NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

PEDRO MIGUEL MATOS SILVA

BSc in Computer Science

# BETTER MONITORIZATION AND PREDICTABILITY FOR COMPUTATION INFRASTRUCTURES THROUGH DATA FUSION

NOVA University Lisbon
03, 2023

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

# BETTER MONITORIZATION AND PREDICTABILITY FOR COMPUTATION INFRASTRUCTURES THROUGH DATA FUSION

**PEDRO MIGUEL MATOS SILVA**

BSc in Computer Science

**Adviser:** João Leitão
*Associate Professor, NOVA University Lisbon*

**Co-advisers:** Cláudia Soares
*Assistant Professor, NOVA University Lisbon*

Belinda Lourenço
*Central DSI Director, novobanco*

NOVA University Lisbon
03, 2023

**Better monitorization and predictability for computation infrastructures through data fusion**

# Acknowledgements

I want to thank everyone that accompanied me during this journey. Firstly, a special word of gratitude to professors João Leitão and Cláudia Soares, as well as Mrs. Belinda Lourenço and Mr. Paulo Alvarez for supervising this thesis and for providing their support in its realization.

Next, I want to thank my family for their incredible effort that allowed me to have the opportunity to be here, as well as for the strength and encouragement that they continuously provided.

I would also like to thank all my friends at *novobanco*, particularly Gonçalo Mateus and Pedro Teixeira, for all the ideas exchanged, which helped immensely during this thesis work.

Finally, I want to show my gratitude to all my friends and colleagues that helped me throughout the years, their presence and support were crucial for me to reach this point.

*"No journey is too great, when one finds what one seeks."*
*(Friedrich Nietzsche)*

# Abstract

Cloud computing provides a wide variety of services that allow its users to develop and maintain applications at a relatively low cost. The services offered by a cloud system also provide the ability to flexibly scale the resources of any application. Because of these benefits, this type of computing has become extremely popular. Although cloud computing offers these great dynamic resource capabilities, for reasons such as the protection and confidentiality rules for sensitive data, some companies opt to store a part of or all of their resources in a private on-premise data center. However, on-premise infrastructure entails a higher level of management since it requires constantly making changes, such as the continuous scaling of resources, in order to fit the ever-changing needs of its applications and services. This puts a heavy burden on the staff, which monitors the infrastructure and is responsible for making such management decisions.

In this work, in collaboration with *novobanco*, we aim to design procedures that can be used to build a decision support system with the objective of facilitating the management process of the data center infrastructure. These procedures include ways to extract and combine data from the various hosts in order to analyze the similarity between them, as well as machine learning approaches that, by using user input about a labeled issue, can identify similar infrastructure where the same issue might be observed.

We focus on labeling over-provisioned infrastructure and suggesting decisions on the labeled results. Additionally, we analyze the economic impact of following the recommended decisions by using the prices of similar hardware instances from an external provider. Our estimates indicate relative savings of 12% to 18%, depending on the utilized approach and how the changes are applied.

**Keywords:** Decision Support Systems, Monitoring Systems, Cloud Computing, Semi-supervised Learning, Dynamic time warping

# Resumo

A computação em nuvem fornece uma grande variedade de serviços que permitem aos seus utilizadores desenvolver e manter aplicações a um custo relativamente baixo. Os serviços oferecidos por um sistema de computação em nuvem proporcionam também a capacidade de escalar de forma flexível os recursos de qualquer aplicação. Devido a estes benefícios, este tipo de serviços tornaram-se extremamente populares. Embora a computação em nuvem ofereça grandes capacidades dinâmicas de recursos, por razões como a protecção e a confidencialidade de dados sensíveis, algumas empresas optam por armazenar uma parte ou a totalidade dos seus recursos localmente. No entanto, a infra-estrutura local implica um nível de gestão mais elevado por parte da empresa, uma vez que requer mudanças constantes, tais como o contínuo ajuste de recursos, de modo a adaptar-se às necessidades das suas aplicações e serviços (que mudam constantemente). Isto representa um aumento na carga de trabalho para os funcionários especializados que gerem a infra-estrutura e tomam estas decisões de gestão.

Neste trabalho em colaboração com o *novobanco*, pretendemos conceber procedimentos que possam ser utilizados para construir um sistema de apoio à decisão com o objectivo de facilitar o processo de gestão de infraestrutura. Estes procedimentos incluem formas de extrair e combinar dados de vários servidores, de modo a analisar a semelhança entre eles, bem como abordagens de aprendizagem automática, que, utilizando *feedback* do utilizador sobre problemas identificados, podem identificar servidores semelhantes onde o mesmo problema possa ser observado. Concentramo-nos na identificação de infra-estruturas sobreprovisionadas e sugerimos decisões sobre estas identificações. Além disto, analisamos o impacto económico de seguir as decisões recomendadas, utilizando os custos de *hardware* semelhante de um fornecedor externo. As nossas estimativas indicam poupanças de custo relativas de 12% a 18%, dependendo da abordagem utilizada e da forma como as recomendações são aplicadas.

**Palavras-chave:** Sistema de Apoio à Decisão, Sistemas de Monitorização, Computação em Nuvem, Aprendizagem Semi-supervisionada, Alinhamento Temporal Dinâmico

# Contents

# List of Figures

# List of Tables

# Acronyms

$DTW_d$ Dependent Dynamic Time Warping 16
$DTW_i$ Independent Dynamic Time Warping 16
$RBF_G$ Gaussian Radial Basis Function 18

**CI** Configuration Item 23
**CMDB** Configuration Management Data Base 23
**CPU** Central Process Unit 7

**DC** Data Center 5
**DSS** Decision Support System 8
**DTW** Dynamic Time Warping 13

**EDA** Exploratory Data Analysis 27

**FaaS** Functions as a Service 5
**FTP** File Transfer Protocol 24

**HTTP** Hypertext Transfer Protocol 24

**IaaS** Infrastructure as a Service 5
**IP** Internet Protocol 4

**KNN** K-Nearest Neighbors 17

**PaaS** Platform as a Service 5
**PCA** Principal Component Analysis 18

**QoS** Quality of Service 1

**RAM** Random Access Memory 7
**RBF** Radial Basis Function 17
**RRA** Round Robin Archives 25
**RRD** Round Robin Database 25
**RRDs** Round Robin Databases 25

**SaaS** Software as a Service 5
**SAP** Systems Applications and Products 22
**SNMP** Simple Network Management Protocol 24
**SSH** Secure Shell 24

**XML** Extensible Markup Language 36

# 1 Introduction

## 1.1 Context

Cloud computing [34, 49] offers an enormous share of digital services that provide flexible and fast services for users at a relatively cheap price. It is one of the major computation infrastructure technologies because it is quicker and easier to develop and maintain applications while reducing the initial hardware cost. Adding to all of this, cloud computing provides the potential for high scalability and elasticity which is relevant to this case, since when the load increases, cloud services can allocate more resources to handle more requests. This is called auto-scaling and it allows cloud infrastructures to adapt to the application needs, ensuring reliability and quality of service while also minimizing cost expenditure (when the load decreases and fewer resources are needed).

Because of this, cloud data centers are exceedingly popular as a way to host or serve software or application data. However, even though cloud services are widely considered to be safe and generally a good choice, they might not be an option for all applications. Some companies, for legal and confidentiality reasons, are unable to use the cloud to host applications or services which deal with sensitive data. For reasons like these, and other motivations such quality of service (QoS) issues, companies frequently resort to private on-premise infrastructure for some applications. On-premise infrastructure, however, entails a higher level of management by any company that chooses to have full control and responsibility over its total or partial computation infrastructure.

As solutions and services become more complex, branched, and interconnected, specific approaches need to be designed to deal with events on different portions of the data center, hence progressively increasing managing demand. Frequently, some applications start to have a lower or higher usage and require resource adjustments to ensure their favorable functioning or to minimize the costs of over-provisioning resources for under-utilized hardware. The progress of cloud computing and the increase in the needs of many systems might also invite changes that involve the migration of entire applications (or some services within them) to the cloud.

These decisions are often based on a set of rules to determine firstly the problems of

the considered data set and then what type of resolution process needs to be applied to solve its specific issues or to improve upon a given system. However, as companies grow so do the applications and machine specifications, which tend to become more diversified, making it difficult to define a set of rules that apply to the whole server network. Often, there are too many factors to consider, and since the infrastructure becomes more and more heterogeneous, the decision-making about the management of such infrastructure starts to apply to relatively smaller and smaller clusters of machines.

All the decisions and adjustments about the specific portions of the server infrastructure needs to be made based on the knowledge and intuition of specialized technical personnel, which results in increased management complexity and costs.

