forecasting. [56] presents a case study discussing the comparison between supervised artificial neural network and ARIMA models. As part of the analysis, one of the first problems to occur was the search of appropriate input lag terms for training the network. This led the experiment to account for performance of different NN models with varying order of the network, number of parameters and activation function in use. Results showed that the the most appropriate model in the fitted Neural Network (NN) models showed a lower SSE than the ARIMAX fitted model.

In [6], a large scale comparison study for major machine learning models, including MLP , for time series forecasting took place, using time series provided from the M3 Competition. While different preprocessing techniques were approached, both MLP and Bayesian Neural Network (BNN) stood out as presenting the best overall results. In this study, the performance metric in use was the sMAPE-TOT. First, each model ran 10-fold validation, so that an overall Symmetric Mean Absolute Percentage Error (sMAPE) for each model could be obtained. The sMAPE-TOT accounts for the total sMAPE on all 1045 considered time series. The average sMAPE is computed from the validations. Result analysis led to believe that for each time series category, MLP consistently presented the lowest (best) sMAPE-TOT. As an addition, a time complexity study was provided, showing that while MLP presented the best sMAPE-TOT, it was not necessarily the fastest.

In [36], the correlation between crude palm oil price (CPO), selected vegetable prices, crude oil and the monthly exchange rate is explored. Preliminary analysis showed a positive, high correlation between CPO price and soy bean oil price, and CPO price and crude oil price. The undergoing experiments used MLP , Support Vector Regression with Sequential Minimal Optimization (SMO) and HW. Error metrics in use include Root Mean Square Error (RMSE), Mean Absolute Error (MAE), MAPE (and Direction Accuracy (DA) as a means to understand whether the forecast value tends in the same direction as the series). In this study, SMO was consistently more accurate than MLP , but the worst model in use was HW.

As expected, MLP made its way onto the M3-Competition. In [48], Makridakis et al. assess machine learning models' disadvantages when compared to classical approaches, both in forecasting accuracy and computational complexity. In that sense, this paper proposes possible ways forward as well as an explanation for ML approaches' sub-par performance when compared to classical approaches. One of the major doubts is present in the error associated with MLP applied at the most appropriate preprocessing, when compared with that of ARIMA and the seasonally adjusted random walk model, which appear to perform marginally better. One way to improve forecasting on current ML approaches is to have the data deseasonalized. Another, is to have Machine Learning (ML) methods access information about surpassing the available data for training, with the objective being to minimize future errors rather than fitting the model to available data. Another important concern is the extent of randomness associated with the series and the ability of the ML models to differentiate noise from the data, all while avoiding over-fitting. This task can prove difficult as, unlike classical models, where noise is easily parametrized by information criteria (such as Akaike information criterion (AIC)), ML methods are non-linear and training is performed dynamically.

### Bayesian Neural Network

Bayesian Inference is a statistical inference method in which Bayes' theorem is used to update probabilities in a hypothesis as more information is committed. The prior, $p(\theta)$, is the *prior* probability of a parameter $\theta$ before having seen the data. The likelihood, $p(D|\theta)$, is the probability of the data $D$ given $\theta$. Using Bayes' rule, we can determine the *posterior* probability of $\theta$ given the data, $D$, as

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{2.14}$$

In general, this provides an entire distribution for values of $\theta$ rather than a single most likely value.

A BNN is a neural network based on Bayesian inference, thus related to the statistical concept of Bayesian parameter estimation as well as regularisation [14]. BNNs closely resemble MLP, but weights are observed as random variables which obey some prior distribution. According to [6], this distribution is designed to favor low complexity models. Following data observation, the posterior distribution of the weights is evaluated and the network prediction can be assessed. Predictions reflect both smoothness imposed through the prior and fitness accuracy imposed by the observed data.

As seen for MLP, the comparison study in [6] included BNNs. Both MLP and BNN stood out as presenting the best overall results, with the BNN fitted model only marginally worse than MLP. The provided time complexity study showed that BNN had a slower execution than MLP, suggesting that this approach, while interesting considering sMAPE values, might not be the most appropriate, at least for the considered time series.

The M3 Competition dataset once again provides insight into ML forecasting techniques, namely BNN [48]. Considering only ML approaches, BNN manages to obtain the lowest overall sMAPE, followed closely by MLP .

### Radial Basis Function Neural Network

In the Perceptron and MLP, separability is linear, as they're composed of input and output layers (although MLP attains non-linearity because of hidden layers). Thus, to deal with non-linear separation, at least one hidden layer is necessary.

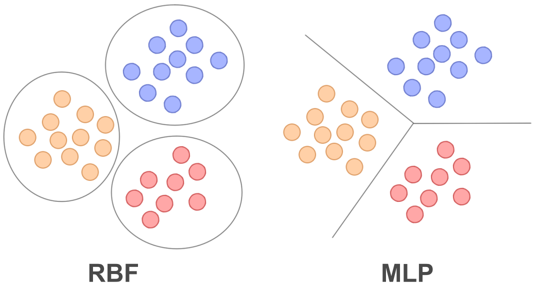


Figure 2.2: Distinction between MLP and Radial Basis Function Neural Network (RBFNN).

The RBFNN appears similar in structure to the MLP, but in its architecture only one hidden layer is allowed, as proposed in [19].

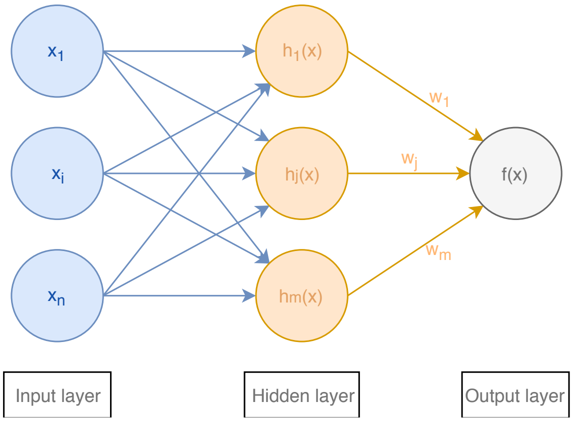


Figure 2.3: Traditional RBFNN.

Thus, as seen in figure 2.3, an RBFNN has exactly one input, one hidden and one output layer. Here, the hidden layer is typically referred to as feature vector and each of its neurons houses a basis function. The output of the RBFNN is a linear combination of the hidden nodes, defined as

$$y = \sum_{j=1}^{m} w_j h_j(x) \tag{2.15}$$

where $w_j$ denotes the weight on hidden node $j$ and $h_j(x)$ the basis function.

Any set of functions can be used as the basis set, although a well behaved set (differentiable) is preferred. Here, a special family of functions is highlighted: the radial functions. Radial functions' characteristic feature is that their response decreases (or increases) monotonically with distance from a center point. A frequently used radial function is the Gaussian which, in the case of a scalar input, is defined as

$$h(x) = exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{2.16}$$

with parameters $c$ being the center and $r$ being the radius.

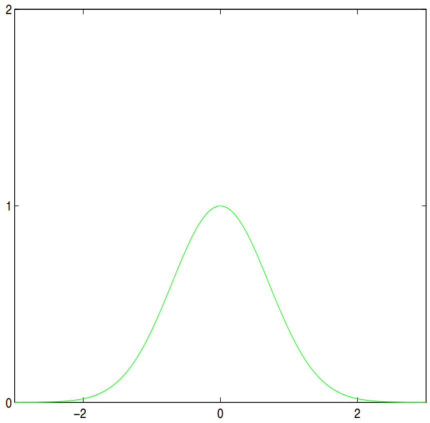


Figure 2.4: Gaussian RBF with center 0 and radius 1.

RBFNNs were first formulated in [19], as a mean to turn interpolation between known data points explicit. RBFNN learning is equivalent to solving a set of linear equations, and, while these networks represent non-linear relationships, they have a "*guaranteed learning rule*".

In [22], a demonstration on the use of neural networks in solar radiation modeling is presented. The NNs in use are the MLP and RBFNN. The RBFNN architecture in use comprises 5 input nodes (month of the year, latitude, longitude, altitude and sunshine ratio), hidden nodes use gaussian function and the output node's activation function is linear. Performance evaluation through the use of RMSE showed that performance of the different approaches are similar, but the RBFNNs are recommended as they "*don't need as much computing power as the MLP networks*".

As was the case for the previously highlighted ML approaches, in [6] the RBFNNs are also subject to comparison using the M3-Competition monthly series data. Although MLP and Gaussian Process (GP) present the best overall performance (sMAPE-TOT), it becomes clear in the analysis of computational complexity that RBFNNs present a clear advantage over MLP , while GP continue to execute in less time than the former two. It is important to emphasize the RBFNN 's poor overall performance on all the used time series, for the MOV-AVG preprocessing method, achieving the worst sMAPE of all considered ML approaches. The study on the relation between computational complexity and performance metric could shed some information on which model to adopt, given cases where this multi-objective scenario needs to be handled.

While the previous use case denoted this method's poor performance, things seem brighter for the study conducted on the M3-Competition dataset, as seen in [48], where the authors compare their results to the ones obtained in [6]. While the sMAPE has improved greatly, computational complexity seems to have degraded, now being worse than both MLP and GP. These results account only for the one step ahead forecasting horizon, but considering the most appropriate preprocessing method per considered approach.

### k Nearest Neighbor Regression

k-Nearest Neighbours (k-NN) is a non-parametric regression method that bases its forecasts on a similarity measure (distance function) from the new data points to the training set and outputs a prediction value based on the average of the $k$ nearest neighbors [52]. This approach is considered a lazy algorithm because it doesn't learn a discriminative function, but instead stores the entire training dataset. In a logistic regression algorithm, training time entails model weight learning, while the same doesn't happen for k-NN. Thus, training time in k-NN is null. However, the prediction process is costly: every time we want to make a prediction, the algorithm computes the similarity for the $k$ nearest neighbors, with the prediction being the result of an average on the previous $k$ closest points. The most common distance functions are shown below.

$$D_{Euclidean} = \sqrt{\sum_{i=1}^{k}(x_i - y_i)^2} \tag{2.17a}$$

$$D_{Manhattan} = \sum_{i=1}^{k}|x_i - y_i| \tag{2.17b}$$

$$D_{Minkowski} = \left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{\frac{1}{q}} \tag{2.17c}$$

Handling regression problems, it seems logical for k-NN to be targeted for analysis on time series forecasting. In [6], k-NN is compared with several other ML approaches. Results show that for the overall performance on all the used time series, this approach presents one of the worst results of the comparison, while the MLP stands out. However, as expected, the computational complexity of this algorithm is very reduced. Results on this variable show that k-NN has the lowest computation time of all methods.

In the M-3 Competition, results analysis also befalls on k-NN [48]. Yet again, the forecasting performance on this method falls short of the competition, scoring the fourth to worst overall result, only slightly better than Long Short Term Memory (LSTM), but still worse than the Ahmed's study. Regarding computational complexity, it is shown again that k-NN executes quickly, even though its accuracy (sMAPE) is slightly degraded.

In [41], the goal is to reduce the cost related to management of the monitoring systems on solar radiation. To achieve this, experiments entailing plausability on the short term prediction of the solar radiation based on data collected in the near past on the same site and at weather stations located 10 kilometers away are conducted. The prediction techniques in use are Support Vector Regression (SVR), Extreme Learning Machine (EML) and Autoregressive models. these approaches are then compared with persistence and k-NN predictors. When comparing the presented approaches with k-NN, results show that k-NN is able to achieve performance comparable with more sophisticated models.

### Classification and Regression Trees

Decision Trees are an important type of algorithm for machine learning prediction modeling. The classical decision tree algorithms have been around for decades and modern variations are among the most powerful techniques available. Clasification and Regression Trees (CART) is a classification or regression model based on a hierarchical tree-like partition of the input space [17]. The input space is divided into local regions identified in a sequence of recursive splits. The tree structure consists of internal nodes and terminal leaves. Given a data example, the tree is traversed, through binary decisions, from the root node to a final leaf node, where a prediction is made.

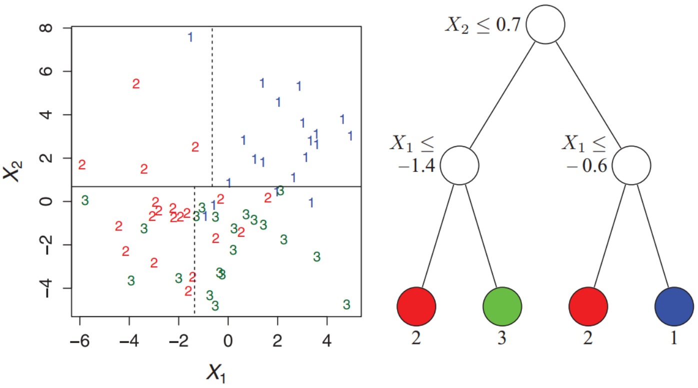


Figure 2.5: Input space partitions (on the left) in relation to the tree structure (on the right). Based on [43].

The CART was originally proposed in [17]. Its usage in time series forecasting was frequently highlighted in papers that were previously referenced in this document, in [6] and [48]. In the former, CART sMAPE positioned the algorithm at the second to worst result, while for the latter, it presents average results, considering ML methods.

### Support Vector Regression

Support Vector Machine (SVM) (proposed in [15]) integrate a specific class of algorithms, characterized by their usage of kernels, absence of local minima, sparseness of the solution and capacity control obtained by parametrization of the margin and the number of support vectors. SVMs are frequently used in classification problems where different classes are not linearly separable at their original dimensionality. Thus, the use of kernels projects data features into a higher dimension where linear separation hyperplanes can be achieved. The capacity of the system is controlled by parameters that do not depend on the dimensionality of feature space.

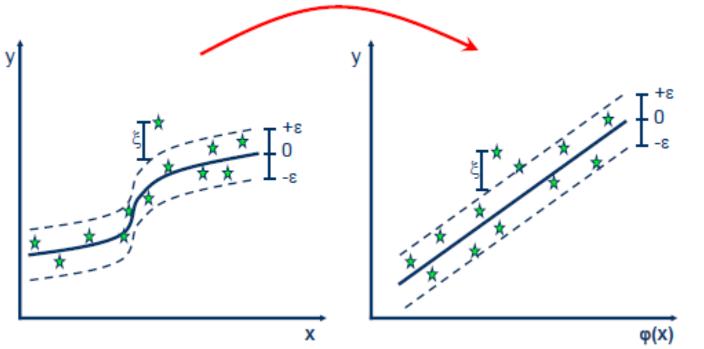


Figure 2.6: SVM kernel projects the data into a higher dimensional feature space to allow linear separation. Based on [60].

SVMs can be applied not only to classification problems, but also regression. SVR is the regression process performed by a SVM which tries to identify the hyperplane that maximizes the margin between two classes while minimizing the total error under tolerance. The error function is given by

$$J = \frac{1}{2}\|w\|^2 + C\sum_{m=1}^{M}|y_m - f(x_m)|_e \tag{2.18}$$

where the first term penalizes model complexity and the second term describe the $\epsilon$-insensitive loss function, defined as

$$|y_m - f(x_m)|_e = max\{0, |y_m - f(x_m)| - \epsilon\} \tag{2.19}$$

Errors below $\epsilon$ are not penalized, allowing the parameters to adjust, in order to reduce model complexity.