## 1.2 Motivation

It is crucial for modern computation systems to reduce or facilitate human interaction with their functionalities and internal management. The importance of dynamizing and facilitating decision-making reflects itself on a better usability of the system as well as on the minimization of unnecessary expenditure.

To enhance the decision-making process on a large and diverse data set, it is first necessary to improve the identification of the reasons behind the designated taken decisions and what patterns of data influence, contribute to, or explain those decisions.

The longer important beneficial changes take to be applied, the higher the costs associated with not taking such changes tend to be. However, a continuous non-automated way of identifying and deciding which changes need to be done is often slow and costly, since it involves multiple tasks from the staff[2], such as workload profiling to discern if there is a problem of under or over-utilization, analyzing which applications and services are hosted on the machines as well as their importance and QoS needs, and estimating the cost, effort and benefit of taking the decisions on the intended subset of machines.

By easing the management process it is possible to speed up the recognition of systematic problems and, therefore, their resolution, lowering the costs of maintaining unnecessary (over-provisioned) infrastructure and lowering the periods of worse usability caused by unidentified problems of resource under-provisioning.

## 1.3 Objectives

In this thesis, we focus on building mechanisms that, by using all the available data about a data center (in this case, the *novobanco* data center), can identify and recommend courses of action based on previously acquired knowledge. Thus, we designed a process to extract and combine data from the platforms that monitor and manage the *novobanco* infrastructure. Subsequently, by taking user input labels about identified problems in the hosts (or hosts which are candidates for potential changes), we explored ways to use machine learning to extend this user domain knowledge to other infrastructure within

the *novobanco* data center. This was done by using the combined data to measure the similarity between all the different hosts and, using that similarity measure, propagate the host labels to similar infrastructure. Using the results from the machine learning models we implemented a simple decision support tree that recommends actions based on heuristics discussed with the *novobanco* staff. Finally, after validating the results with the staff, we estimate the economic impact of following the recommended actions by mapping the hardware specifications of the studied data set of hosts to instances with equivalent specifications on a cloud service (Azure). The contributions of this thesis work are summarized in Figure 1.1.

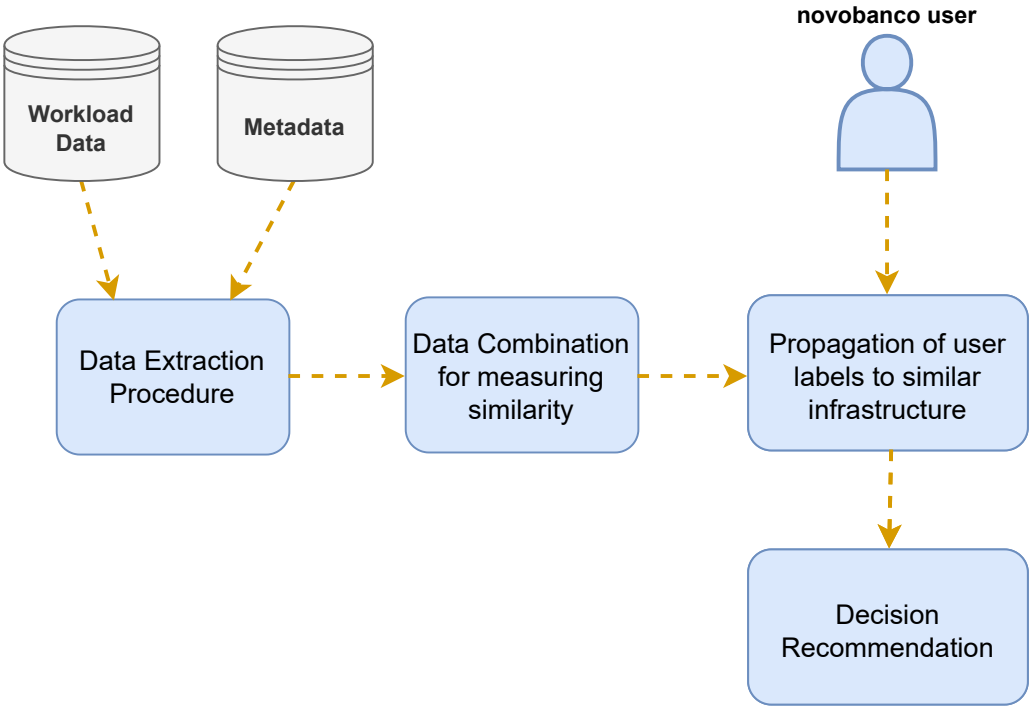



Figure 1.1: Thesis Overview

## 1.4 Research Context and Security Constraints

This dissertation is done under the context of a research project conducted in collaboration with *novobanco* DSI. *novobanco* [33] is a banking company established in 2014 with 292 active branches, 20 Corporate Centers, 4090 employed professionals, and more than 1.5 million customers. The company is responsible for supporting several social aspects and promoting digital inclusion in the Portuguese economy.

Due to the security and confidentiality constraints of *novobanco*, we are unable to reveal information that could be used in a harmful way. This information includes the

name and IP address of the hosts, their domains, names of staff personnel, infrastructure locations, and other critical information. Therefore, some specific details from our results only have meaning inside the context of the *novobanco* closed network environment and need to be hidden.

## 1.5 Document Structure

The following chapters are organized as follows:

- In **Chapter 2** (Related Work) we introduce some of the research that was done to understand the concepts of the subsequent chapters.
- In **Chapter 3** (Novobanco Data Center) we present the *novobanco* data center. This chapter has the purpose of describing its overall functioning as well as the technologies which are used to monitor, manage and extract information from the infrastructure machines. We also perform an analysis of the workload features, explaining what is being monitored and which information can be relevant to our work.
- In **Chapter 4** (Methodology) we describe in detail the followed thesis process. We explain how the data was extracted, which methods were used to combine all the data features, and how to use a similarity representation of the data to expand labels given by a user. We also explore ways to use the results to aid the staff in their management decision-making process.
- In **Chapter 5** (Cost-Benefit Analysis) we estimate the economic impact of following the decisions recommended in **Chapter 4**. To do this we utilize the costs of similar infrastructure from a cloud provider.

# 2

# Related Work

## 2.1 Data Centers and Cloud Computing

A Data Center (DC) [37] is a facility that houses computing systems and their associated infrastructure. It is used to meet the need of various businesses or organizations that host considerable amounts of data. Each DC is a collection of numerous servers and network infrastructures, which, together with specialized external cooling systems and power supplies tries to ensure a reliable and available functioning of its systems [6]. With the increase in popularity in the development of cloud-based systems, they have become crucial to the development of the information technology industries, since they provide the physical infrastructure and resources that assure the effectiveness of all services offered by cloud providers making their functioning critical for cloud development.

A DC can be classified into numerous different categories, depending on its intended purpose. By categorizing them by their reliance on external cloud providers, it is possible to isolate three main DC types: **on-premise** data centers, **cloud-based** data centers, and **hybrid** data centers. **On-premise** data centers are the solution used by companies that want to house and maintain their own physical infrastructure, while **cloud-based** data centers solutions refer to companies that want to host their services and applications using infrastructure that is managed by an external cloud provider. **Hybrid solutions** combine both previous approaches, housing some of its service infrastructure while keeping some running on a cloud environment. Cloud-based systems can provide various distinct services made to fulfill different company requirements, these services include: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Functions as a Service (FaaS). Cloud-based systems can be used to improve efficiency, and security and reduce the costs of a service, however, these advantages are relatively nuanced. Although a cloud system can be widely considered safe due to security features such as encrypted communications, data breaches can still happen in cloud environments, and by storing its data in external infrastructure, a company is trusting its security and confidentiality to the correspondent cloud providers[2]. Another discussed topic is the cost of cloud infrastructure versus on-premise infrastructure, specifically how the cost

adds up over time. [22] On local infrastructure, the cost of the hardware is typically fixed, and new provisioning of hardware is a slower and longer process. This can stall long-term investments in resources, which affects the agility of running services. Using cloud resources changes the rising costs of maintaining an on-premise approach to a model where the users pay what they use, so instead of being forced to invest in physical resources with a certain fixed amount of capacity, it is possible to adapt the resource capacity as requirements change.

Although the cloud has these dynamic capabilities, there is still the need to consider various factors before making any decision. A pay-as-you-go model entails that increasing the resources of a service also increases its cost per time. Consequently, **over-provisioning** resources can be expensive long term, and **under-provisioning** resources can cause performance issues to any service. A cost analysis is crucial to determine exactly what each service needs and to decide if a service is even a good candidate for a cloud environment. Since cost is directly related to usage in a cloud model, these decisions should be made based on the workload of each service or application, which tends to change over time based on its specific demands. This workload will often be unpredictable, but it can also follow certain predictable patterns. Using a cloud model, it is possible to dynamically alter resource consumption in a way that answers the performance requirements, and that also tries to minimize the costs, this is especially helpful when the workload has a considerable amount of fluctuation. Some examples of workloads that are beneficial for a cloud approach when compared to a local data center include:

- Workloads that are periodically inactive, enabling periodic shutdowns of resources during these periods of inactivity and avoiding unnecessary costs (Figure 2.1).

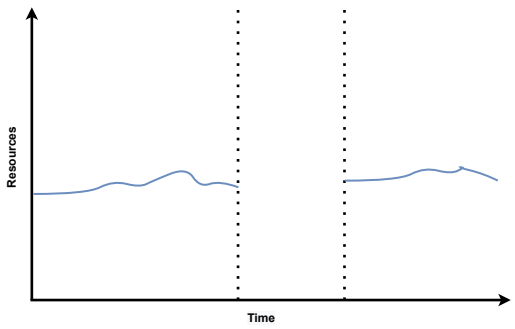



Figure 2.1: Illustration of a periodically inactive workload

- Workloads that have unexpected peaks of usage, in order to account for these, on-premise infrastructures need to have resources for the maximum usage values, which implies a waste of resources. On a cloud approach resources might have the possibility to be dynamically allocated when a peak happens (Figure 2.2).

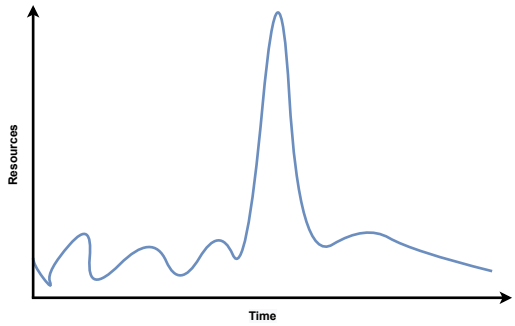



Figure 2.2: Illustration of a workload with unexpected peaks of usage

- Workloads with seasonal peaks of usage. The advantage of a cloud model is also notorious here. It is possible to provision services just before those peaks of usage are expected to happen, not only avoiding a waste of resources in the periods where the usage is low but also not sacrificing performance when the usage spikes (Figure 2.3).

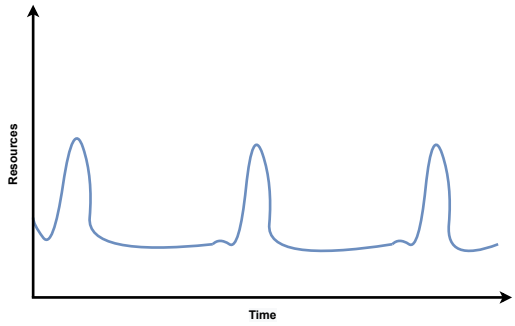



Figure 2.3: Illustration of a workload with seasonal peaks of usage

Companies employing a hybrid solution for their data centers where some services are running on cloud infrastructure and some on on-premise infrastructure, need to constantly contemplate the pros and cons of migrating some of the on-premise services to the cloud in order to evaluate if such migrations provide any benefit. They also have to keep in consideration the needs of the services currently hosted on local physical infrastructure, which, as previously stated, keep changing over time. If an on-premise service starts to grow, it is necessary to scale the infrastructure accordingly, this scaling can be done vertically or horizontally. **Vertical scaling** is the process of adjusting the capacity of current resources, for example adding (or removing) RAM, CPU, or storage space to an existing server. **Horizontal scaling** on the other hand is the process of adjusting the number of server instances, by adding (or removing) resources to a service or resource cluster. In order to avoid service malfunctioning, companies tend to prepare their local

hosts for the maximum usage of a service, this however entails a higher level of resource waste, which can reflect in an unnecessary increase in the costs. It might be necessary to then scale down the resources by reducing the hardware of specific resources or scale in by consolidating or removing the number of instances running the service. The scaling of resources is often difficult to automate, each service has its different intricacies, needs, and usage levels, so, as applications grow, resource management decisions start to become a considerable challenge.

## 2.2 Decision Support Systems

A Decision Support System (DSS) [19] is a computer program that facilitates decision-making by analyzing data and providing insights and recommendations. These systems have a variety of current uses, including forecasting stock values and offering investment advice in the stock market[20], analyzing the behavior of customers, determining their preferences to improve future marketing strategies[45], and even in healthcare to facilitate diagnosis, clinical decisions, and treatment planning [23].

Generally, [32], DSS have three separate components, a **knowledge base**, a **model management system**, and a **user interface**. The knowledge base is in essence a database that contains information related to the context at hand from both internal and external sources, and that is used as a foundation for future courses of action. The model management system is the component responsible for providing a better understanding of how the system works and what can be done to make it better, companies employ the models in this system to forecast how outcomes will change as a result of various system modifications or staff decisions. The user interface is simply the end component for user interaction with the entire system. The relations between these components are displayed in Figure 2.4.

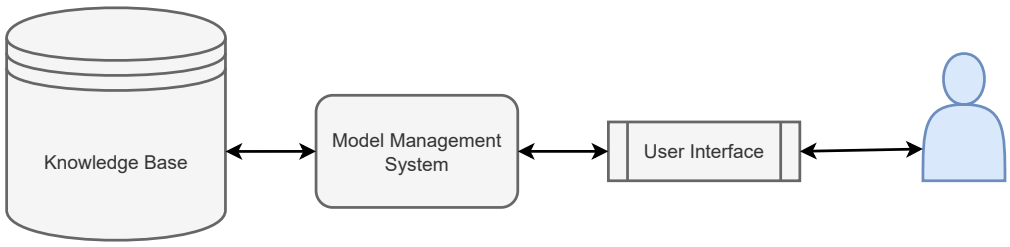



Figure 2.4: Decision Support System Components

In the context of data centers, DSS can be used for managing server infrastructure, which is often heterogeneous, by helping to level the resource distribution to the needs

of a business or application[10], there are even instances of their usage to promote sustainability and reduce carbon dioxide emissions in data centers [29]. Another example of DSS usage in this context is in cloud migration. There are often difficulties associated with adapting applications for a cloud environment, which include: [2]

- **Economic and Legal issues** - Some companies (especially in the banking field) have legal requirements regarding user data, as an example, the European Union mandates that the physical location of private data from European users must be inside its borders. [3] Migrating applications to the cloud also might implicate some total or partial redesigns of such applications, which multiplies the costs of the company.
- **Scalability issues** - As previously stated, there are 2 types of scalability, **vertical** and **horizontal** scalability. In a cloud environment, these definitions have a similar meaning. **Vertical** scalability entails the increase of computing resources for a service, while **horizontal** scability entails the increase of service instances. Generally, vertical scalability can be always applied, although the performance of this scalability will depend on the services offered by the provider, horizontal scalability on the contrary is best suited to be utilized on the specific architecture of an application.
- **Quality of Service issues** - The migration of an application to the cloud implies relying on the provider to assure the quality of service needs of the application. If the application has the need for a lower network latency or a stable performance with low levels of variability, it might not be advisable to migrate the application to a cloud provider.
- **Security and Confidentiality issues** - Although cloud services are widely considered secure (often considered safer than on-premise solutions) some companies have specific data retention policies which ensure that eventual data breaches are the responsibility of the company assigned to hold that data.

Decisions about data center services such as migrations to a cloud environment or the adjustment of physical resources, require multiple tasks to be carried out by its specialized staff. Although these decisions are fairly different, some of the tasks performed to reach a decision are identical or at least performed in a similar or comparable manner. Some of these tasks include [2]:

- **Workload profiling** - Establishing target workloads where a decision is applicable. For example, comparing the expected workload of an application versus its current usage to define elasticity decisions.
- **Identification of critical services** - Recognizing the security and quality of service constraints of the critical services in the data center and discerning if the to-be-applied decision obeys such constraints.

- **Cost and effort estimation** - Identifying the cost of applying a decision and what kind of effort would be needed to apply it. Evaluating the work it takes to adapt the infrastructure and/or its applications in order to follow the decision procedure.

Each one of these tasks represents a point of entry to the decision-making process. It is necessary then to combine all this heterogeneous information in a way to feed it to a model that can then use it to predict future courses of action. In Section 2.3 we present some approaches to combine different types of data from multiple sources, and in Section 2.4 we describe some of the approaches that can be used to extend an identified choice onto the rest of the data set.