SVRs were first documented in [23] and have had a frequent presence in the domain of time series forecasting. Recurrent references, [6] and [48], included the SVR in their comparison study of classical and machine learning approaches. Results on the former portray a reasonable overall performance, for the LAGGED-VAL pre-processing, while the remaining choices deteriorate SVR and most of the methods' sMAPE. For computational complexity, the SVR also stands reasonable, at fifth for the longer computation time. Regarding the latter, the used configuration showed notable improvements. The kernel in use is the radial basic one, $\epsilon$ is set to the noise level of the training sample while $C$ is fixed to the maximum of the target output values. Next, both the $\gamma$ parameter and number of inputs $N$ are optimized through 10-fold validation. Results show promise of this approach, as it stands right behind MLP and BNN which consistently obtain good results for pure ML methods.

In [41], SVRs were one of the implemented models to pitch against the persistence and k-NN predictors. Parameter optimization (input dimensionality, $\epsilon$ and regularization trade-off) was done ad-hoc and the effectiveness was assessed through cross validation. SVR appropriation was tested in the second experiment (remote measurements) through the prediction error defined within this paper. The model with the lowest achieved error was the best SVR model (error of 40.5, when compared to the best k-NN which scored 41.4, followed by the best EML model, with error 42.7).

In [67], an overview on machine learning methods used for prediction is given. Methodologies such as regression trees, random forest and gradient boosting are highlighted to show that more methods, other than the popular neural networks and SVM/SVR, are being used in the context of forecasting, in the environment of solar radiation. In the conclusions, the authors assess SVM/SVR frequency, noting that in the following years they expected this method's use to rise, given that "*results (...) are very promising and some interesting studies will certainly be produced the next few years*".

In [36], the aim was to study variable correlation (crude palm oil price, selected vegetable oil prices and others), followed by price forecasting. Methods in use for forecasting included MLP , HW and SVR. In this particular case, the authors used a variant proposed in [63] referred to as SVR with

SMO, with the consideration that "*SMO has the good ability to model regression, prediction with non-linear data*". Results show that, for all considered performance metrics (MAE, DA, MAPE, RMSE and MSE), SMO stood out with considerably lower error.

### Generalized Regression Neural Network

The Generalized Regression Neural Network (GRNN), also called Nadaraya-Watson estimator or kernel regression estimator was first proposed by Nadaraya and Watson in [55, 69]. The machine learning implementation (usually referred to as GRNN) was proposed by Donald Specht in [65]. The GRNN is a non-parametric model where predictions are computed by averaging the target outputs of the training dataset points according to their distance from the observation.

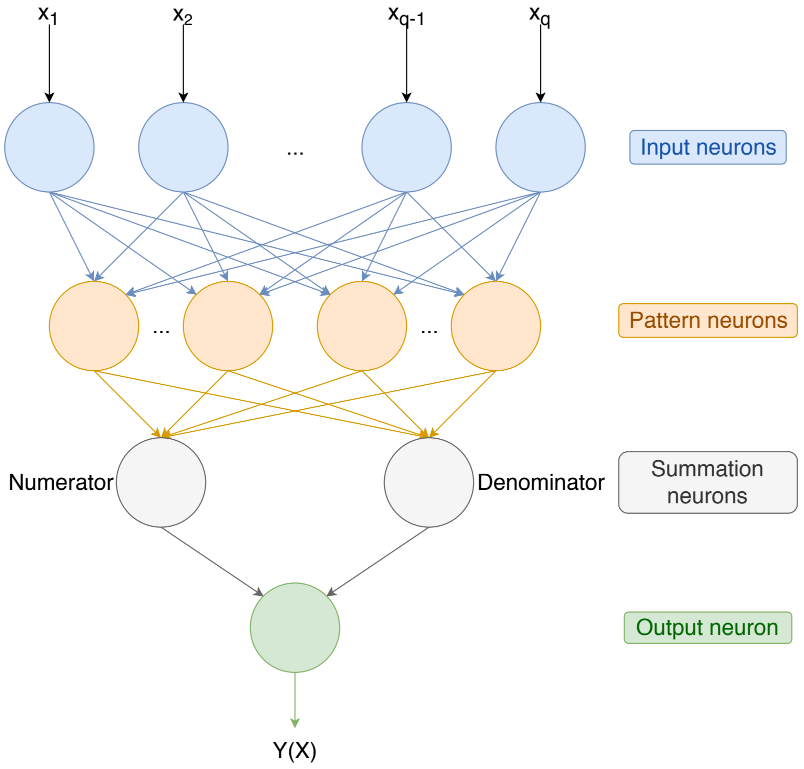


Figure 2.7: GRNN built for use as a parallel Neural Network. Based on [12].

As shown in [6], the estimation is the weighted sum of the observed responses, given by

$$\tilde{y} = \sum_{m=1}^{M} w_m y_m, \tag{2.20}$$

where the weights, $w_m$, are given by

$$w_m = \frac{\mathfrak{K}\left(\frac{\|x-x_m\|}{h}\right)}{\sum_{m'=1}^{M} \mathfrak{K}\left(\frac{\|x-x_{m'}\|}{h}\right)}, \tag{2.21}$$

where $y_m$ is the target output for training data point $x_m$, and $\mathfrak{K}$ is the kernel function. The bandwidth, $h$, is an important parameter as it determines the smoothness of fit.

The GRNN is similar in structure to the RBFNN , having the following properties in common:

- One-pass learning, such that no backpropagation occurs;
- High accuracy in estimations, as gaussian functions are used;

- Robust to noise in the data.

However, as seen in the following use cases, the GRNN also has some disadvantages:

- It can increase in size, as the number of pattern neurons should be the same as the number of input neurons. This leads to a high computational complexity;
- There is no optimal method to improve this structure.

GRNNs have been explored in the literature. In [6], the gaussian kernel function is used. In order to assess the $h$ parameter fit, 10-fold validation is performed. Results highlight the GRNN (and the CART) as the method with the best rank out of all the tested methods, for the DIFF pre-processing method. Generally, the GRNN presents satisfactory results, placing this approach in the middle of the ranking of all considered methods. Regarding computational complexity, the GRNN holds the third place for fastest executing algorithm.

The same methodology for parameter selection is adopted in [48], for both the $h$ parameter and number of inputs $N$. Implementation details state this method was designed using the `R` statistical package. Results for the one-step-ahead forecasts with the most appropriate pre-processing technique per method show improvements over Ahmed et al. variant, decreasing the sMAPE by almost 1%, positioning this method behind Recurrent Neural Networks (RNNs) and in front of RBFNNs. Overall, the performance of this method shows promise, especially considering its low computational complexity.

### Gaussian Process

In a simple linear regression, we have a dependent variable $y$. It is assumed that this variable can be modeled as a function of an independent variable, $x$ through the expression

$$y = f(x) + \epsilon, \tag{2.22}$$

where $\epsilon$ is the error associated with the prediction. It is further assumed that function $f$ defines a linear relationship, such that the objective is to find the slope and intercept of the line:

$$y = mx + b + \epsilon \tag{2.23}$$

In [10], it is highlighted that Bayesian linear regression "*provides a probabilistic approach to this by finding a distribution over the parameters*".

However, the GP approach is non-parametric, in the sense that it finds a distribution over the possible functions $f(x)$ that are consistent with the observed data. As with Bayesian methods, it starts with a prior distribution and updates the data as points are observed, outputting posterior distributions over functions. A GP assumes that $p(f(x_1), ..., f(x_N))$ is jointly Gaussian, with some mean $\mu(x)$ and covariance $\sum(x)$, given by $\sum_{ij} = k(x_i, x_j)$ where $k$ is a positive definite kernel function. If $x_i$ and $x_j$ are deemed by the kernel to be similar, the output of the function is expected to be similar too. [54].

GPs have made recent appearance in the forecasting community, namely in popular studies [6] and [48]. In the former, parameter optimization follows the model selection algorithm proposed in [59], as it maximizes the marginal likelihood function. The authors state that such a criterion function doesn't favor complex models, thus overfitting is unlikely. Results show that while the LAGGED-VAL pre-processing presents the best results overall, GP sMAPE-TOT performed considerably well, independently of the pre-processing technique, always ranking in the top three lowest sMAPE methods. Data category results favor this model in regards to demographic time series. In regards to computation complexity, GP performs similarly to RBFNN , while BNN is severely slower in execution and k-NN is faster. Makridakis' study reveals a slight improvement in sMAPE, surpassing RNN , GRNN and RBF. This could be due to parameter (number of input variables, kernel initial noise variance and tolerance of termination) optimization, which undergoes 10-fold validation. The kernel in use is the radial basis.

### Long Short Term Memory

To better understand the Long Short Term Memory Neural Network, one must first consider the Recurrent Neural Network [24]. The RNN, also known as Elman Network, is similar to MLP in structure, but includes a feedback loop, such that the output from step $n-1$ is fed back to the net to affect the output from step $n$ and so forth, for each subsequent step. For example, if a RNN is exposed to a word letter by letter and a suggestion for the following letter is requested, the first letter of the word can help determine what the RNN thinks the second letter can be.

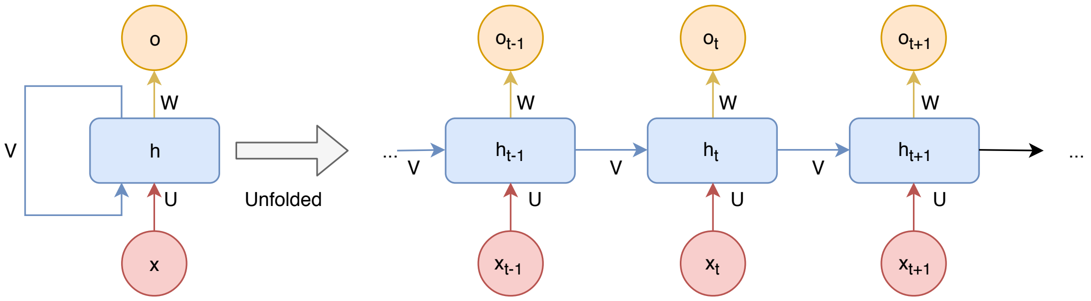


Figure 2.8: Unfolded RNNdiagram. Based on [57].

RNNs are a powerful set of NN algorithms especially useful for processing sequential data such as sound, time series or written natural language. RNNs and FFNNs both "remember" something about the data they're exposed to. After training, FFNNs produce static models of the training data that can classify new examples with acceptable accuracy. In RNNs, the models exhibit dynamic temporal behavior, so the classification always depends on the context of the observations they're exposed to. This is possible because of the RNNs' hidden states that determine the previous classification in a series. In each subsequent step, that hidden state is combined with the following step's input data to produce a new hidden state and a new classification.

Just as neural networks as a whole take inspiration in the human brain, RNNs mostly weight on the human memory aspect. Human memories are context-aware, recycling previous awareness cases to properly interpret new situations. For example, assuming two people in the following circumstances: one is expecting a delivery from Amazon; the other has a close relative in the hospital. Both receive a phone call, and, while for the person expecting the delivery the thought is that the courier couldn't deliver the order, the person with the relative at the hospital might react worse. Interpretation for the phone call is different for each of them, because they retain a hidden state affected by their short-term memories and preceding sensations.

Different short-term memories should be recalled at different times and situations, in order to assign the correct meaning to the current input. Some of these memories are more recent than others. With this in mind, the RNN that binds memories and time input is called a LSTM.

The LSTM is similar to the RNN described above, only it avoids the long-term dependency problem of the former [32]. The evident advantage in LSTMs is their ability to keep information over long periods of time due to the structure of the units that compose them. These units consist of several gates that manage the weight attributed to the information in the current state.

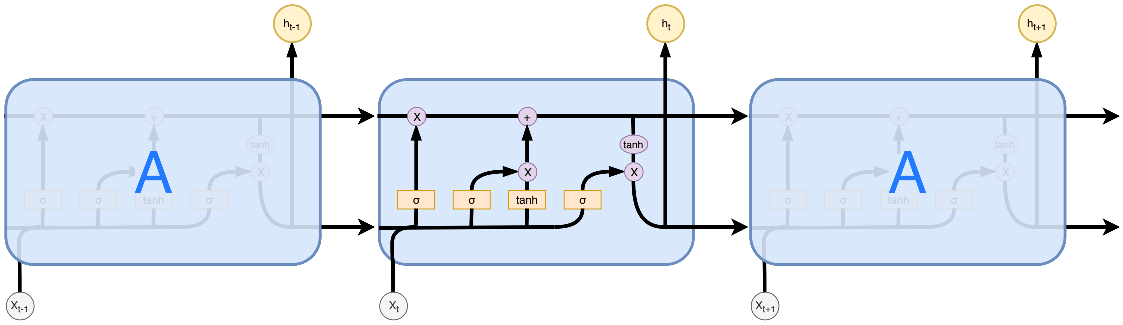


Figure 2.9: Module present in an LSTM. Based on [57].

As seen in figure 2.9, what differentiaties LSTMunits from RNNis the number of gates per unit.

In LSTMs, the cell state is the straight horizontal line running through the top of the diagram. The cell state is the core of the LSTMallowing for information to flow along the network with minimal changes. Using the results from the gates, it is possible to modify the information.

LSTM gates are structures that optionally let information through to the cell state. They incorporate a sigmoid neural net layer (with outputs between 0 and 1) and a point-wise multiplication operation.

The first gate layer manages what information is kept or thrown away and is dubbed *forget gate layer*.

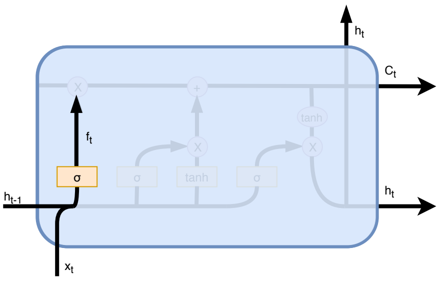


Figure 2.10: Module present in an LSTM- forget gate layer highlighted. Based on [57].

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.24}$$

It looks at the new input, $x_t$, and the previous module's output, $h_t$, and outputs a number between 0 and 1, for each number in cell state $C_{t-1}$.

The second gate layer decides the weight of new information on the cell state. This gate is called *input gate layer*.

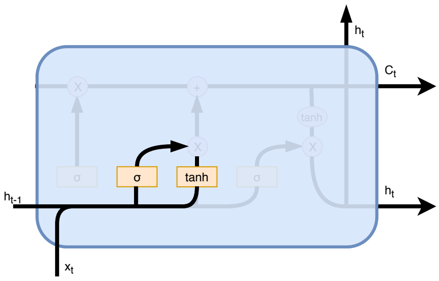


Figure 2.11: Module present in an LSTM- input gate layer highlighted. Based on [57].

First, a sigmoid layer decides which values to update.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.25}$$

Next, a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{2.26}$$

Finally, these vectors are combined to create an update to the state.