## 2.3 Data Fusion

With the increase of available heterogeneous data from different origins, the interest in accessing and combining information through a consistent interface has grown [8]. This is the objective of **Data Fusion**, which is the process of extracting and combining data from multiple sources in order to create accurate inferences about a system or subject. Data fusion is based on three tasks: data alignment, data correlation, and attribute/identity estimation [42].

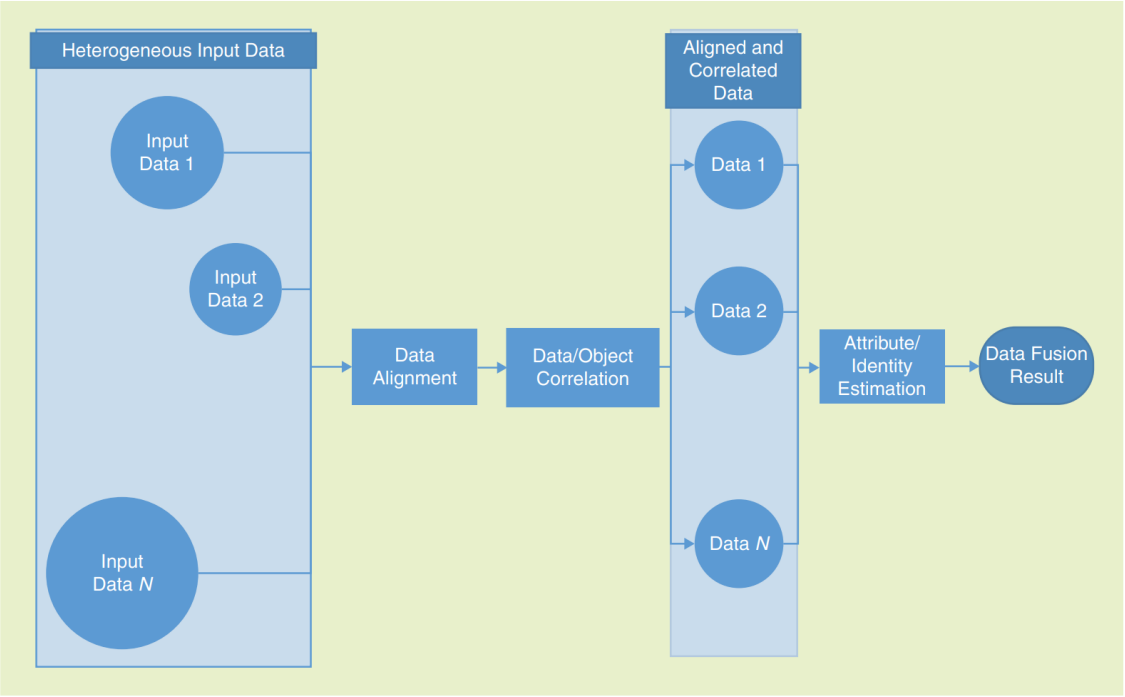



Figure 2.5: Generalized Data Fusion Process, [42]

As shown in Figure 2.5 the data alignment task corresponds to the first step, and is generally defined by coordinate transformations and unit adjustments that have the objective to provide a representation of an entity. The data correlation step is about

associating (correlating) multiple input measures with each other, effectively choosing which measurements are going to be combined in the final step. In the attribute/identity estimation step, the aligned and associated measurements from heterogeneous sources are then combined in an estimation framework to infer the desired information about the subject at hand. Attribute/identity estimation is the main step of the data fusion process and it is a current focus of research in statistical estimation theory and machine learning fields.

Data fusion can be separated into three main categories:

- **Observation-level fusion** - Where the raw data measurements are combined directly to produce a result.
- **Feature-level fusion** - Where there is a preparatory step of extracting some features that can represent the original source data.
- **Decision-level fusion** - Where besides a feature extraction step, there is an intermediate analysis of the extracted data before it can then be combined.

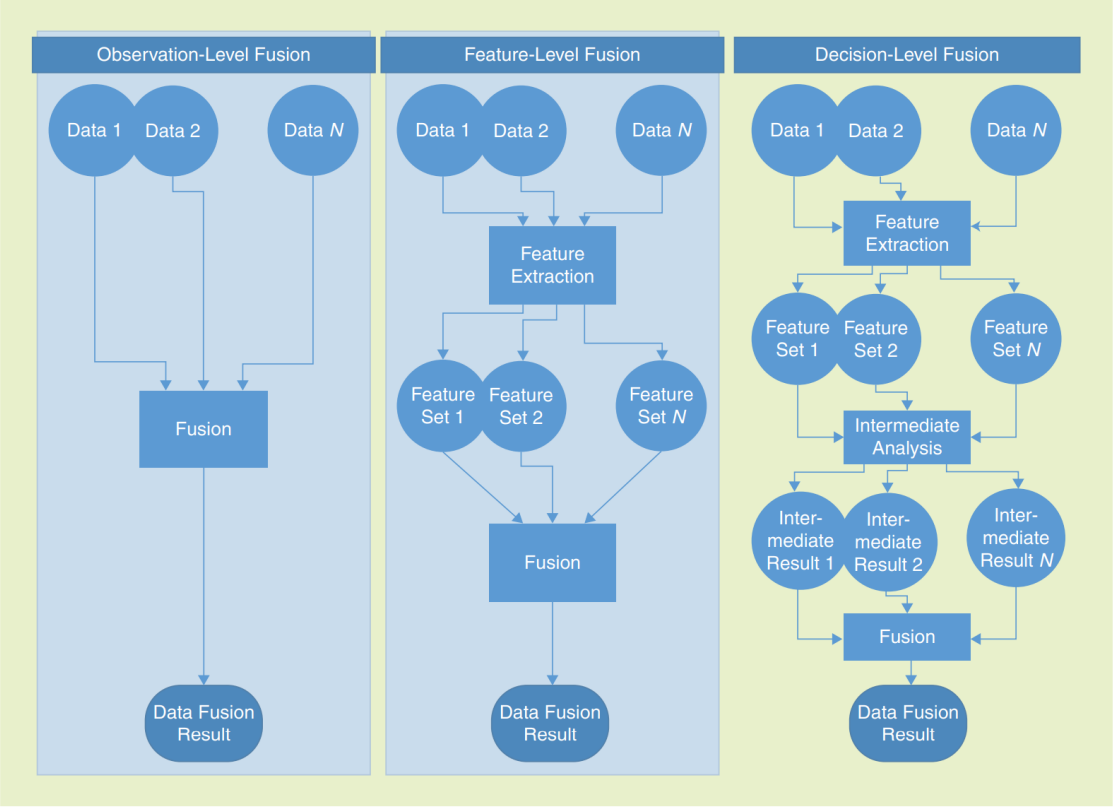



Figure 2.6: The three main types of Data Fusion, [42]

In this work, we utilize a decision-level data fusion framework on the data set. Besides the data extraction which will be explained in 4.3, for the intermediate analysis and fusion steps we employed some techniques that will be explained in the following sections.

## Encoding Categorical Data

Categorical data is all the information that is used to label a data point, instead of representing a numerical value, categorical variables, often called nominal variables, can give a broader insight into certain data sets, and therefore it is important that machine learning algorithms can absorb information from these variables. However, the input features for these algorithms are usually required to be numeric, and therefore to make use of categorical variables, these need to be converted to an interpretable numeric format. This conversion can be done in different ways [9], the most common being **ordinal encoding**, **one-hot encoding** and **dummy variable encoding**.

**Ordinal encoding** assigns a numeric value to each category represented in a categorical variable. This conversion method entails an ordinal (or ranking) relationship between the variables since the numeric values are represented as integer values and integer values have a naturally ordered relationship that will be interpreted by future algorithms. If there is no ranking visible between the variables, this method may produce undesired results, and therefore a different encoding method needs to be used.

**One-hot encoding** is used when there is no ordinal relationship between the categories. It works by creating a feature for each category that exists in a categorical variable. Using the colors of a traffic light as an example: [red, yellow, green], each color (category of the variable traffic light) will appear with a binary representing if the color is present, for example, a red light would be represented as [1,0,0].

By creating a variable for each category, **one-hot encoding** introduces redundancy in the representation of a categorical feature. To represent $n$ different categories there is no need for $n$ features, it is possible to represent those $n$ features with $n-1$ features. This method of representation is called **dummy variable encoding**, and it is used to mitigate this redundancy issue.

## Gower's Distance

Gower's Distance [16], introduced by mathematician Timothy Gowers, is a coefficient which measures the dissimilarity between two sampling units. It is used to measure the difference of mixed data points, which are data points that contain features from more than one of three types of features described by Gower's: **quantitative features**, **qualitative features** and **dichotomous features**. The formula for the Gower's distance ($d_G$) between two observations is represented in (2.1). The $\delta_{ijk}$ is an indicator function that takes the value 1 if the $i$-th and $j$-th observations can be compared along the $k$-th feature and 0 otherwise, and $s_{ijk}$ is the similarity score between the $i$-th and $j$-th observations along the $k$-th feature. The resulting Gower's distance ranges between 0 (when two observations are identical) and 1 (when two observations are completely different).

$$d_G(x_i, x_j) = 1 - \frac{\sum_{k=1}^{p} \delta_{ijk} s_{ijk}}{\sum_{k=1}^{p} \delta_{ijk}} \tag{2.1}$$

The similarity score ($s_{ijk}$) is computed in a different way for each type of feature. For the **quantitative features**, it is measured using the range normalized Manhattan distance (2.2).

$$s_{ijk} = 1 - \frac{|x_{ik} - x_{jk}|}{R_k} \tag{2.2}$$

For the **qualitative features**, the similarity measurement is based on the hamming distance (2.3)

$$s_{ijk} = 1\{x_{ik} = x_{jk}\} \tag{2.3}$$

For the **dichotomous features** (which are not considered in this study), the similarity ($s_{ijk}$) and the validity ($\delta_{ijk}$) values are computed using a table of dichotomous caracter comparisons (Figure 2.7).

| | Values of character $k$ | | | |
|---|---|---|---|---|
| Individual $i$ | + | + | − | − |
| $j$ | + | − | + | − |
| $s_{ijk}$ | 1 | 0 | 0 | 0 |
| $\delta_{ijk}$ | 1 | 1 | 1 | 0 |

Figure 2.7: Scores and Validity of Dichotomous Character Comparisons , [16]

By using the Gowers distance we can combine (or fuse) different types of features onto a single representation. By computing this distance on every pair of data points in a data set, we can discern how different each data point is in comparison to the others.

## Dynamic Time Warping

Dynamic Time Warping (DTW) [24], is a technique that has the purpose of finding an optimal alignment between two sequence patterns. It is often used in speech pattern recognition [30], in the stock market [15] and other fields such as music recognition [31] and time series clustering [48].

The idea behind using DTW is that two time series which can be considered similar, might not coincide in every point by, for example, starting at a different moment of their

pattern or by having different pattern speeds. In Figure 2.8 its possible to observe two fairly similar patterns that, if compared by direct point-to-point matching (euclidean matching) will have a higher dissimilarity simply because they do not coincide. Dynamic time warping aims to solve this issue by trying to find the minimum distance between each point of both sequences.

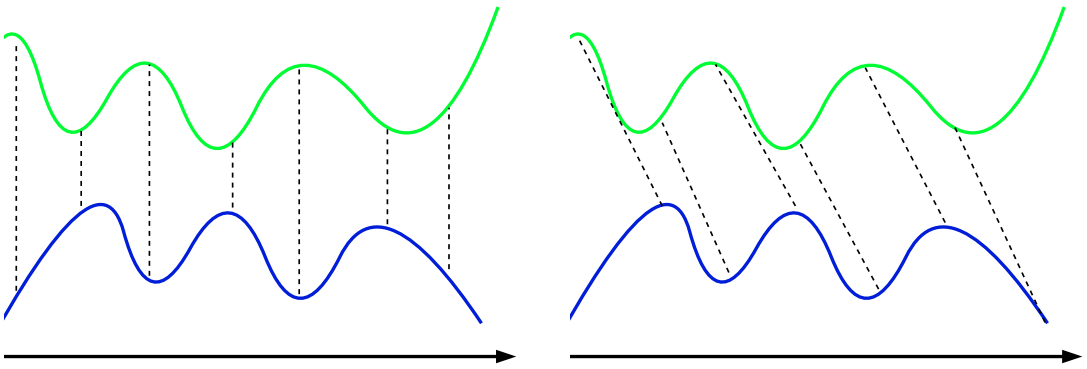


Figure 2.8: Euclidean matching and Dynamic Time Warping matching comparison

The objective of these time series comparison methods is to compute the distance between two input time series. In classical euclidean matching, this distance between two time series (x and y) is simply calculated as a sum of the distances (d) between all matched points (2.4).

$$\sum_{i=1}^{n} d(x_i, y_i) \tag{2.4}$$

In dynamic time warping [44] this distance function is often named "cost function", and the task of searching for the optimal alignment between two sequences is the task to find an alignment that minimizes this cost function. This algorithm starts by taking a local cost matrix, calculated using the Manhattan distance (2.5).

$$C_l \in \mathbb{R}^{N \times M} : c_{i,j} = \|x_i - y_j\|, \quad i \in [1:N], \; j \in [1:M] \tag{2.5}$$

Then it proceeds to find the optimal alignment path (or warping path), which has to satisfy three conditions:

- **Boundary condition**: The first points of the warping path have to be the starting points of the aligned sequences, and the last points of the warping path have to correspond to the last points in the aligned sequences.

- **Monotonicity condition**: The time-ordering of the points must be maintained.

- **Step size condition**: While aligning sequences, the warping path is restricted from making big shifts in time.

This optimal alignment path [44] is found using the distance function represented in (2.6), where x and y are the two sequences being compared and $p \in P_{N\times M}$ is the **accumulated cost matrix** computed by the algorithm 1. With the accumulated cost matrix built, the optimal warping path is found by following algorithm 2.

$$dtw(x,y) = c_p * (x,y) = \min_{p \in P_{N\times M}} c_p(x,y) \tag{2.6}$$

**Algorithm 1:** AccumulatedCostMatrix(x,y,c)

$n = |x|$;
$m = |y|$;
$dtw = new[n,m]$;
$dtw_{0,0} = 0$;
**for** $i \leftarrow 1$ *to* $n$ **do**
  $dtw_{i,1} = dtw_{i-1,1} + c(i,1)$;
**end**
**for** $j \leftarrow 1$ *to* $m$ **do**
  $dtw_{1,j} = dtw_{1,j-1} + c(1,j)$;
**end**
**for** $i \leftarrow 1$ *to* $n$ **do**
  **for** $j \leftarrow 1$ *to* $m$ **do**
    $dtw_{i,j} = c(i,j) + \min(dtw_{i-1,j}, dtw_{i,j-1}, dtw_{i-1,j-1})$;
  **end**
**end**
**return** $dtw$

**Algorithm 2:** OptimalWarpingPath(dtw)

```
Input: dtw, the accumulated cost matrix
Output: path, the optimal warping path
path ← new array; i ← rows(dtw); j ← columns(dtw); while (i > 1) (j > 1) do
    if i == 1 then
        j ← j - 1;
    else
        if j == 1 then
            i ← i - 1;
        else
            if dtw(i - 1, j) == min(dtw(i - 1, j), dtw(i, j - 1), dtw(i - 1, j - 1)) then
                i ← i - 1;
            else
                if dtw(i, j - 1) == min(dtw(i - 1, j), dtw(i, j - 1), dtw(i - 1, j - 1)) then
                    j ← j - 1;
                else
                    i ← i - 1; j ← j - 1;
                end
            end
        end
    end
    path.add((i, j));
end
return path
```

## Multidimensional Time Warping

The previous explanation of the DTW algorithm was focused on comparing two one-dimensional sequences, there are however many occasions where the same entity might have multiple different sequences representing it, for example when analysing color in an image which is represented by 3 variables (red, green and blue) each with two sequences of the image from top to bottom and from left to right). According to Shokoohi-Yekta *et al.* [46], there are two simple methods to generalize the DTW algorithm for these multi dimensional use cases, the dependent dynamic time warping ($DTW_d$) and the independent dynamic time warping ($DTW_i$).

The $\text{DTW}_i$ is computed by adding the dtw distance of all dimensions computed separately. This is described in (2.7), where $X$ and $Y$ are two time-series with $M$ dimensions, the $DTW_i$ is the sum of all DTW distances of the $m$th dimension.

$$DTW_i(X, Y) = \sum_{m=1}^{M} DTW(X_m, Y_m) \tag{2.7}$$

As the name suggests, this approach considers each dimension to be independent from the others which is the case in many applications of this algorithm. There is however the case where the dimensions are inherently dependent on each other and therefore we should not assume independence when computing the multidimensional DTW. This mutual dependence is assumed by the $DTW_d$ method, which is computed using the original DTW process but by changing the cost function ($c_{X_i,Y_j}$) to the cumulative squared Euclidean distances of M data points, this is defined in (2.8), where $x_{i,m}$ is the $i$th data point of X and $y_{j,m}$ is the $j$th data point of Y both of the $m$th dimension.

$$c(X_i, Y_j) = \sum_{m=1}^{M} (x_{i,m} - y_{j,m})^2 \tag{2.8}$$

To make both multidimensional DTW methods viable, each dimension must have the same scale, and therefore might be subject to a normalization or standardization process.