In the previous layers we decided what old information to keep and what new information to add. At this point, from the old state, $C_{t-1}$, the new state, $C_t$, is created.

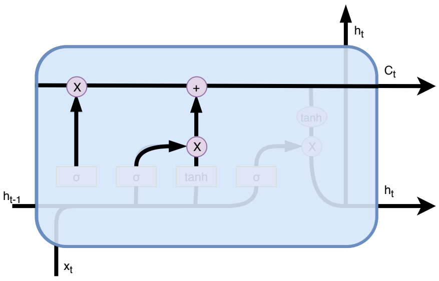


Figure 2.12: Module present in an LSTM- new state creation highlighted. Based on [57].

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{2.27}$$

The output from the module is now dependent on a filter from the new state. First, a sigmoid layer decides what parts of the cell state are output. Next, the cell state passes *tanh* to normalize the values between $-1$ and 1. Finally, these values are multiplied by the output of the sigmoid gate, so that only the wanted parts are output.

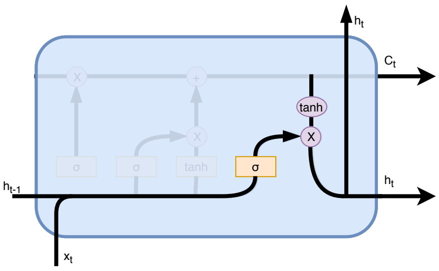


Figure 2.13: Module present in an LSTM- output gate layer highlighted. Based on [57].

$$\begin{gathered} o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h_t = o_t \times tanh(C_t) \end{gathered} \tag{2.28}$$

The LSTM finds its origins in the late 90s, proposed by [32]. In [48], both RNNs and LSTMs are explored. The RNNin use has two layers: a hidden one containing recurrent nodes; an output one containing one or more linear nodes. Parametrization of this structure did not benefit from k-fold validation, as the computational requirements were too high.

As seen in the RNN modeling, the LSTM structure contains both a hidden and an output layer, without benefiting from validation as the computational complexity remains, as expected. For one-step-ahead forecasts, the RNN manages to perform reasonably, while the LSTM presents the third to worst ML method performance, considering the sMAPE metric. Regarding computational complexity, while MLP presents one of the best perfomances for ML methods, both RNN and

LSTM are substantially faster in their result output. Another interesting result is the case of LSTMs, when compared to simpler NNs like RNN and MLP, reporting better model fitting (MSE normalized by the mean value of the time series being examined) but worse forecasting accuracy.

At Uber, the resurgence of LSTMs motivated [40] to propose a new LSTM-based architecture that outperforms the current state of the art event forecasting methods on Uber data and has an acceptable generalization capability to the public M3-Competition dataset. First, a strategy for uncertainty computation(both model and forecasting uncertainty) is presented, followed by a scalable neural network architecture for time series forecasting. From experience, the authors conclude that neural network models are most appropriate when the data size, data length and correlation among time series is high.

Not long after, one of the previous work's main contributors, Smyl, proposed a solution in the M4-Competition which ended up as the top resulting implementation [47]. The hybrid combination consists of an exponentially smoothed RNN. This method was particularly interesting because it incorporated information from both individual time series and the entire dataset, exploiting data in a hierarchical way.

### 2.1.3 Remarks

Regarding ML models, there is a clear advantage when comparing the presented models. In [48] this is especially evident, as all shown and discussed models are built and tested on the same data set. When considering forecasting through machine learning, one should look first for MLP, BNN and SVR. The use cases for LSTM architectures also present interesting results and should be worth investigating in the context of the project associated with this document.

## 2.2 Anomaly Detection

**Anomaly Detection** (also often called outlier or novelty detection) is "*an important data analysis task that detects anomalous or abnormal data from a given dataset*" [4]. Outliers are important because they indicate rare events that usually prompt critical actions to address in a variety of domains. Recently, the number of computer systems (desktop devices, such as smart TVs and computers, and mobile devices, much like laptops and smartphones) has come to rise at a fast rate, given the technological coverage observed within the last ten years. Naturally, the growth of networks ensued, promoting a rising concern regarding security [50]. In this scenario, it is expected that unusual traffic in the networks could mean that a device is being exploited. Another example is concerned with credit card fraud, where anomalies in a credit card report could indicate fraudulent activities associated with it. In the case of an EEG scan, anomalies in the frequencies could indicate a seizure and/or its pre/post conditions. An MRI image with visual anomalies can indicate the presence of a tumor.

Anomalies can be extrapolated from a wide range of areas. With this in mind, [5] categorize them in the following groups:

**Point Anomaly:** When a particular data point deviates from the normal pattern seen in the dataset. For example, when a person's normal car fuel usage is 5 liters per day, but for a particular day it becomes five hundred liters, that's a point anomaly.

**Contextual Anomaly:** When a data instance describes anomalously in a particular environment, it is called contextual or conditional anomaly. For example, it is usual for a person to spend more money during Christmas and new year season than for the rest of the year. Although spent money can be high, it may not be anomalous as high expenses are common during these periods. On the other hand, an equally high expenditure during a non-festive season could be deemed a contextual anomaly.

**Collective Anomaly:** When a collection of similar data instances behave erratically with respect to the entire dataset, the group of data instances is called a collective anomaly. In a EEG scan, the existence of low values for a long period indicates an outlying phenomenon translating to an abnormal premature contraction, whereas one low value by itself is not anomalous.

The following subsections present state of the art approaches on anomaly detection. A brief description and application are provided for each of them.

### 2.2.1 Classification-Based

Classification-based techniques rely on experts' knowledge on anomaly features. When an expert provides an attack's details to the detection system, an attack with a known pattern can be detected as soon as it is launched. For Intrusion Detection Systems (IDSs), this is usually deemed the signature-based detection method. This type of system can only detect attacks that have been previously learned by means of a signature. Even if a new attack's signature is introduced into the system, the effects of the first approach cannot be undone. The advantage of classification-based methods lies in their capability to detect novel attacks, if these present an ample enough deviation. However, given the lack of sufficient data to form a normal profile from which to learn, the system is prone to a high false classification rate [4]. In addition, these techniques require an expert. Next, follows a description and use cases of classification-based techniques.

#### Support Vector Machine

As seen in section 2.1.2, Support Vector Machine (SVM) are a special class of machine learning algorithms that perform classification through projection of the data instances in a dimension where separation of classes is achieved through a decision hyperplane. SVMs are also frequent in the domain of anomaly detection.

In [26], a framework on unsupervised intrusion anomaly detection is presented. The algorithms in use by the framework include k-NN as well as One-Class Support Vector Machine (OCSVM). The implemented OCSVM uses the RBF kernel for feature space mapping. SVM is a supervised learning algorithm by nature. In this use case, the data is unlabeled, so the SVM works differently:

- Instead of separating one class from another, it tries to separate the data points from the origin. This is possible through a quadratic program solution that penalizes points close to the origin while simultaneously trying to maximize the distance of the hyperplane from the origin.
- Support vectors also computed differently. The algorithm attempts to find a region where most of the training data lies and labels points in that region as class +1, while points in other regions are labeled as −1.

Overall, results show that for all considered methods, the detection rate was acceptable, considering how the threshold also affects the false positive rate. In any case, one has to account for the difficulty in unsupervised anomaly detection, given that the training data has no label and may not be clean.

In [30], a Host-based Intrusion Detection System (IDS) that monitors accesses to Windows Registry using Registry Anomaly Detection (RAD) is presented. This IDS uses a OCSVM trained on a dataset of normal registry accesses and is compared with a Probabilistic Anomaly Detection (PAD) algorithm. The implemented OCSVM was modified to compute kernel entries dynamically, given memory limitations. A total of three kernels were tested: linear, polynomial and gaussian. This implementation also uses Sequential Minimal Optimization (SMO). Results show that, overall, the linear kernel helped produce the best ROC curve, considering all tested kernels, but the PAD algorithm consistently presented better results than the OCSVM. The authors claim the PAD algorithm's advantage lies in the use of a Dirichlet-based hierarchical prior to estimate probabilities.

In [33], the performance of Robust Support Vector Machines (RSVMs) was compared with conventional SVMs and nearest neighbor classifiers, in separating anomalies from regular usage profile in computer programs. The authors present a new approach based on RSVMs that deals with noise in the dataset and thus exacerbates RSVMs' superiority in generalization capability over noisy data. Moreover, the number of support vectors in use for the RSVM (30 for clean data, 15 for noisy data) is significantly lower than for conventional SVMs (45 for clean data, 40 for noisy data), thus promoting a shorter execution time. Results show promise for RSVM, with this algorithm scoring the highest attack detection rate, while minimizing the false positive rate, in comparison with other algorithms in use, for both clean and noisy data.

### Long Short Term Memory Recurrent Neural Network

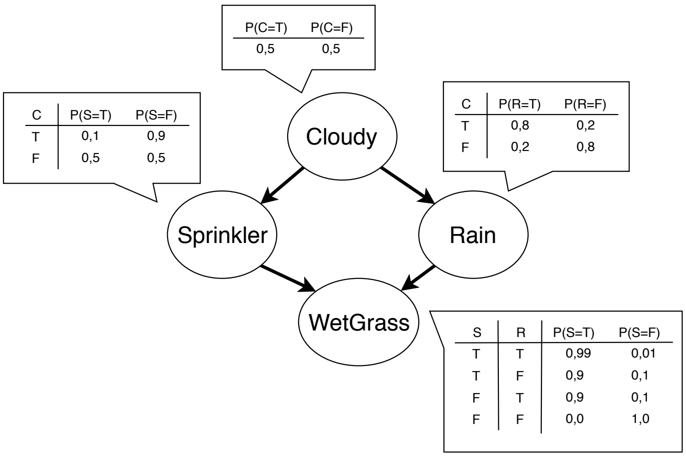
As explained in section 2.1.2, LSTMs are Recurrent Neural Networks (RNNs) that bind memories and time input, and have also been recently explored in regards to anomaly detection.

Audio novelty detection is addressed through the use of a purely unsupervised and novel approach in [49]. The implemented algorithms consist of autoencoders (basic, compressed, denoising and non-linear predictive) built on RNNs (both conventional and bidirectional) with LSTM memory blocks. The performance evaluation is weighted on datasets detailing normal behaviour (characterized by sounds from home environments) and assessed through the F-score metric. Results show that all considered autoencoders show promise, with the lowest scoring implementation (LSTM-Compression Auto Enconder) at 89.1%. The best configuration was the Non-Linear Predictive Bidirectional LSTM Denoising Auto Encoder, scoring at 94.4%.

Another application of Anomaly Detection through the use of LSTM is present in [21], where a predictive model for healthy ECG signals is proposed. This model consists of a deep LSTM neural network where the memory units are stacked in order to feed forward to the following unit. Experiments on this model consist of training the models on the MIT-BIH Arrhythmia Database, which contains both normal and irregular behaviour. Irregular behaviour is categorized into four different types of beats that should be discriminated. It is worth noting that the authors highlight a related objective of the paper "*that the deep LSTM generalize to multiple anomaly* *types (Arrhythmias)*". Results show that the model obtains an F1-score above 93% for all of the irregularities except Atrial Premature Contraction, which scores 85%.

### Bayesian Network

The Bayesian Network (BN), also known as belief network (or Bayes net) is a directed acyclic graph in wich each edge corresponds to a conditional dependency between unique random variables (represented as graph nodes). BNs belong to the family of probabilistic graphical models [13] and use Bayesian inference for probability computations.



| P(C=T) | P(C=F) |
|---|---|
| 0,5 | 0,5 |

| C | P(S=T) | P(S=F) |
|---|---|---|
| T | 0,1 | 0,9 |
| F | 0,5 | 0,5 |

| C | P(R=T) | P(R=F) |
|---|---|---|
| T | 0,8 | 0,2 |
| F | 0,2 | 0,8 |

| S | R | P(S=T) | P(S=F) |
|---|---|---|---|
| T | T | 0,99 | 0,01 |
| T | F | 0,9 | 0,1 |
| F | T | 0,9 | 0,1 |
| F | F | 0,0 | 1,0 |

Figure 2.14: Bayesian network with highlighted joint probability distributions attached to each random variable. Based on [64].

If an edge (A,B) exists in the graph connecting random variables A and B, $P(B|A)$ is a factor in the joint probability distribution, so we must know $P(B|A)$ in order to conduct inference for all values of B and A.

Anomaly based approaches can detect previously unknown attacks, but suffer from difficulty in building a robust model which may result in a large number of false positives. In [38], the authors identify two main reasons for this proportion: simplistic aggregation of model outputs in the decision phase and lack of integration of additional information into the decision process. To deal with these shortcomings, an event classification scheme based on Bayesian Networks is proposed, in the context of intrusion detection on system calls. The integrated models in the net verify string length, character distribution, structure and tokens. The experiment includes a comparison with a naive threshold-based approach, and results show that the BN significantly reduces the false positives throughout the ROC curve. In addition, when all attacks are detected, only half as many false positives in the threshold-based approach are registered.

## 2.2.2 Statistical Anomaly Detection

Statistical theories are also useful when modeling intrusion detection systems. For example, the popular chi-square theory is used for anomaly detection [70]. In this technique, anomaly detection derives from events that are sufficiently deviated from normal. A distance measure based on the chi-square test statistic is developed as

$$X^2 = \sum_{i=1}^{n} \frac{(X_i - E_i)^2}{E_i}, \tag{2.29}$$

where $X_i$ is the observed value, $E_i$ is the expected value and $n$ is the number of variables. $X^2$ has a low value when an observation is near the expected. Following the $\mu \pm 3\sigma$ rule, when an observation, $X^2$, is greater than $\overline{X^2} + 3S_X^2$ it is considered an anomaly.

Next, follows a description and use cases of statistical techniques.

### Principal Component Analysis

Principal Component Analysis (PCA) is a technique most used in the pattern recognition pipeline for feature reduction. The central idea for PCA is to reduce data dimensionality while retaining as much information as possible in the original dataset. This is made by transforming the data to a new set of variables, principal components(PCs), which possess the following properties, according to [34]:

- uncorrelated;
- orthogonal;
- their variances are sorted from highest to lowest or their total variance is equal to the variance in the original data.

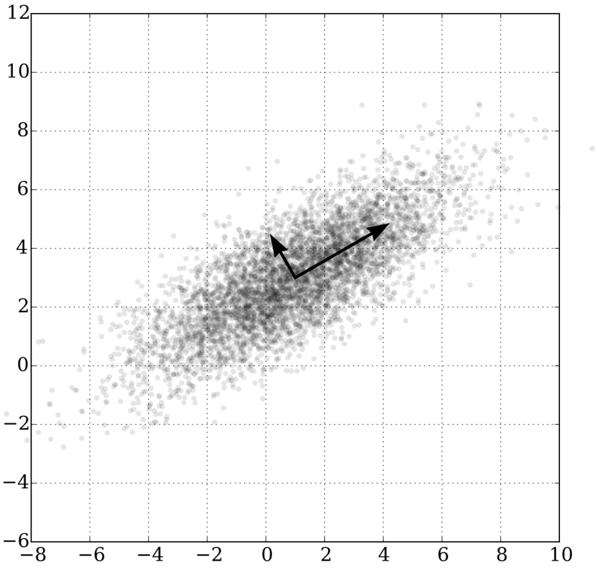


Figure 2.15: PCA applied in dataset. The first 2 PCs' directions are displayed.