## 2.4 Semi-Supervised Learning

On modern data sets, a common problem is labeling data that can be utilized for training future models. This labeling of the data can become expensive since it frequently involves expert knowledge for each individual data point [7]. This leads to many data sets with the majority of their data unlabeled, which is data that might provide helpful information and therefore should not be discarded to train a model. Semi-Supervised Learning [47] is a field of machine learning which focuses on solving these issues. It considers the previously mentioned situation where a data set has a small subset of labeled data and a large subset of unlabeled data. In contrast to a supervised approach, which utilizes only labeled data to train a model, a semi-supervised approach utilizes both labeled and unlabeled data to train and improve the accuracy of its models.

There are numerous different semi-supervised learning approaches, we chose to focus on two techniques which we describe in the following sections.

### 2.4.1 Label Propagation

Zhu *en at.* [51], proposed a model to propagate labels based on node similarity, where the original input labels are fixed and function as sources that extend labels through data that is unlabeled. The label propagation algorithm begins by initializing all of the labels from the labeled subset and the set of unlabeled data. Subsequently, a fully connected graph between each data point is constructed defining the edges by a similarity function. The Python implementation of this algorithm from Scikit-Learn offers two kernels to measure similarity between the data points, a K-Nearest Neighbors (KNN) based kernel and a Radial Basis Function (RBF) kernel. The labels from the labeled subset are then iteratively propagated to each unlabeled data point in the unlabeled subset by taking into account the weighted average label distributions of its neighbors and its current label.

#### K-Nearest Neighbors Kernel

The KNN kernel [28] is a non-parametric kernel used in machine learning mainly for classification and regression problems. It works by assigning to each data point, a label based on a predetermined number of samples with the lowest distance to that data point. All samples can contribute equally to the classification or be proportional to the distance from the point to be assigned. It is defined in scikit-learn [43] using (2.9).

$$1[x' \in kNN(x)] \tag{2.9}$$

The $k$, which is the number of nearest neighbors to take into account for each data point, is given as an input parameter to the model.

#### Radial Basis Function Kernel

The RBF kernel computes the similarity between two data points based on their distance in a specific input space. It is defined in scikit-learn [43] using (2.10).

$$RBF(X, Y) = \exp(-\gamma|x - y|^2), \gamma > 0 \tag{2.10}$$

Where $|x-y|$ is the euclidean distance between x and y and $\gamma$ is given as a parameter to the model. The gamma parameter [35] controls the smoothness of the kernel by defining the influence of each training example. A low value of $\gamma$ results in a wider kernel, which makes the kernel smoother and less sensitive to noise. A higher value of $\gamma$ means a narrower kernel making it sensitive to individual examples in the training set. The RBF most commonly used is the Gaussian RBF ($RBF_G$) (2.11) which assigns the $\gamma$ to $\frac{1}{2\sigma^2}$, where $\sigma$ is the standard deviation of the data.

$$RBF_G(X, Y) = \exp(-\frac{|x - y|^2}{2\sigma^2}) \tag{2.11}$$

### 2.4.2 Label Induced By a Clustered Representation

Chapelle *en at.* [14] proposed the idea of expanding labels onto unlabeled data points by using the cluster assumption, which states that two data points are likely to have the same label if they belong to the same cluster. Many different kernel designs that implement this cluster assumption were also presented. One of the kernel designs presented was an algorithm similar to a Principal Component Analysis (PCA) kernel and involved inducing the label of each data point by a clustered representation. It worked by first computing the affinity matrix using a Radial Basis function kernel and changing the elements of this matrix to 0 instead of 1. Then by representing the sum of the rows in a diagonal matrix, the Laplacian Matrix can be constructed. Subsequently, the first k eigenvalues are found using a process called eigendecomposition on matrix, which can then provide a representation of the data where the points are better clustered and where a simple

classifier can be applied (such as a K-Means classifier). This process will be described (as it is applied in the studied dataset) in Section 4.7.

### K-Means

K-Means [18] is a popular and widely considered simple clustering algorithm available on the python scikit-learn library, that separates data points into $K$ clusters by minimizing their within-cluster sum-of-squares (2.12). It starts by first randomly selecting $K$ points from the dataset as initial centroids and assigns each remaining data point to the nearest centroid, based on the Euclidean distance between the two. After all the points have been assigned to a cluster, the algorithm updates the centroid of each cluster by computing the mean of all the points assigned to it. This process is repeated until the centroids no longer change, or until a maximum number of iterations is reached.

$$\sum_{i=1}^{n} \min_{\mu_j \in C} (\|\mathbf{x}_i - \mu_j\|^2) \tag{2.12}$$

Since the centroid initialization is done randomly, this algorithm tends to provide different results per run. In some occasions, the result might be sub-optimal due to an "unlucky" initialization of the centroids. To solve this issue there is a variation of the algorithm called **K-Means ++** which initializes the centroids in a smart way providing better accuracy and speeding up convergence. This initialization is done [4] using the shortest distance ($D(x)$) from a data point to the closest center already chosen and by following the following steps:

- Step 1 - One centroid ($c_1$) is chosen at random from the data set
- Step 2 - A new centroid ($c_i$) is then chosen using the probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$ where $x$ is a data point in the data set $X$.
- Step 3 - Step 2 is repeated until all K centroids have been selected.
- Step 4 - The K-Means algorithm can proceed with the now initialized centroids.

### DBScan

Density-Based Spatial Clustering of Applications with Noise (DBScan) is another popular clustering algorithm, that uses low-density regions to separate high-density regions in a feature space, clustering the points belonging to the high-density regions. The algorithm has two main parameters, the $\epsilon$, and the min_samples. A cluster core is defined as a data point where there is a number of points higher than the parameter min_samples within a distance lower than $\epsilon$, these data points are called neighbors of the cluster core data point. The data points without enough neighbors will be assigned as noise and will not belong to any cluster.

This clustering approach allows this algorithm to adapt to any shape, simply by following the areas where the distance between the points is relatively low, it also provides the advantage of not having to specify the number of clusters beforehand.

# 3

# Novobanco Data Center

## 3.1 Data Center Structure

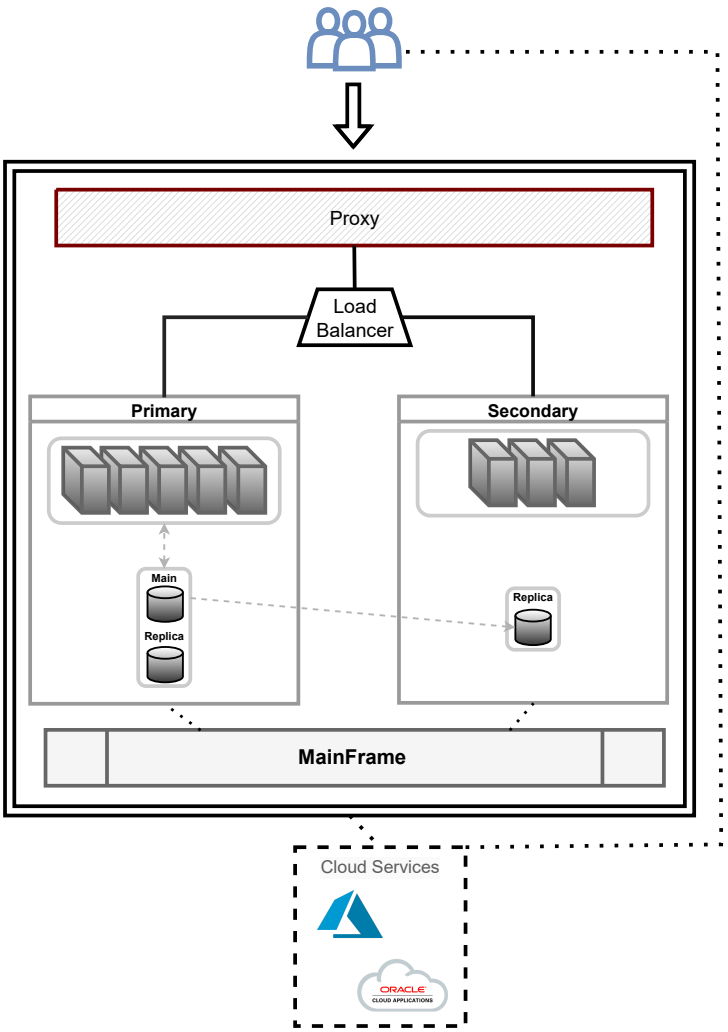



Figure 3.1: *novobanco* Data Center Structure

The structure of the *novobanco* data center, as illustrated in figure 3.1, encompasses both on-premise and cloud solutions. Most machines are managed mostly by external suppliers. The cloud solutions include hosts configured mostly on Azure and Oracle cloud, but different platforms are also being utilized or will be utilized in the future. There is also intercommunication between hosts on-premise and hosts executing in a cloud environment.