To better understand how the PCs are obtained, one should look for the definition of key terms in PCA. The **variance** is a measure of variability that gives information on how far the data is spread. It's value is given by the squared deviation from the mean.

$$Var(X) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n} \tag{2.30}$$

**Covariance** is a measure of the joint variability of two random variables. It's value is computed as shown below.

$$Cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n} \tag{2.31}$$

Assuming a matrix $A$, almost all vectors change direction, when multiplied by $A$. An **eigenvector** is in the same direction as $Ax$. Multiply an eigenvector by $A$, and the vector $Ax$ is a number $\lambda$ times the original $x$. The number $\lambda$ is an **eigenvalue** of A [66].

In PCA, the first step is to compute the covariance matrix of the data points, from which to extract the eigenvectors. At this point, the eigenvectors are sorted in descending order, by their eigenvalues. Depending on the number of PCs in use, the first $k$ eigenvectors will be the new $k$ dimensions. Finally, the original $n$ dimensional data is transformed into $k$ dimensions.

Many statistical based intrusion detection methods assume a normal distribution of the data or resort to the central limit theorem, requiring the number of features in the data to be greater than 30. In this case, it is usual for the data on these problems to present a high dimensionality. While PCA has found its use in feature reduction, it came to find its use in anomaly detection, in [62], where a novel scheme that uses robust PCA for intrusion detection in unsupervised data is proposed. Robustness on this approach is attained through multivariate trimming on the correlation matrix of the training data (preferably free of any outlier data points to begin with).

Adding to the uniqueness of the approach, the authors implement two functions of principal component scores, one for the major and another for the minor components. The second one is present to assist classify observations that do not conform to the normal correlation structure.

The number of components in use is determined by the variation in the data that is accounted for by the PCs. In the experiments, for one, a comparison on the number of considered PCs is carried out. Next, the authors compare the best configuration of the PCC to the Canberra metric, density based local outliers (LOF) and k-NN, with $k = 1$ and $k = 5$, through the KDD'99 dataset. Evaluation of the approaches is obtained through ROC curves plotted by variation of the false alarm rate threshold.

For the comparisons drawn out between different setups of the PCC (regarding variation accounting by the PCs), the PCC based on the major components that can explain 50% of the variation is the best, for a low false alarm rate, but also presents adequate results for higher rates.

When compared with other classifiers, the PCC stands out, with a near 99% consistent detection rate on all false alarm ratings, followed by 1-NN, but by a large margin.

### Mixture Model

It is usual to refer to probabilistic models which take the form of simple distributions (gaussian, for example) in order to better analyze a data set. However, the data in use is not always simple. For example, it can be multimodal. That is, it contains several modes (regions of high probability mass) in between. Suppose we've collected temperature data from the shore and a mountainous region, but didn't label the readings for the region in place. Plotting the collected data might show the distribution has at least two modes, as seen in figure 2.16.

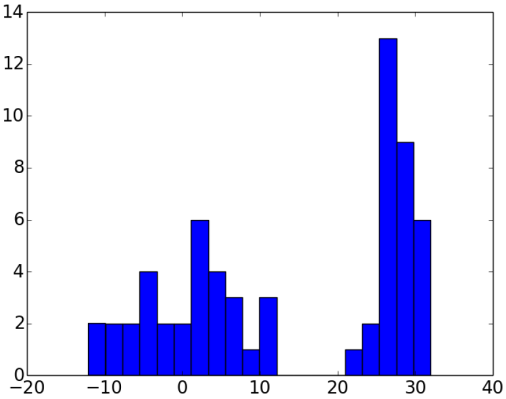


Figure 2.16: Histogram of daily high temperatures in °C.

In this case, one might model the data in terms of a mixture of components, where each component has a simple parametric form (such as a Gaussian). In other words, it is assumed each data point belongs to one of the components, and we try to infer to infer the distribution for each component. The **mixture model** is formally defined in terms of latent variables, which are never observed, usually denoted $z$. Variables which are always, or even sometimes, observed, are called observables

[29]. In the previous example, the region is the latent variable while the the temperature is the observable. In mixture models, the latent variable corresponds to the mixture component, and it takes values in a discrete set. In general, a mixture model assumes the data is generated through the following process:

1. Sample $z$

2. Sample the observables $x$ from a distribution dependent on $z$

, i.e.

$$p(z, x) = p(z)p(x|z) \tag{2.32}$$

, with $p(z)$ being a multinomial distribution, $p(x|z)$ can take a variety of parametric forms, but for the example at hand, we'll assume it's a gaussian distribution. Thus, this model is dubbed a mixture of Gaussians, as seen in figure 2.17.

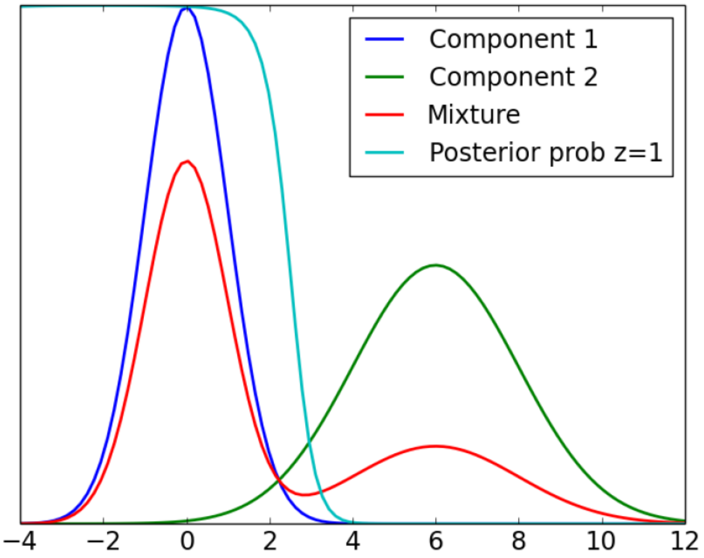


Figure 2.17: Example of a mixture of Gaussians model.

In [25], a mixture model is used for extrapolation on anomaly presence in system call data. Their approach for anomaly detection uses machine learning techniques to estimate a probability distribution and then applies a statistical test in order to detect anomalies. The data is modeled through the following generative distribution.

$$D = (1 - \lambda)M + \lambda A \tag{2.33}$$

, with $M$ being a structured probability distribution estimated over the data through a machine learning technique. $A$ is a distribution modeled on anomalous elements. Detection of the anomalies is attained by computation of the likelihood for each distribution. Performance comparison of their method is carried out in relation to the approaches `stide` and `t-stide`, based on their results in [68]. Results are presented in ROC curves and show that the presented method outperforms the baseline methods, when trained over noisy data, and performs comparably when trained with noisy data and compared with the baseline methods trained over clean data (as in, there are no anomaly data instances). It is important to note that the presented approach makes relevant assumptions regarding the data: normal data can effectively be modeled using the probability distribution; anomalous instances are sufficiently different from normal ones, in order to be correctly detected; the number of anomalies is considerably small compared to the number of normal instances (comparisons are drawn out only over program traces where the percentage of intrusions is less than 5%).

### 2.2.3 Information Theory

**Information theoretic** methods compute information measures of a dataset. These methods assume that anomalies significantly alter the information content of the otherwise "normal" dataset. Usually, metrics are calculated using the whole dataset and then the sequences of points whose elimination induces the biggest change in the metric are labeled anomalous. In [42], information theory measures (entropy, conditional entropy, relative conditional entropy and information gain and cost) are used to characterize several case studies in the domain of internet security.

**Entropy** measures the uncertainty of a collection of data items. For a dataset X where each data item belongs to a class $x \in Cx$, the entropy of $X$ relative to this $|C_X|$-wise classification is defined as

$$H(X) = \sum_{x \in C_X} P(x) log \frac{1}{P(x)}, \tag{2.34}$$

where $P(x)$ is the probability of $x$ in $X$.

Usually, in Information Theory, entropy is regarded as specifying the number of bits required to encode the classification of a data item. The value is smaller when the class distribution is skewer. For example, if all data items belong to the same class, then entropy is 0, because there is only one possible outcome out of the classification.

In anomaly detection, entropy can be used to measure the regularity of the audit data [42]. Each unique record in an audit dataset represents a class. The smaller the entropy, the fewer the number of distinct records. Thus, the audit dataset is more regular.

The **conditional entropy** of $X$ given $Y$ is the entropy of the probability distribution $P(x|y)$, such that,

$$H(X|Y) = \sum_{x,y \in C_X, C_Y} P(x,y) log \frac{1}{P(x|y)}, \tag{2.35}$$

where $P(x,y)$ is the joint probability of $x$ and $y$ and $P(x|y)$ is the conditional probability of $x$ given $y$.

The **relative entropy** between two probability distributions $p(x)$ and $q(x)$ that are defined over the same $x \in C_X$ is

$$relEntropy(p|q) = \sum_{x \in C_X} p(x) log \frac{p(x)}{q(x)} \tag{2.36}$$

The relative conditional entropy between two probability distributions $p(x|y)$ and $q(x|y)$ that are defined over the same $x \in C_X$ and $y \in C_Y$ is

$$relCondEntropy = \sum_{x,y \in C_X, C_Y} p(x,y) log \frac{p(x|y)}{q(x|y)} \tag{2.37}$$

Based on the previously presented information measures, appropriate anomaly detection models can be built, as shown in [42].

### 2.2.4 Clustering-Based

Clustering entails grouping similar data points within a small area (a cluster), while those that appear different are either in another cluster of similar points, or simply do not belong to any formed cluster [4]. This approach is primarily an unsupervised technique, although semi-supervised clustering has been explored [11]. Although there are different types of clustering techniques, the focus in this documents befalls on regular and co-clustering. There are three key assumptions to make when using clustering algorithms for anomaly detection which are presented below.

**Assumption 1:** normal data instances belong to a cluster, while anomalies belong to any other cluster they form.

**Assumption 2:** normal data instances are close to their cluster centroid. Anomalies are far away from any normal cluster centroid.

**Assumption 3:** normal data instances belong to large and dense clusters, while anomalies belong to small and sparse clusters.

### Regular Clustering

**Regular clustering** techniques cluster the data considering the rows of the dataset. One popular example of this variant is k-means clustering [53].

In k-means, the goal is to find $k$ groups in the data. Iteratively, the algorithm assigns each data point to one of the $k$ groups based on the features that are provided. In the end, the algorithm outputs the centroids of the $k$ clusters, which can be used to label new data and labels for the training data (whose points are assigned to their respective cluster).

Once the clusters are built, new data points are subject to the following assumptions.

- An instance is classified as normal, if it is closer to the normal cluster centroid than to any anomalous one. The reverse holds for an anomalous data instance;
- If the distance between the data instance and any normal cluster centroid is larger than a pre-defined threshold, the instance is deemed anomalous;

As seen for SVMs, in [26], a framework on unsupervised intrusion anomaly detection is presented. Other than the OCSVM, both k-NN and a Cluster-based estimation algorithm are experimented with. The cluster-based estimation's goal is to compute how many points are near each point in the feature space. The distance threshold is defined manually. Complexity on this approach is high ($O(n^2)$), given the pairwise distance measures between all points. To overcome this, the authors perform fixed-width clustering on the entire training data set. For each new point, if it falls outside of the fixed-width of the cluster, it becomes the new centroid of a new cluster. A similar problem is faced and dealt with in the deployment of k-NN. The authors employ canopy clustering [51] in order to compute the k-nearest points to a given point, which significantly reduces the algorithm's execution time while promoting real-time application in IDS systems. As previously shown, results for all methods present acceptable results, making the use of clustering algorithms a promising choice in anomaly detection.

In [53], the first part of the paper focuses on Network Data Mining (NDM), while in the second part, a flow-based anomaly detection scheme based on k-means clustering is proposed. The NDM approach in use deploys k-means in order to separate time intervals with normal and abnormal traffic in the training dataset. The resulting centroids are used for anomaly detection in new unseen data.

It should be noted that the authors employ k-means not only for classification of the data points, but also for outlier detection. A unseen data point may be classified as anomalous if the distance to the abnormal centroid is smaller than the distance to the normal centroid. It is an outlier, and, therefore, an anomaly, if the distance from the normal centroid exceeds a defined threshold. Experimental results on this model are based on both data generated from a local testbed, and `tcpdump` traces from a real network. The testbed experiments enables an assertion on the basic anomaly detection capabilities of the proposed model. With the real traffic traces, an analysis on the UDP traffic revealed a short burst (of 15 to 20 seconds on the original time scale) which the authors account to as being the result of the client program of a game updating its list of available game servers. A more sophisticated performance metric has not been provided.

### Co-Clustering

Co-clustering is similar to regular clustering, but instead of processing the dataset for rows, it processes rows and columns simultaneously. It can produce a set of $c$ column clusters of the original columns (C) and a set of $r$ row clusters of the original row instances ($R$). Co-clustering defines and optimizes a clustering criterion to find subsets of rows and columns of a data matrix [4].

Advantages of this variation over Regular clustering include:

- Simultaneous row and column grouping provides a more compact representation of the data while preserving information;
- Co-clustering can be considered a dimensionality reduction technique and is suitable for creating new features;
- Provides significant reduction in computational complexity. Traditional k-means has $O(mnk)$, where m is the number of rows, n is the number of columns and k is the number of clusters. In co-clustering, computational complexity is $O(mkl+nkl)$, where l is the number of column clusters. It is clear that $O(mnk) > O(mkl + nkl)$.

In [3], an adaptation of co-clustering is developed in order to analyze network traffic patterns, in regards to DoS attacks as a collective anomaly. The authors extend co-clustering by enabling the algorithm to handle mixed attribute data instances, which translates to an increment in detection accuracy. They also claim to be the first to explore collective anomaly detection, only drawing comparison of the results with [58], who used co-clustering for network anomaly detection considering only cluster purity as a performance measure, and only seven numerical attributes for the clustering algorithm. In any case, the former manages to outperform the latter's implementation. The obtained F-score reaches 96%.

# Chapter 3

# Resources and Tools

This chapter details all resources and tools to consider for the approach proposed in this report. The resources constitute a list of datasets from which to input into the models, while the tools include model implementation, data processing and other Machine Learning (ML) pipeline modules.

All resources and tools must be licensed under open-source licenses, such as: BSD License[1], Apache License[2], MIT License[3] and GPL[4].

## 3.1 Resources

In this section, some useful resources for the framework are presented. The datasets provided by the resources that follow provide a good benchmark to consider when assessing the selected model's quality and performance, given that some of them even provide solutions in the forecasting front. These resources stem from renowned competitions and publicly available repositories. While the internship takes place at Novabase Business Solutions, it is worth mentioning a few considerations regarding the use of this data. The sector in which this internship unfolds is the Financial Solutions Industry, which specializes in solutions for banks, insurance agencies and capital markets. Given the data sensitivity, regarding privacy, provided in these areas, the usage of this data won't be considered as a viable solution for resources.