All the *novobanco* infrastructure works under the constraints of a closed network. Every request and access from the outside needs to pass through a firewall that filters out most traffic and access is limited to authenticated users with different levels of privilege. Each user interaction with a service goes through a manager that works as a load balancer. The *novobanco* on-premise infrastructure is organized in two sites:

- The **Primary** site: Which is the main location of the data center, the updates on the hosts of this location are then propagated to the secondary site.
- The **Secondary** site: Which is the backup location, the hosts on this location do not run the main applications unless the corresponding hosts in the primary location are down, however, they stay updated and consistent with the main hosts.

The topology of the architecture for each service or application is designed for the specific use case, therefore it is not feasible to give a main overview of how the applications are structured. Inside both data centers the various machines (or hosts) have one or multiple installed services and/or applications. Each application/service can have a different architecture and therefore can have different layers or system specifications.

## 3.2 SAP system landscape

The *novobanco* servers are arranged in a SAP (Systems Applications and Products) system landscape. A SAP system landscape [41] is a 3-tier system with three environments of workflow: **Development**, **Quality**, and **Production**.


Figure 3.2: SAP system landscape phases

- The **Development** environment, includes the code and programs that the development team has installed and is currently using. This environment does not affect in any direct way what the client sees. It can also be used to write a procedure for updating machines in other environments.

- The **Quality** environment is a close simulation of the Production environment. It is normally used to test updates or procedures before the deployment on production.
- The **Production** environment is the environment available to the public, where all the applications and final products are up and running for business use.

All the systems are included in one of these three environments. It needs to be considered that the workload of a machine in the production environment will be different when comparing the workload of a machine in the quality or development environments. A host in the development environment will probably spike randomly during the development periods of an application while a host in the production environment will have a workload that is derived from the real usage of a product/service. A host in the quality environment, on the other hand, will probably spike its usage during testing periods.

## 3.3 novobanco CMDB

In order to manage and configure its services novobanco utilizes a Configuration Management Data Base (CMDB). A CMDB[21] is a centralized database (or a collection of databases) that contains all the information about the components and relationships in a specific infrastructure.

A CMDB includes multiple configuration records[17], each one representing the life cycle of one configuration item (CI) or a relation between CI components. A CI is an element of an infrastructure that can be managed as a single unit, with a distinct lifecycle and identifiable characteristics, it can be any hardware, software, staff, or document from the environment being managed. The intent behind the usage of a CMDB is to track any changes made to the configuration items in a systematic way.

In the case of novobanco, the CIs are the bank infrastructure components and their relationships. The novobanco CMDB tracks the attributes of its infrastructure including host metadata such as the owner of a host, the type of services a host is providing, and other information, some of which will be used in this study.

## 3.4 Monitoring Software - Nagios and Check_mk

*novobanco* utilizes Check_mk as its main monitoring tool, which is software built on top of Nagios.

Nagios [26] is a considerably popular open-source software application used for monitoring and alerting IT infrastructure components, including host resources (such as servers, applications, and services), network devices, and system metrics. It provides real-time monitoring, alerting, and other reporting capabilities and it is well known for helping organizations quickly identify and fix potential issues before they become problematic.

Nagios [27] is highly configurable which means it can meet the specific monitoring requirements of different organizations. It supports a variety of plugins and extensions [25] that allow users to monitor a wide range of services and protocols, including HTTP,SNMP , FTP, SSH, and many others. Nagios also has the ability to notify and alert users in real time when critical events occur (for example, host failures or when a predefined performance threshold is exceeded).

On the other hand, Check_Mk [12, 11] is an alternative solution to Nagios for infrastructure monitoring. Although Nagios was and might still be often considered the standard monitoring tool, as time goes by, it is struggling to meet the necessities of modern systems, since the configuration of these systems tends to be more complex and Nagios is sometimes not sufficiently-scalable. To react to the current dynamic environments, monitoring tools need to be flexible and adjust automatically to new obstacles while being also easily scalable.

Check_Mk aims to solve the previously mentioned issues by firstly, allowing migration of existing Nagios plug-ins, hosts, host groups and users, and also providing the following advantages:

- Scalable monitoring spread across multiple millions of devices, and all these devices are possible to manage from a single central instance.
- A high level of automation, offering features such as the auto-discovery function which, as the name suggests, automatically discovers every single service and recommends metrics and thresholds for all system components.
- Simpler installation on multiple platforms and easier configuration via a web-based graphical user interface.
- A centralized management of all installed agents
- Conceptualized solutions for cloud systems, volatile environments and on-premises architectures

The monitoring use cases for check_mk include servers [13], networks, applications, databases, cloud servers, containers and IoT monitoring, hence its utilization in the *novobanco* server landscape. In the bank data center the monitoring load is shared between multiple slaves. A Check_mk slave is a secondary instance of the Check_mk monitoring software that runs on a separate machine and communicates with the master instance of Check_mk, with the purpose of distributing the monitoring load across multiple servers, allowing the monitoring of a large number of hosts without overloading a single server. The master instance is responsible for coordinating the monitoring tasks of the slaves, by aggregating and processing the data from the slaves to provide a centralized view of the network and infrastructure status.

## 3.5 Round Robin Database

In order to monitor a high number of machines, the novobanco monitoring system needs to store large amounts of data over an extended period of time. However the space requirements of, for example, storing CPU used every 5 minutes can become a bit heavy when the time interval intended for storing the data is 4 years. Considering also, thousands of different machines, the space necessary to store all that monitoring information can become severe.

To solve the previously mentioned issue, the monitoring software utilizes Round Robin Databases (RRDs) for infrastructure and network monitoring. An RRD [38] is a system for storing and retrieving performance data in an efficient manner. It is often used in network monitoring and management systems to store performance metrics and facilitate quick retrieval of historical performance data.

In an RRD [36], the data is stored cyclically in order to maintain its size, when new entries are added, old entries are consolidated or removed. The .rrd files are binary files that have the data points of the same consolidation setup stored into Round Robin Archives (RRA), facilitating the definition of the different data consolidation time intervals for different time periods. For example for the past hour, data measures could be stored every second, for the past 24 hours, the data measures could be stored every 10 minutes, and for the past 2 months, the data measures could be stored every 2 hours, with all consolidations co-existing in the same file. There are 4 type of consolidation functions [39]:

- The **AVERAGE** function, where the average value of a list of collected points is computed
- The **LAST** function, where the last read value is used to represent the whole list of collected points.
- The **MAX** function, where the maximum value is used to represent the whole list of collected points.
- The **MIN** function, where the minimum value is used to represent the whole list of collected points.

By using Round Robin Archives there is the insurance that old data, or data older than the last time timestamp defined, is eliminated. With the consolidation method this format also makes it possible to maintain data for a long time while overcoming space limitations by gradually decreasing granularity and assuring that the size of each file remains unchanged.

Besides the .rrd files, Round robin databases also keep .info files to store metadata information about their structure and configuration. This information can include the names of data sources, the types of data sources, the minimum and maximum values for each data source, the heartbeat and the archive definition.