### 3.1.1 Makridakis Competition (M-Competition)

The M-Competitions represent one of the most worldwide popular forecasting time series competitions. The datasets provided for these competitions usually contain a substantial number of time series, of different natures. The most recent datasets (stemming from the M-4 competition) are worth mentioning, as the resulting methods are publicly available[5] and can be used as benchmarks for the validation and evaluation of the approach proposed in this report. The M4 dataset consists of 100,000 time series coming mainly from Economic, Finance, Demographic and Industry areas, while also including Transportation, Natural Resources, Environment and other domains. The minimum number of observations for each time period is: 13 for yearly; 16 for quarterly; 42 for monthly; 80 for weekly; 93 for daily; 700 for hourly.

[1] https://opensource.org/licenses/BSD-3-Clause

[2] https://www.apache.org/licenses/LICENSE-2.0

[3] https://opensource.org/licenses/MIT

[4] https://www.gnu.org/licenses/gpl-3.0.html

[5] https://github.com/M4Competition/M4-methods

### 3.1.2 UCI Machine Learning Repository

The University of California Irvine (UCI) Machine Learning Repository started as an FTP archive in 1987, created by David Aha and graduate students at UCI, and retains a collection of databases, domain theories and data generators that serve as benchmarks for machine learning implementations. For the matter at hand, the repository provides both univariate and multivariate time series datasets which may be used for the validation and evaluation of the approach proposed in this report.

## 3.2 Tools

The approaches enumerated in chapter 2 aggregate a few of the state of the art options when considering forecasting and anomaly detection practices applied to time series data. The programming frameworks and tools in use to implement these approaches play an important part, as some present richer development environments and simplified deployment. It is important to consider machine learning libraries that feature good runtime performance, tools support, a large community of developers and a healthy ecosystem of supporting packages. This section provides a few details on these tools.

### 3.2.1 Scikit-learn

Scikit-learn[6] is a Python machine learning library that features algorithms to handle classification, regression, clustering and dimensionality reduction problems. Data pre-processing and model selection modules are also provided. It is designed to interoperate with the Python numerical and scientific libraries NumPy[7] and SciPy[8]. Scikit-learn is licensed under the New Berkeley Software Distribution (BSD) License.

### 3.2.2 TensorFlow

TensorFlow[9] is an open-source library, created by Google Brain, for numerical computation and large-scale machine learning. It bundles together a large selection of machine learning algorithms, mainly divided by deep learning approaches (Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), etc) and everything else (Support Vector Machine (SVM), k-Nearest Neighbours (k-NN), etc). TensorFlow uses Python to provide a front-end API for building applications with the framework, while executing those applications in high performance C++. TensorFlow is licensed under the Apache License 2.0.

### 3.2.3 Weka

Waikato Environment for Knowledge Analysis (WEKA)[10] is a Java library containing machine learning algorithms for data mining tasks, while also being able to handle big data problems and perform deep learning. It includes tools for data processing, classification, regression, clustering and data visualization. WEKA is licensed under the GNU General Public License.

[6] https://scikit-learn.org/stable/

[7] http://www.numpy.org/

[8] https://www.scipy.org/

[9] https://www.tensorflow.org/

[10] https://www.cs.waikato.ac.nz/ml/weka/

### 3.2.4 PyTorch

PyTorch[11] is a Python based library built to provide flexibility as a deep learning development platform. It has many similarities to TensorFlow, such as hardware-accelerated components and a highly interactive development model that allows for design-as-you-go work. PyTorch is BSD-style licensed.

### 3.2.5 CNTK

The Microsoft Cognitive Toolkit (CNTK)[12] is a deep learning toolkit that describes neural networks as a series of computational steps via a directed graph. CNTK allows users to easily realize and combine popular model types such as CNNs and recurrent networks (RNNs and Long Short Term Memorys (LSTMs)). It handles many neural network jobs fast, and has a broad set of Application Programming Interfaces (APIs) (Python, C++, C# and Java). CNTK is licensed under the Massachusetts Institute of Technology (MIT) License.

### 3.2.6 Apache MXNet

Apache MXNet[13] is a deep learning framework that supports state of the art deep learning models (CNNs and LSTMs) and is widely used by Amazon Web Services (AWS). Given its use of a distributed parameter server (based on research at Carnegie Mellon University, Baidu and Google), it can achieve almost linear scale with multiple Graphics Processing Units (GPUs) or Central Processing Units (CPUs). MXNet also supports a broad range of APIs (Python, C++, Scala, R, JavaScript, Julia, Perl and Go). MXNet is licensed under the Apache License 2.0.

Table 3.1 enumerates the previously described tools.

| Tool | Initial Release | Written in | Interface | Software license |
|---|---|---|---|---|
| Scikit-learn | 2007 | Python | Python | New BSD |
| TensorFlow | 2015 | Python, C++, | Python, C/C++, Java, Go, JavaScript, R, Julia, Swift | Apache 2.0 |
| Weka 3 | 1997 | Java | Java | GNU General Public License |
| PyTorch | 2016 | Python, C, | Python | BSD |
| CNTK | 2016 | C++ | Python, C++, BrainScript | MIT license |
| Apache MXNet | 2015 | C++ core library | C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl | Apache 2.0 |

Table 3.1: Tools comparison table.

[11] https://pytorch.org/

[12] https://www.microsoft.com/en-us/cognitive-toolkit/

[13] https://mxnet.apache.org/

# Chapter 4

# Competitor Analysis

Time series span throughout a wide range of areas, from financial information, to meteorological readings, to IoT data. It is only natural that to better understand trendings, anomaly occurrences and forecasting, one has to perform analysis on these data structures. This procedure can be facilitated with the help of frameworks which provide the analyst with tools to perform forecasting and anomaly detection. The following chapter addresses competitors to time series analysis tools and how they're related, or not, to each other. Both forecasting and anomaly detection applications are considered. Finally, comparisons between each competitor are summarized.

## 4.1 Forecasting

### 4.1.1 NCSS 12

NCSS is a company created in 1981 by Jerry L. Hintze that specializes in statistical software applied to medical and business research, quality control, academia and teaching/learning. The latest iteration on their data analysis product, NCSS 12[1], has been released in November, 2018 and aggregates a rich library on statistic procedures (including descriptive statistics, parametric and nonparametric tests, forecasting, multivariate analysis, time series and model fitting). Operations on this platform appear intuitive and clear: one may import and format the initial data through a table structure; open a procedure which they want to apply; and view the output on another window. Major data extensions (such as MATLAB, SPSS, text, R, Excel and more) are also supported, making this suite's data management flexible. Procedure selection is also presented in a straight-forward manner, through a dropdown menu, search bar and even a category tree. Output handling is an easy task, as direct interaction with the plots highlights each one of them, should the user want to isolate one in a separate file.

### 4.1.2 RapidMiner Studio

RapidMiner Studio[2] is a visual data science workflow designer accelerating the prototyping and validation of models. Data access and management is flexible, as the application supports both traditional structured data (time series) and unstructured data (text, images, and media). Information extraction from these types and transformation from unstructured to structured data is also attainable. Data exploration is also delivered, with the system being able to compute descriptive statistics and to render graphical information. Data preparation is available through the use of processes such as sampling, transformations, partitioning, binning, weighting and selection. The selection of offered models is rich, including both supervised and unsupervised learning. Both classical and machine learning (including deep learning) approaches are available. Validation on

[1] `https://www.ncss.com/software/ncss/`

[2] `https://rapidminer.com/products/studio/feature-list/`

the previous models can be performed on a set of provided performance measures. The characteristic the company advertises for standing out from the competition is the abstraction drawn from actual programming of the previously mentioned data analysis pipeline. The company boasts a set of utility process control operations that enables process construction that behave like a program over which one can iterate tasks, branch flows and system resource calls.

### 4.1.3 Autobox

Automatic Forecasting Systems Inc. was founded in 1975 with the goal of presenting the first marketable forecasting toolset. At AFS, two central beliefs are presented: the Box-Jenkins approach provides the proper framework for forecasting; procedures that consist of methods applied in a consistent way are subject to automation. Thus, they developed a forecasting engine that applies the modeling philosophy of Box-Jenkins to time series and builds models automatically. The result is a package that provides a set of tools for both beginners and experts, Autobox[3]. Autobox won the "Best Dedicated Forecasting Program" in [8]. Major features include adaptation of the model to the data through causal variable discovery by patterns from historical forecast errors and outliers identified by the Autobox engine. Standard features include outlier classification in four areas and forecasting diagnostics through ARIMA and Transfer Function models.

### 4.1.4 Alteryx Designer

Formerly known as SRC, LLC, founded in 1997, Alteryx is a software company specialized in data science and analytics. Their product, Alteryx Designer[4], streamlines the process of data preparation, blending and analytics by delivering a repeatable workflow for self-service data analytics, leading to deeper insights within an acceptable time window, all while maintaining an intuitive user interface. Alteryx promises data preparation and blending capabilities that are up to a hundred times faster than traditional approaches. Data access is done through data warehouses and cloud applications as well as local sources, such as spreadsheets. With the use of Alteryx Visualytics, changes in the preparation, blending and modelling stages can be interpreted in a graphical manner. Prediction analysis makes use of R-based analytics while removing the coding layer from the end-user, by option. Given the recent popularity of Big Data, IoT, mobile consumer and social media data, spatial analytics are embedded in this software. Finally, insights sharing is simplified in the form of graphics exportation to popular reporting formats (PNG, HTML, PDF and Microsoft Office extensions).

### 4.1.5 Microsoft Azure Data Explorer

Microsoft's Azure Data Explorer[5] was published by Microsoft and presents a fast and scalable data exploration service for log and telemetry data. It is a good option for analyzing large volumes of data stemming from websites, applications and IoT devices. This tool mostly focuses on quickly identifying trends, patterns or anomalies in all data types inclusive of structured, semi-structured and unstructured data, through a powerful query language optimized for ad-hoc data exploration.

### 4.1.6 Amazon Forecast

Amazon stands out globally for being the most valuable company in world[6]. This company focuses in diverse sectors, such as e-commerce, cloud computing and artificial intelligence. It comes as no surprise that they hold a product capable of Machine Learning (ML) induced forecasting.

[3] https://autobox.com/cms/index.php/products/autobox

[4] https://www.alteryx.com/products/alteryx-platform/alteryx-designer

[5] https://docs.microsoft.com/en-us/azure/data-explorer/time-series-analysis

[6] https://edition.cnn.com/2019/01/08/investing/amazon-most-valuable-company-microsoft-google-apple/index.html

Amazon Forecast[7] is an accurate time-series forecasting service that depends on ML to deliver accurate forecasts. Accuracy on the forecasts is promoted on time series data as well as additional variables that have influence. Machine learning expertise is also not an issue as this service includes AutoML capabilities. Once the data is uploaded, Amazon Forecast processes the data, selects the appropriate algorithms, trains a model, provides registered metrics and generates forecasts. If conformant with Amazon Forecast requirements, consumers can provide additional algorithms to the platform and even customize existing ones. Finally, forecasts can be visualized in the console, exported in batch in `CSV` format or be retrieved using the Amazon Forecast API.

### 4.1.7 Forecasting Competitors Analysis

It is clear that, while Autobox has been one of the first publicly available products, it is severely outperformed by its competitors. NCSS, RapidMiner and Alteryx Designer present the software suites with the richest set of features, while Microsoft's Azure Data Explorer is not entirely a consumer ready product, but a powerful service to be used majorly by experts in the domain of computer science and statistics, given it's optimized query language. Amazon Forecast takes a different approach, presenting a cloud solution. However, its AutoML capabilities render it an appealing choice, for consumers without ML expertise. In table 4.1 lies a summary on forecasting competitors' features.

Table 4.1: Forecasting competitor feature comparison.

| | | Competitors | | | | | |
|---|---|---|---|---|---|---|---|
| | | NCSS | RapidMiner | Autobox | Alteryx Designer | MS Azure Data Explorer | Amazon Forecast |
| Features | GUI | • | • | • | • | | |
| | Semi-structured data | • | • | | • | • | |
| | Unstructured data | • | • | | • | • | |
| | Descriptive Statistics | • | • | • | • | • | • |
| | Data Visualization | • | • | • | • | • | • |
| | Data Processing | • | • | • | • | • | • |
| | Supervised Learning | • | • | | • | | • |
| | Unsupervised Learning | • | • | | • | | • |
| | Export format flexibility | • | • | | • | | • |

[7] `https://aws.amazon.com/forecast`

## 4.2 Anomaly Detection

### 4.2.1 Microsoft Azure Machine Learning Studio

Another ambitious service provided by Microsoft is the Azure Machine Learning Studio[8]. This collaborative visual development environment helps build, test and deploy predictive analytics solutions in the cloud. The Machine Learning Studio contains a module specifically designed for time series data processing, namely in anomaly detection, but one can also take advantage of the R and Python scripts in use to manually manipulate the data, and apply it to the set of given classification and regression models. Classification modules range from binary to multiclass and include popular algorithms such as Decision Forest, Decision Jungle, Neural Network, Support Vector Machine (SVM), Decision Tree and classification models ensembles. Regarding Regression models, the platform includes bayesian linear, boosted decision tree, fast forest quantile, neural network, ordinal and poisson regressions.

### 4.2.2 Elastic Stack

Elastic Stack is a collection of open-source software that facilitates log search, analysis and visualization, commonly referred to as centralized logging. Centralized logging is useful when trying to identify problems, in servers and applications, and issues that span multiple servers by correlating their logs during a specific time window. Elastic Stack has four main modules:

**Kibana:** gives shape to data and is the extensible user interface for configuring and managing Elastic Stack;

**Elasticsearch:** distributed, JSON-based search and analytics engine;

**Beats:** platform for lightweight data shippers that send data from edge machines to Logstash and Elasticsearch;

**Logstash:** data processing component of Elastic Stack which sends incoming data to Elasticsearch.

Kibana[9] is a module that stands out for enabling time series analysis (including queries, transformations and visualizations) on its dedicated UI. Anomaly Detection is present and applied with the use of unsupervised ML features. The forecasting feature is also boasted, through ML techniques, but no clear list of implemented algorithms is provided, other than a mixture of techniques which include clustering, various types of time series decomposition, bayesian distribution modeling and correlation analysis.

### 4.2.3 Sophie AIOps (Loom Systems)

Loom Systems is a company founded in 2015 which created an ML powered log analysis software that provides real time solutions for IT incidents, Sophie AIOps[10]. Sophie parses structured and semi structured data, classifies them and then applies the appropriate measurement method for each property. Anomaly detection is supported by ML algorithms and optimized with dynamic thresholds based on data signature in real-time. Sophie applies cognitive reasoning (handling conceptual/symbolic data) to detect cross-environment and cross-application issues in order to analyse the root cause for the identified issues.

[8] `https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/time-series`

[9] `https://www.elastic.co/products/kibana`

[10] `https://www.loomsystems.com/product-overview`

### 4.2.4 Anomaly Detection Competitors Analysis

Regarding Anomaly Detection, Microsoft's Azure Machine Learning Studio presents the software with the richest libraries in terms of algorithms in use, and even though information on Elastic Stack is lacking, it still is the only open-source solution at hand. Sophie AIOps heavily markets its cross-correlation capabilities in order to better understand identified anomalies.In table 4.2 anomaly detection competitors' features are presented.

Table 4.2: Anomaly detection competitor feature comparison.

| | | **Competitors** | | |
|---|---|---|---|---|
| | | **MS Azure Machine Learning Studio** | **Elastic Stack** | **Sophie AIOps** |
| **Features** | **GUI** | • | • | • |
| | **Semi-structured data** | • | • | • |
| | **Unstructured data** | • | • | |
| | **Descriptive Statistics** | • | • | • |
| | **Data Visualization** | • | • | • |
| | **Data Processing** | • | • | • |
| | **Supervised Learning** | • | | • |
| | **Unsupervised Learning** | • | • | • |
| | **Export format flexibility** | • | | • |

# Chapter 5

# Approach

This chapter presents the proposed approach for development in the context of the internship. First, a list of functional and non-functional requirements is provided, followed by the associated risks, concluding with the architecture of the framework to implement.

## 5.1 Requirements

This section presents the requirements that have been defined for the project. While requirements usually present a detailed description as well as prioritization, this specification is not granular to the point where it would make sense to prioritize. Rather, we opted to define the high level requirements of the project, which are, by definition, the most important and with highest priority.

### 5.1.1 Functional Requirements

The baseline functional requirements collected for the time series analysis framework are present in table 5.1.

Table 5.1: Baseline functional requirements.

| ID | Name | Description | Dependencies |
|---|---|---|---|
| FR.01 | Implement data processing capabilities | The framework should handle data preparation and processing for use in the framework. | - |
| FR.02 | Implement a Forecasting framework | Provide a forecasting interface that manages different forecasting approaches for differently defined time series and selects the most appropriate approach, appropriately parametrized. | FR.01 |
| FR.03 | Implement a Anomaly Detection framework | Integrate an anomaly detection framework that, similarly to the forecasting framework, manages different anomaly detection approaches and outputs the most appropriate, given the input dataset's features. | FR.01 |
| FR.04 | Implement a REST API to support the framework operations | The implemented framework should be accessible through REST calls in a wrapping API. | FR.01, FR.02, FR.03 |

**Implement data processing capabilities** The framework should be able to provide and apply data processing techniques to clean and prepare the input dataset for processing within the framework.

**Implement a Forecasting framework** The framework should provide a forecasting framework that handles both baseline and optimized forecasting approaches, for both single and multi value time series, being able to select the most appropriate approach, based on the features of the given time series.

**Implement a Anomaly Detection framework** The framework should provide a anomaly detection framework that handles both baseline and optimized anomaly detection approaches, for both single and multi value time series, being able to select the most appropriate approach based on the features of the given time series.

**Implement a REST API to support the framework operations** The framework should be accessible through REST API that provides all the operations needed to use the previously mentioned forecasting and anomaly detection frameworks.

### Implement data processing capabilities (FR.01)

Some time series analysis methods require the data to incorporate, or not, certain features. As discussed in the introduction to chapter 2, time series are usually categorized in relation to seasonality, trend and outliers. Thus, this requirement states the use of techniques that identify and, if necessary, remove components such as seasonality or trend (both of which are necessary to have in order to correctly apply Holt-Winters Exponential Smoothing (HW), for example). This requirement also contemplates the handling of other time series features, such as missing values and noise existence.

### Implement a Forecasting framework (FR.02)

In table 5.2 the functional requirements for the implementation of the forecasting framework are presented.

Table 5.2: Requirements for Implementation of a Forecasting framework.

| ID | Name | Description | Dependencies |
|---|---|---|---|
| FR.02.01 | Implement support for forecasting in time series | The framework should implement base and optimized approaches for forecasting in single and multi value time series. | - |
| FR.02.02 | Implement Autonomous Approach Selection | Implement support to automatize the decision of the best forecasting approach, including model parametrization. | FR.02.01 |

**Implement support for forecasting in time series** The forecasting framework should provide different forecasting methods, including a baseline method which is the default choice for most situations is available. However, the framework should also supply other, more optimized approaches, which adapt to time series containing different features, such as the previously mentioned seasonality, trend, noise and others. Both single and multi value time series approaches should be considered.

**Implement Autonomous Approach Selection** The forecasting framework should be able to autonomously select the most appropriate approach, considering the time series' features, as well as select the best parameters to handle the given time series.

### Implement a Anomaly Detection framework (FR.03)

In table 5.3 the functional requirements for the implementation of a forecasting framework are shown.

Table 5.3: Requirements for Implementation of a Anomaly Detection framework.

| ID | Name | Description | Dependencies |
|---|---|---|---|
| FR.03.01 | Implement support for anomaly detection in time series | The framework should implement base and optimized approaches for anomaly detection for both single and multi value time series. | - |
| FR.03.02 | Implement Autonomous Approach Selection | Implement support to automatize the decision of the best anomaly detection approach, including its parametrization. | FR.03.01 |

**Implement support for anomaly detection in time series** Similarly to the forecasting framework, the anomaly detection framework should provide methods for anomaly detection. There should be a baseline method which is the default choice for most situations, and other, more optimized approaches, which adapt to time series containing different features, such as the previously mentioned seasonality, trend, noise and others. Both single and multi value time series approaches should be considered.

**Implement Autonomous Approach Selection** The anomaly detection framework should be able to autonomously select the most appropriate approach, considering the time series' features, as well as select the best parameters to handle the given time series.

#### Implement a REST API to support the framework operations (FR.04)

The time series analysis framework results of an aggregation of the forecasting and anomaly detection frameworks, as well as their data processing and model management capabilities. This wrapping framework is a REST API which provides all of the system's operations and enables easy integration within other services.

### 5.1.2 Performance Requirements

It is important that the Approach Selector takes the processing time for each algorithm into account, as some algorithms are more complex then others, thus taking more time to process. The Approach Selector itself should fit in a optimal time frame for selecting the most appropriate approach. Table 5.4 presents the proposed performance requirements.

Table 5.4: Performance requirements.

| ID | Name | Description | Dependencies |
|---|---|---|---|
| PR.01 | Approach processing time | Processing time for each approach should be taken into account by the Approach Selector. | - |
| PR.02 | Selection processing time | The implementation of the Approach Selector should reduce the time for the selection as much as possible. | - |

**Approach processing time** This document explores Machine Learning (ML) approaches whose computational complexity ranges greatly. Some approaches have a bigger execution time, which can be a problem. This requirement states that the model's implementation should meet a satisfactory performance threshold in order to be usable. In that sense, a ratio relating quality with complexity of the algorithm should be considered, and manipulable by the user.

**Selection processing time** Each of the selectors have at their disposal a wide selection of algorithms to deal with different problems in time series analysis. The decision for the most appropriate method to use should impact execution time as little as possible. In that sense, this requirement should provide a threshold to consider for the execution time of the autonomous selector.

### 5.1.3 Technological Requirements

The collected technological requirements stem from chapter 3. In that sense, the following requirements detail both resources, tools and implementation. Table 5.5 presents the technological requirements in place for the project.

Table 5.5: Technological requirements.

| **ID** | **Name** | **Description** | **Dependencies** |
|---|---|---|---|
| TR.01 | Programming Language | The project should be developed in Java or Python. | - |
| TR.02 | Open-source Licenses | Licenses from all used software should allow its use in a commercial environment, without additional charge. Allowed licenses are: BSD, Apache, GPL and MIT. | - |

### 5.1.4 Quality Requirements

It was shown in chapter 2 that, for both forecasting and anomaly detection practices, different approaches, with different parametrization, present different results. However, given that some of the gathered datasets already provide satisfactory solutions, it is of interest to attain a relative threshold of quality. Table 5.6 shows the quality requirements at hand for the project.

Table 5.6: Quality requirements.

| ID | Name | Description | Dependencies |
|---|---|---|---|
| QR.01 | Optimum Threshold | The framework should define a minimum accuracy according to a given benchmark. | - |

**Optimum Threshold** This requirement takes advantage of the already existing solutions, for the datasets in use, to specify a quality threshold given by the metrics they use. This presents an advantage, as the system can be calibrated to produce results with comparable quality to the M4-Competition resulting methods, as well as other implementations provided by developers resorting to the University of California Irvine (UCI) repository.

## 5.2 Risks

### 5.2.1 Performance issues in the Autonomous Approach Selector

Both the forecasting and the anomaly detection autonomous approach selector implementation could have a high computation time.

**Contingency plan:** Reach for a trade-off solution between computational complexity and approach selection quality that meets a proposed threshold.

### 5.2.2 Implementation of the Autonomous Approach Selector

This risk is a generalization of the previous one. The implementation of the Autonomous Approach Selector can be revealed to be more complex than expected, causing a delay on the project development.

**Contingency plan:** Simplify the requirements for the selector, should there be no workaround.

### 5.2.3 Accuracy issues in the approaches

Both forecasting and anomaly detection approaches could deliver subpar quality in their output models.

**Contingency plan:** Refer to the benchmarks in use, in order to either maintain or optimize the integrated models.

### 5.2.4 Unavailability of desired approaches' implementation

There is no implementation of a desired approach that we want to use.

**Contingency plan:** A first approach considers resorting to implementing the approach ourselves. Should this not be possible within the considered time window, the approach might not be considered for the framework.

## 5.3 Architecture

This section presents the general architecture for the proposed approach. Figure 5.1 shows the modules that compose each layer of the architecture.

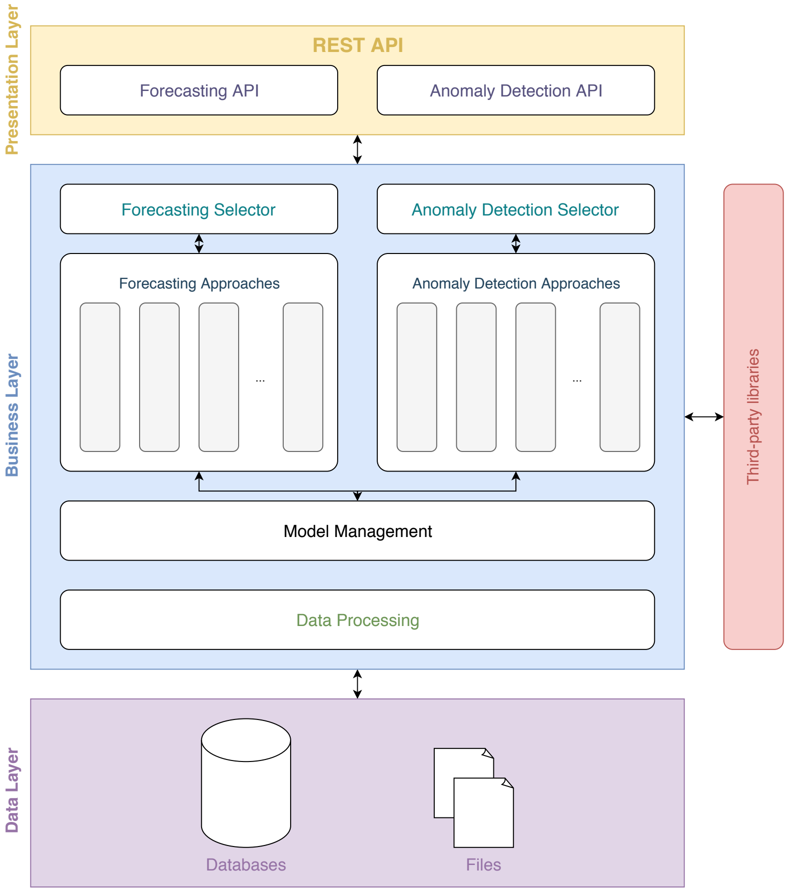


Figure 5.1: General architecture for the framework.

The data layer handles data persistence for all the considered data structures. The business layer contains all functionality logic associated with the forecasting and anomaly detection frameworks, whose approaches' implementations, as well as model management and data processing, derive from the third-party libraries. Finally, the presentation layer provides a REST API to the user through which he can access the time series analysis framework.

The following subsections detail each platform layer.

### 5.3.1 Data Layer

The data layer represents the sources that will be used to process the time series data, as well as the persistence of the data necessary for use of the framework functionalities. The framework should support the processing of time series data in both databases and files.

### 5.3.2 Business Layer

The business layer handles the logic behind the framework. It's main modules are the forecasting and anomaly detection frameworks.

#### Forecasting Selector

The forecasting selector is responsible for managing operations based on data for which the user requests forecasting. It maintains a selection of forecasting approaches which can be appropriately parametrized (specified in the model management component shown in the architecture in figure 5.1) to meet the data features. The selector considers baseline approaches which will be the default choice for most scenarios, for both single and multi value time series, but also provides other, optimized approaches, which should present better results for time series with unusual features, ranging from their time resolution, to noise in the data, to even volume of their instances. This is essentially a forecasting framework to embed in the analysis framework.

#### Anomaly Detection Selector

The anomaly detection selector is similar to the forecasting selector, only it handles and provides anomaly detection approaches. This selector also handles scenarios provided by single and multi valued time series, with support of baseline and optimized methods for each of those types of data.

#### Data Processing

The data processing module applies techniques in order for the data to be correctly ingested by the analysis selectors and to enhance the performance and accuracy of the approaches. The techniques which integrate this component also stem mainly from the provided third-party libraries, and include operations for time series base features (seasonality, trend and outliers).

### 5.3.3 Presentation Layer

The system provides an REST API which allows for easy integration within other services, providing all of the aforementioned functionalities.

# Chapter 6

# Methodology and Planning

This section presents the methodology adopted for this project as well as the initial proposed plan and modifications that have been made during the first semester.

## 6.1 Methodology

The internship takes place at Novabase, and, as such, follows the development methodology in practice at the company. In this case, the methodology in use is Scrum[1]. *Scrum* is an iterative and incremental methodology, used in *Agile Software Development*[2], by small teams, which contains a set of roles and practices that can be adjusted to the environment of each team it integrates. A key principle in *Scrum*, also present in the *Agile Manifesto*, is that requirements change throughout the timeline of a software project. Unlike traditional methodologies, this means that unpredictable changes can be managed easily. This methodology uses an empirical approach and accepts that a problem scope cannot be fully understood, let alone defined, rather focusing on quickly reacting to emergent challenges.

The development process in *Scrum* evolves by iterations, the *Sprints*, which usually last from two to four weeks. Each *Sprint*'s tasks, the *Sprint Backlog*, are determined in the *Sprint Planning Meeting*, chosen from a list of requirements to develop, prioritized, the *Product Backlog*. Requirements defined in the *Sprint Backlog* cannot change during the *Sprint*, which itself must always end at the defined time. Should a requirement not have been fulfilled in a *Sprint*, it goes back to the *Product Backlog*. Progress on a *Sprint* is supported by the use of *Burn Down Charts*, which monitor the time expenses associated with the tasks.

Progress is also discussed in daily meetings, which usually last 15 minutes, called *Daily Scrum Meeting*. In these, each team member presents finished tasks, the tasks to finish at that day and possible impediments they're facing. When a *Sprint* finishes, the team reviews the work and presents it to the *stakeholders* in a *Sprint Review Meeting*. To conclude, a *Sprint Retrospective Meeting* allows the team to reflect on the last *Sprint* and suggest changes where deemed necessary.

The major roles defined in *Scrum* are as follows.

***Product Owner*** Responsible for maximizing the value of the product resulting from the work of the Development Team. Solely the *Product Owner* manages the *Product Backlog* (including expressing product backlog items and ordering items to maximize goals achievement).

***Scrum Master*** Responsible for promoting and supporting *Scrum* by helping every role understand *Scrum* theory, practices, rules and values. The *Scrum Master* also provides services to both the *Product Owner* (finding techniques for effective *Product Backlog* management, fa-

[1] https://ieeexplore.ieee.org/document/854065

[2] https://agilemanifesto.org/principles.html

cilitating scrum events, etc) and the Development Team (coaching the developers, removing impediments to the development progress).

***Team*** Consisting of members who deliver increments of the product at the end of each *Sprint*. The team does not recognize any titles for its members, regardless of the work being performed by each person.

For this project, the team will be composed of Daniel Moura (*Product Owner*), Miguel Oliveira (*Scrum Master*) and Pedro Costa (*Team*). This internships folds out in the context of a solution that has no direct integration with any part of Novabase's solutions, thus, its development rests solely in the aforementioned people. Sprints are projected to have a duration of two weeks.

The Scrum process will be implemented using Jira[3]. The main communication channel is Slack[4]. Project artifacts, which include source code and documentation, which are produced throughout the internship are stored and versioned using repositories hosted on BitBucket[5].

## 6.2 Planning

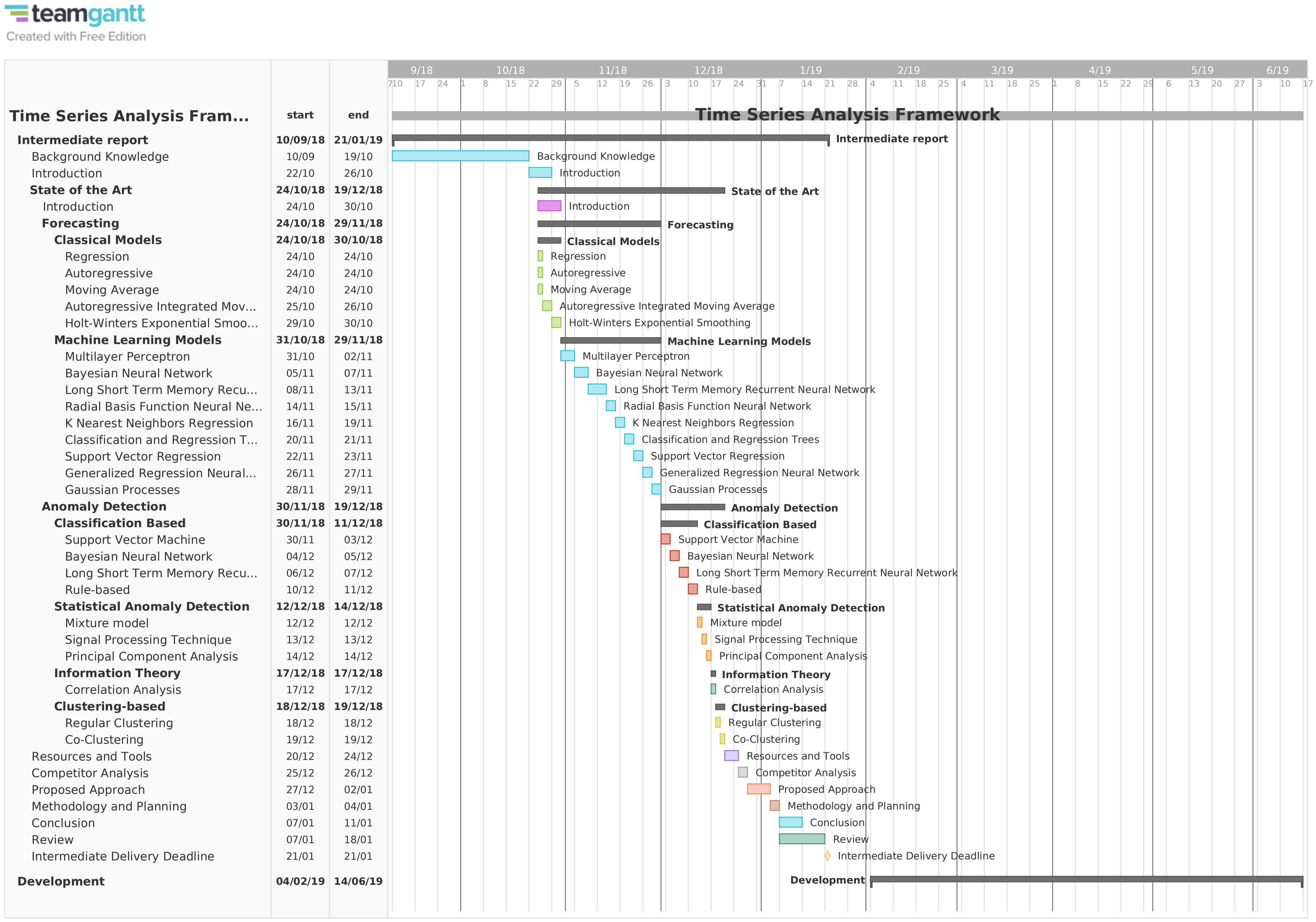
This section provides information on the project planning regarding both the first and second semester. An initial Gantt diagram is present in figure 6.1 and 6.2. However, given how some parts of the work carried out during the first semester exceeded the initial time estimate, figure 6.3 presents the Gantt diagram for the actual time expenses regarding the first semester. Looking closely, one can see that Signal Processing Technique and Correlation Analysis have been removed from the Anomaly Detection state of the art approaches. While they were studied, their weight in the state of the art did not appear significative, especially considering the difficulty in browsing the literature for these methods. In the case of the Information Theory section, 2.2.3, a use case for the introduction was provided, thus this section was kept. However, Anomaly Detection proved an area where its problems are difficult to solve, effectively shifting the duration for this task up to January $4^{th}$, while the remainder of the tasks also got shifted. The last modification to the original schedule was made at Department of Informatics Engineering (DEI), where the delivery date was pushed from January $21^{st}$ to January $23^{rd}$. In any case, the first semester was mainly used for information search on the internship area, regarding state of the art, tools, competitors and finally the proposal of the approach.

The second semester addresses the development of the project, which is divided into *sprints*, in accordance to the methodology in use. The Gantt for the second semester is not as dense, as the tasks to fulfill are not defined beforehand. In order to take advantage of *Scrum*, each sprint deals with the reporting tasks detailed in the previous section.

[3] `https://www.atlassian.com/software/jira`

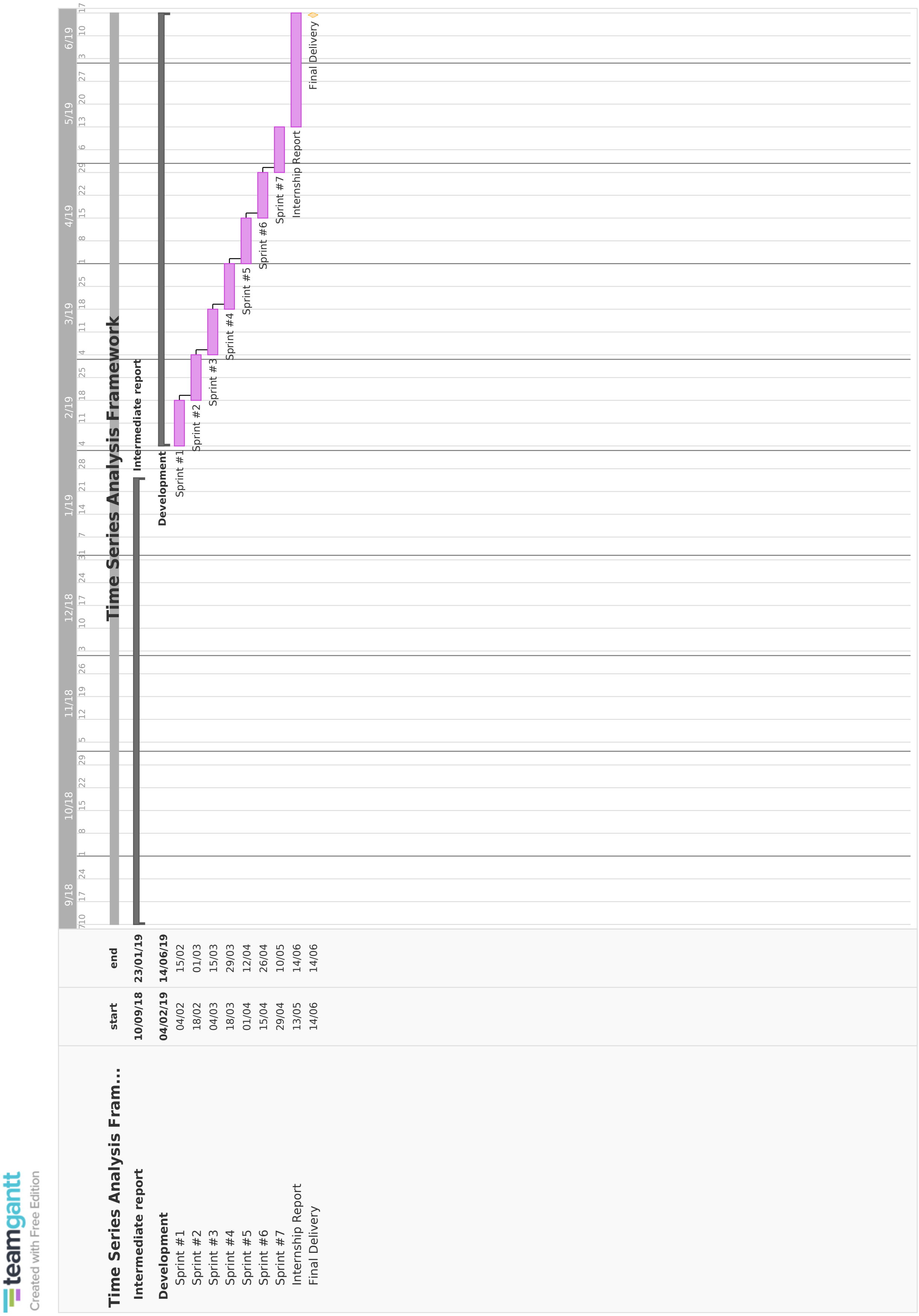
[4] `https://slack.com/`

[5] `https://bitbucket.org/`



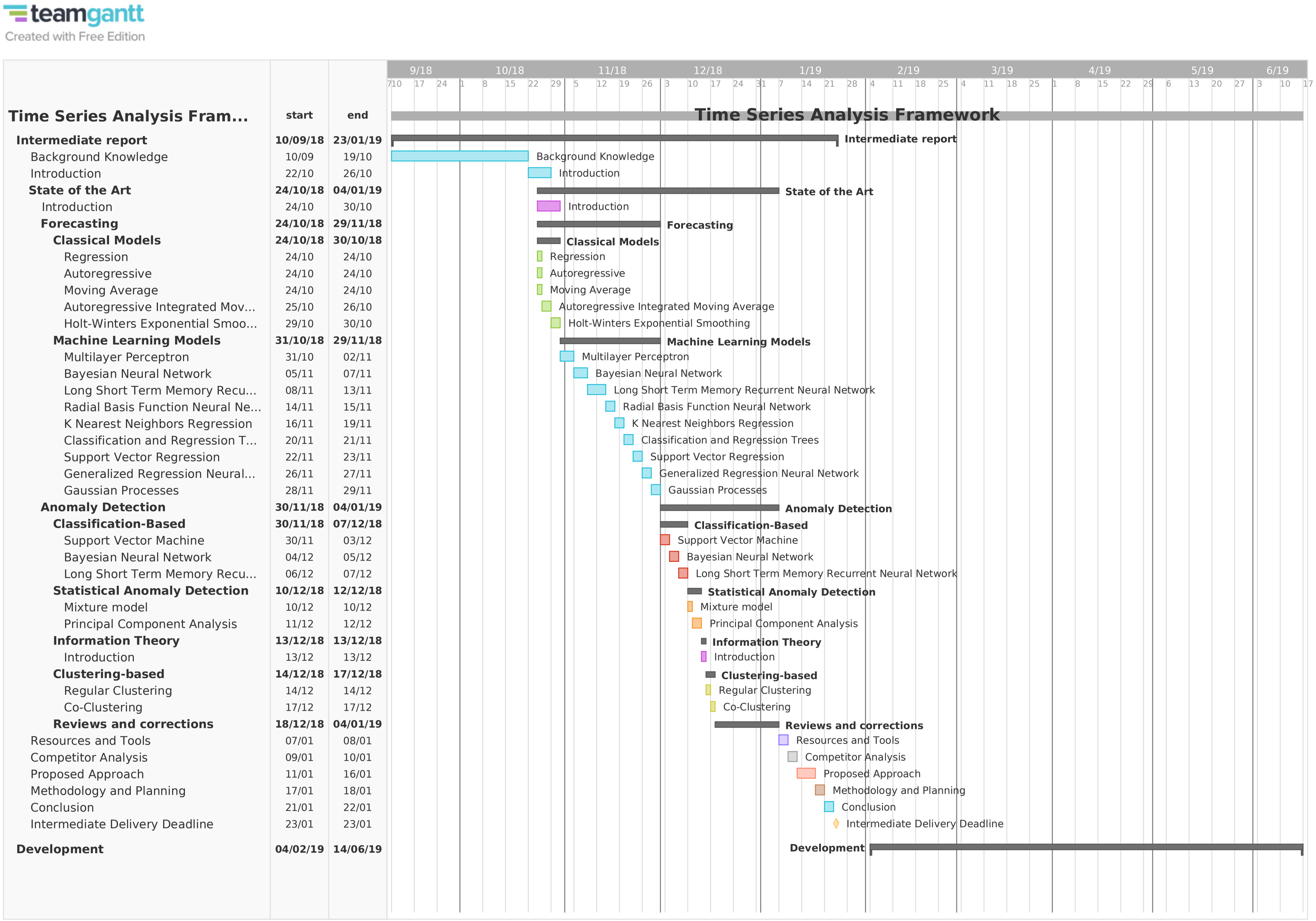
| Time Series Analysis Fram... | start | end |
|---|---|---|
| **Intermediate report** | **10/09/18** | **21/01/19** |
| Background Knowledge | 10/09 | 19/10 |
| Introduction | 22/10 | 26/10 |
| **State of the Art** | **24/10/18** | **19/12/18** |
| Introduction | 24/10 | 30/10 |
| **Forecasting** | **24/10/18** | **29/11/18** |
| **Classical Models** | **24/10/18** | **30/10/18** |
| Regression | 24/10 | 24/10 |
| Autoregressive | 24/10 | 24/10 |
| Moving Average | 24/10 | 24/10 |
| Autoregressive Integrated Mov... | 25/10 | 26/10 |
| Holt-Winters Exponential Smoo... | 29/10 | 30/10 |
| **Machine Learning Models** | **31/10/18** | **29/11/18** |
| Multilayer Perceptron | 31/10 | 02/11 |
| Bayesian Neural Network | 05/11 | 07/11 |
| Long Short Term Memory Recu... | 08/11 | 13/11 |
| Radial Basis Function Neural Ne... | 14/11 | 15/11 |
| K Nearest Neighbors Regression | 16/11 | 19/11 |
| Classification and Regression T... | 20/11 | 21/11 |
| Support Vector Regression | 22/11 | 23/11 |
| Generalized Regression Neural... | 26/11 | 27/11 |
| Gaussian Processes | 28/11 | 29/11 |
| **Anomaly Detection** | **30/11/18** | **19/12/18** |
| **Classification Based** | **30/11/18** | **11/12/18** |
| Support Vector Machine | 30/11 | 03/12 |
| Bayesian Neural Network | 04/12 | 05/12 |
| Long Short Term Memory Recu... | 06/12 | 07/12 |
| Rule-based | 10/12 | 11/12 |
| **Statistical Anomaly Detection** | **12/12/18** | **14/12/18** |
| Mixture model | 12/12 | 12/12 |
| Signal Processing Technique | 13/12 | 13/12 |
| Principal Component Analysis | 14/12 | 14/12 |
| **Information Theory** | **17/12/18** | **17/12/18** |
| Correlation Analysis | 17/12 | 17/12 |
| **Clustering-based** | **18/12/18** | **19/12/18** |
| Regular Clustering | 18/12 | 18/12 |
| Co-Clustering | 19/12 | 19/12 |
| Resources and Tools | 20/12 | 24/12 |
| Competitor Analysis | 25/12 | 26/12 |
| Proposed Approach | 27/12 | 02/01 |
| Methodology and Planning | 03/01 | 04/01 |
| Conclusion | 07/01 | 11/01 |
| Review | 07/01 | 18/01 |
| Intermediate Delivery Deadline | 21/01 | 21/01 |
| **Development** | **04/02/19** | **14/06/19** |

Figure 6.1: Initial Gantt Diagram detailing the first Semester.



| Time Series Analysis Fram... | start | end |
|---|---|---|
| Intermediate report | 10/09/18 | 23/01/19 |
| Development | 04/02/19 | 14/06/19 |
| Sprint #1 | 04/02 | 15/02 |
| Sprint #2 | 18/02 | 01/03 |
| Sprint #3 | 04/03 | 15/03 |
| Sprint #4 | 18/03 | 29/03 |
| Sprint #5 | 01/04 | 12/04 |
| Sprint #6 | 15/04 | 26/04 |
| Sprint #7 | 29/04 | 10/05 |
| Internship Report | 13/05 | 14/06 |
| Final Delivery | 14/06 | 14/06 |

Figure 6.2: Initial Gantt Diagram detailing the second Semester.



| Time Series Analysis Fram... | start | end |
|---|---|---|
| **Intermediate report** | **10/09/18** | **23/01/19** |
| Background Knowledge | 10/09 | 19/10 |
| Introduction | 22/10 | 26/10 |
| **State of the Art** | **24/10/18** | **04/01/19** |
| Introduction | 24/10 | 30/10 |
| **Forecasting** | **24/10/18** | **29/11/18** |
| **Classical Models** | **24/10/18** | **30/10/18** |
| Regression | 24/10 | 24/10 |
| Autoregressive | 24/10 | 24/10 |
| Moving Average | 24/10 | 24/10 |
| Autoregressive Integrated Mov... | 25/10 | 26/10 |
| Holt-Winters Exponential Smoo... | 29/10 | 30/10 |
| **Machine Learning Models** | **31/10/18** | **29/11/18** |
| Multilayer Perceptron | 31/10 | 02/11 |
| Bayesian Neural Network | 05/11 | 07/11 |
| Long Short Term Memory Recu... | 08/11 | 13/11 |
| Radial Basis Function Neural Ne... | 14/11 | 15/11 |
| K Nearest Neighbors Regression | 16/11 | 19/11 |
| Classification and Regression T... | 20/11 | 21/11 |
| Support Vector Regression | 22/11 | 23/11 |
| Generalized Regression Neural... | 26/11 | 27/11 |
| Gaussian Processes | 28/11 | 29/11 |
| **Anomaly Detection** | **30/11/18** | **04/01/19** |
| **Classification-Based** | **30/11/18** | **07/12/18** |
| Support Vector Machine | 30/11 | 03/12 |
| Bayesian Neural Network | 04/12 | 05/12 |
| Long Short Term Memory Recu... | 06/12 | 07/12 |
| **Statistical Anomaly Detection** | **10/12/18** | **12/12/18** |
| Mixture model | 10/12 | 10/12 |
| Principal Component Analysis | 11/12 | 12/12 |
| **Information Theory** | **13/12/18** | **13/12/18** |
| Introduction | 13/12 | 13/12 |
| **Clustering-based** | **14/12/18** | **17/12/18** |
| Regular Clustering | 14/12 | 14/12 |
| Co-Clustering | 17/12 | 17/12 |
| **Reviews and corrections** | **18/12/18** | **04/01/19** |
| Resources and Tools | 07/01 | 08/01 |
| Competitor Analysis | 09/01 | 10/01 |
| Proposed Approach | 11/01 | 16/01 |
| Methodology and Planning | 17/01 | 18/01 |
| Conclusion | 21/01 | 22/01 |
| Intermediate Delivery Deadline | 23/01 | 23/01 |
| **Development** | **04/02/19** | **14/06/19** |

Figure 6.3: Initial Gantt Diagram detailing the second Semester.

# Chapter 7

# Conclusions

The selection of the best algorithm to tackle either forecasting or anomaly detection on time series structures can be a complex task. The work carried out through the course of this internship will tackle this important aspect when performing analysis on time series, namely, in regards to the most appropriate model selection, as well as the model's parametrization. To achieve this, a selection of time series' features has to be defined and taken into account by this autonomous selector, while providing the user some degree of freedom in regards to the ratio of quality versus performance.

The State of the Art shows that Machine Learning (ML) approaches have been very frequent in the literature and provide excellent results in the domain of forecasting and anomaly detection, making them a viable option when compared to classical options and when considering the methods to include in the presented architecture for the proposed approach. Nevertheless, the referred classical models are even now a staple in time series forecasting, given their high accuracy and appropriate complexity.

The competitor Analysis shows that time series analysis contains a series of tasks that appear frequently in the business domain, having been explored by both companies who sought out to best market this service, as well as more popular competitors, such as Amazon and Microsoft. Current solutions also provide ML algorithms to cater the aforementioned services, sometimes even employing libraries that are available and were described as resources and tools that may as well be used in the content of this work.

In conclusion, this document was the result of the work for the current semester, detailing an exploration on the background knowledge and state of the art approaches regarding time series analysis, as well as resources and tools that provided some of the studied technologies. An analysis on competitors was also performed, as to infer over the available market solutions. Finally, a proposed approach was developed and presented. The methodology in use as well as planning for the project development have been defined as well, such that the work ahead includes the integral development of the time series analysis framework and a final report detailing the project further.

# Bibliography

[1] (2003). Nist/sematech e-handbook of statistical methods. section 6.4 introduction to time series analysis.

[2] Adhikari, R. and Agrawal, R. (2013). An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*.

[3] Ahmed, M. (2014). Network traffic pattern analysis using improved information theoretic co-clustering based collective anomaly detection.

[4] Ahmed, M., Mahmood, A. N., and Hu, J. (2016a). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications*, 60:19 – 31.

[5] Ahmed, M., Mahmood, A. N., and Islam, M. R. (2016b). A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems*, 55:278 – 288.

[6] Ahmed, N. K., Atiya, A. F., Gayar, N. E., and El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621.

[7] Alon, I., Qi, M., and Sadowski, R. J. (2001). Forecasting aggregate retail sales:: a comparison of artificial neural networks and traditional methods. *Journal of Retailing and Consumer Services*, 8(3):147–156.

[8] Armstrong, J. S. (2001). *Principles of forecasting: a handbook for researchers and practitioners*, volume 30. Springer Science & Business Media.

[9] Arshad, F. M., Ghaffar, R. A., et al. (1986). *Crude Palm Oil Price Forecasting Box-Jenkins Approach*. Universiti Pertanian Malaysia.

[10] Bailey, K. (2016). Gaussian processes for dummies.

[11] Basu, S., Bilenko, M., and Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 59–68, New York, NY, USA. ACM.

[12] Bauer, M. (1995). *General regression neural network, GRNN: A neural network for technical use*. University of Wisconsin–Madison.

[13] Ben-Gal, I. (2008). Bayesian networks. *Encyclopedia of statistics in quality and reliability*, 1.

[14] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

[15] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

[16] Box, G. and Jenkins, G. (1976). *Time series analysis: forecasting and control*. Holden-Day series in time series analysis. Holden-Day.

[17] Breiman, L. (1984). *Classification and regression trees*. Routledge.

[18] Brockwell, P. J., Davis, R. A., and Calder, M. V. (2002). *Introduction to time series and forecasting*, volume 2. Springer.

[19] Broomhead, D. S. and Lowe, D. (1988). Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom).

[20] Chatfield, C. (2003). *The Analysis of Time Series: An Introduction, Sixth Edition.* CRC press.

[21] Chauhan, S. and Vig, L. (2015). Anomaly detection in ecg time signals via deep long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–7. IEEE.

[22] Dorvlo, A. S., Jervase, J. A., and Al-Lawati, A. (2002). Solar radiation estimation using artificial neural networks. *Applied Energy*, 71(4):307–319.

[23] Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., and Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161.

[24] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

[25] Eskin, E. (2000). Anomaly detection over noisy data using learned probability distributions. In *In Proceedings of the International Conference on Machine Learning.* Citeseer.

[26] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer.

[27] Faraway, J. and Chatfield, C. (1995). Time series forecasting with neural networks: A case study. *University of Bath, Bath (United Kingdom), Research Report*, pages 95–06.

[28] G. Brown, R. and D.Litte, A. (1956). Exponential smoothing for predicting demand.

[29] Grosse R., S. N. (2015). Mixture models.

[30] Heller, K. A., Svore, K. M., Keromytis, A. D., and Stolfo, S. J. (2003). One class support vector machines for detecting anomalous windows registry accesses. In *Proc. of the workshop on Data Mining for Computer Security*, volume 9.

[31] Hill, T., O'Connor, M., and Remus, W. (1996). Neural network models for time series forecasts. *Management Science*, 42:1082–1092.

[32] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[33] Hu, W., Liao, Y., and Vemuri, V. R. (2003). Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, pages 282–289.

[34] Jolliffe, I. (2011). Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer.

[35] Kalekar, P. S. (2004). Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi School of Information Technology*, 4329008:1–13.

[36] Kanchymalay, K., Salim, N., Sukprasert, A., Krishnan, R., and Hashim, U. R. (2017). Multivariate time series forecasting of crude palm oil price using machine learning techniques. In *IOP Conference Series: Materials Science and Engineering*, volume 226, page 012117. IOP Publishing.

[37] Kotsialos, A., Papageorgiou, M., and Poulimenos, A. (2004). Long-term sales forecasting using holt–winters and neural network methods. *Journal of Forecasting*, 24(5):353–368.

[38] Kruegel, C., Mutz, D., Robertson, W., and Valeur, F. (2003). Bayesian event classification for intrusion detection. In *null*, page 14. IEEE.

[39] Lane, D. (2003). Online statistics education: A multimedia course of study. In *EdMedia: World Conference on Educational Media and Technology*, pages 1317–1320. Association for the Advancement of Computing in Education (AACE).

[40] Laptev, N., Yosinski, J., Li, L. E., and Smyl, S. (2017). Time-series extreme event forecasting with neural networks at uber. In *International Conference on Machine Learning*, number 34, pages 1–5.

[41] Lazzaroni, M., Ferrari, S., Piuri, V., Salman, A., Cristaldi, L., and Faifer, M. (2015). Models for solar radiation prediction based on different measurement sites. *Measurement*, 63:346 – 363.

[42] Lee, W. and Xiang, D. (2001). Information-theoretic measures for anomaly detection. In *Proceedings 2001 IEEE Symposium on Security and Privacy. S P 2001*, pages 130–143.

[43] Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23.

[44] Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E., and Winkler, R. (1982). The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of forecasting*, 1(2):111–153.

[45] Makridakis, S., Chatfield, C., Hibon, M., Lawrence, M., Mills, T., Ord, K., and Simmons, L. F. (1993). The m2-competition: A real-time judgmentally based forecasting study. *International Journal of Forecasting*, 9(1):5 – 22.

[46] Makridakis, S. and Hibon, M. (2000). The m3-competition: results, conclusions and implications. *International journal of forecasting*, 16(4):451–476.

[47] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018a). The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting.*

[48] Makridakis, S., Spiliotis, E., and Assimakopoulos, V. (2018b). Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3):1–26.

[49] Marchi, E., Vesperini, F., Weninger, F., Eyben, F., Squartini, S., and Schuller, B. (2015). Non-linear prediction with lstm recurrent neural networks for acoustic novelty detection. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE.

[50] Mason, J. (2018). Cyber security statistics.

[51] McCallum, A., Nigam, K., and Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM.

[52] Mitchell, T. M. (1997). *Machine Learning.* McGraw-Hill, Inc., New York, NY, USA, 1 edition.

[53] Münz, G., Li, S., and Carle, G. (2007). Traffic anomaly detection using k-means clustering. In *GI/ITG Workshop MMBnet*, pages 13–14.

[54] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective. Adaptive Computation and Machine Learning.* MIT press.

[55] Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

[56] Noor-Ul-Amin, M. (2010). Forecasting with neural networks: A comparative study using the data of emergency service. *arXiv preprint arXiv:1010.3501.*

[57] Olah, C. (2015). Understanding lstm networks.

[58] Papalexakis, E. E., Beutel, A., and Steenkiste, P. (2014). Network anomaly detection using co-clustering. In *Encyclopedia of Social Network Analysis and Mining*, pages 1054–1068. Springer.

[59] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian process for machine learning.* MIT press.

[60] Sayad, S. (2019). Support vector machine - regression (svr).

[61] Sharda, R. and B. Patil, R. (1992). Connectionist approach to time series prediction: An empirical test. *Journal of Intelligent Manufacturing*, 3:317–323.

[62] Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., and Chang, L. (2003). A novel anomaly detection scheme based on principal component classifier. Technical report, MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.

[63] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

[64] Soni, D. (2018). Introduction to bayesian networks.

[65] Specht, D. F. (1991). A general regression neural network. *IEEE transactions on neural networks*, 2(6):568–576.

[66] Strang, G. (2016). *Introduction to Linear Algebra, 5th Edition.* Wellesley-Cambridge Press.

[67] Voyant, C., Notton, G., Kalogirou, S., Nivet, M.-L., Paoli, C., Motte, F., and Fouilloy, A. (2017). Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569–582.

[68] Warrender, C., Forrest, S., and Pearlmutter, B. (1999). Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE.

[69] Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372.

[70] Ye, N. and Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17(2):105–112.