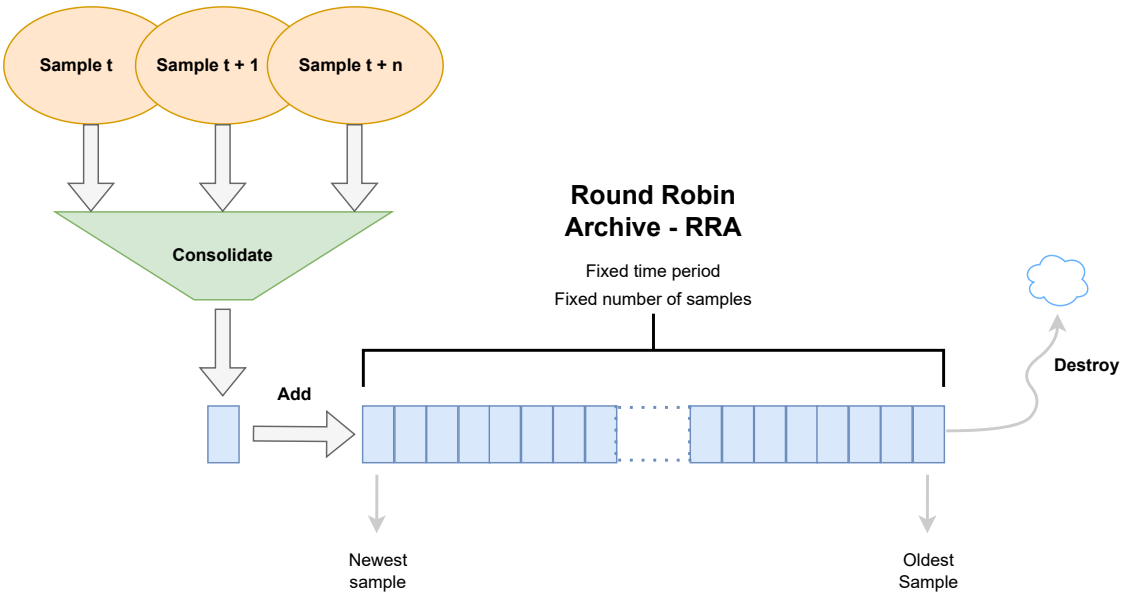



Figure 3.3: Functioning of a RRA archive, adapted from [36]

### 3.5.1 RRDTool

As previously mentioned, .rrd files are in binary format, and although this provides benefits such as a more efficient storage and retrieval of the data, it makes it necessary to use software for their interpretation. The RRDtool [40] is an open-source software application developed by Tobias Oetiker which, besides other functionalities, serves this purpose. This tool is compatible with various operating systems, including Linux, Unix and macOS systems. It can also be used in Windows using third party tools such as Cygwin or MSYS2.

## 3.6 Initial Dataset Overview

The various hosts can have one or many different services or applications installed. Each application/service can have a different architecture and therefore can have different layer specifications.

When considering the size of the entire infrastructure, which included thousands of machines in different environments and locations, we concluded that it was not practical to make an initial analysis of the whole data center. As the decision to which part of the data center had the most benefit in the realization this project was being made, we chose to firstly investigate a set of hosts which were in the production environment and located in the primary data center. These hosts were used as the direct channels of communication with the novobanco customers, being responsible for processing any type of user request (for example, balance checks, credit transfers and others). Although this particular set of hosts did not fully encompass all the intricacies of the entire infrastructure, it provided

a general representation of how the applications are organized as well as their hardware implications. This particular set of machines is of major importance to the company and since the hosts had a considerable amount of workload it was also deemed an appropriate subset to study the correlation between different machine metrics.

In this subset there were four layers specified, the **PL**, **BL**, **SEI** and **BD** layers, each host belonged to one of these layers. The **PL** or **Presentation Layer** is the first layer that contacts directly with the clients, this is the layer where the "front-end" operations are performed. The requests from the clients are subsequently forwarded to the **Business Layer** (**BL** which is the "Back office" layer. The **SEI** layer is simply a layer specifically designed for execution application jobs. Finally, the **Database Layer** (**BD**) comprises the hosts that work as databases for the application.

In the following sections of this chapter, an **Exploratory Data Analysis** (EDA) will be performed based on this subset, where there will be a brief explanation about the technologies used by the bank as well as an analysis of the patterns of the different given metrics.

## 3.7 Exploratory Data Analysis

### 3.7.1 Monitoring Metrics Overview

The initially analyzed monitoring metrics, are specified in the table 3.2, this extraction was made in 09-08-2022. As explained in the previous sections all the data is in RRD format, and the data has time ranges of 2 days, 10 days, 3 months and 4 years as specified in table 3.1. It is again observable that the higher the time-frame the higher the time-step between each entry and the lower the granularity of the data.

| Data Ranges | | | | |
|---|---|---|---|---|
| **Time-frame** | **Time-step** | **Initial date** | **Final date** | **Nr of Entries** |
| Last 2 days | 1 minute | 07-08-2022 | 09-08-2022 | 2880 |
| Last 10 days | 5 minutes | 20-07-2022 | 09-08-2022 | 2880 |
| Last 90 days | 30 minutes | 11-05-2022 | 09-08-2022 | 4320 |
| Last 4 years | 6 hours | 10-08-2018 | 09-08-2022 | 5840 |

Table 3.1: Analysed Round Robin Data ranges

The available Metrics initially analysed are:

| | |
|---|---|
| **cpuAverage** | Measure of the average CPU utilization |
| **ramUsed** | Measure of the Ram used |
| **readThroughput** | Average read throughput of all disks |
| **writeThroughput** | Average write throughput of all disks |
| **readOperations** | Average number of read operations on all disks |
| **writeOperations** | Average number of write operations on all disks |
| **readWaitTime** | Average end to end read wait time of all disks |
| **writeWaitTime** | Average end to end write wait time of all disks |
| **avg_fs_usedSpace** | Average used space on all filesystems |
| **avg_fs_freeSpace** | Average available/free space on all filesystems |
| **avg_fs_size** | Average size of all filesystems |
| **avg_fs_growth** | Average growth on all file systems |
| **avg_fs_shrinking** | Average shrinkage on all file systems |
| **avg_fs_trend** | Average trend of file system growth of all file systems |
| **avg_InputBandwidth** | Average input bandwidth on all network interfaces |
| **avg_outBandwidth** | Average output bandwidth on all network interfaces |
| **avg_input_packets** | Average number of input packets on all network interfaces |
| **avg_output_packets** | Average number of output packets on all network interfaces |
| **uptime** | How much time passed since the last system boot |

Table 3.2: Host Metrics specification

### 3.7.2 Metric Analysis

Since the initial number of metric features was considerably high, it was necessary to filter out some metrics that can be unnecessary to study the usage of a machine. To accomplish that goal we started by identifying all the metrics that were correlated. In order to find these correlated metrics, a correlation matrix was computed for each individual host using the 6 hour time step time series data. Subsequently, a universal average correlation matrix for all hosts was then calculated in order to observe the workload feature correlations that generally exist in every host, which is represented in figure 3.4.

By observing the correlation matrix, and applying a correlation threshold of 0.6 it was possible to determine which metrics were positively and negatively correlated.

As represented in figure 3.5, it is possible to observe a positive correlation between the read/write operations and read/write throughput of the disks, between average file system used space and its trend, between the network average input packets and average input bandwidth and between the network average output packets and the average output bandwidth.

One of each group of correlated features was chosen and the others discarded, all the remaining features were then analyzed individually. In that analysis, in order to avoid missing data we decided to focus only on the two most recent months while still using the 6 hour time-step data, the following analysis will have a time range between June 9 2022 and August 9 2022.

First, we discarded metrics related to the hosts file system since they did not pose

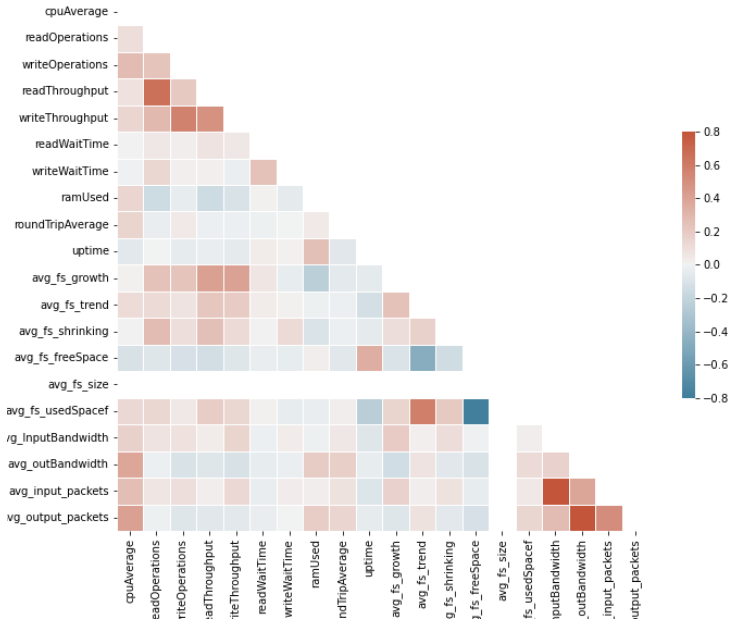



Figure 3.4: Correlation Matrix across all layers

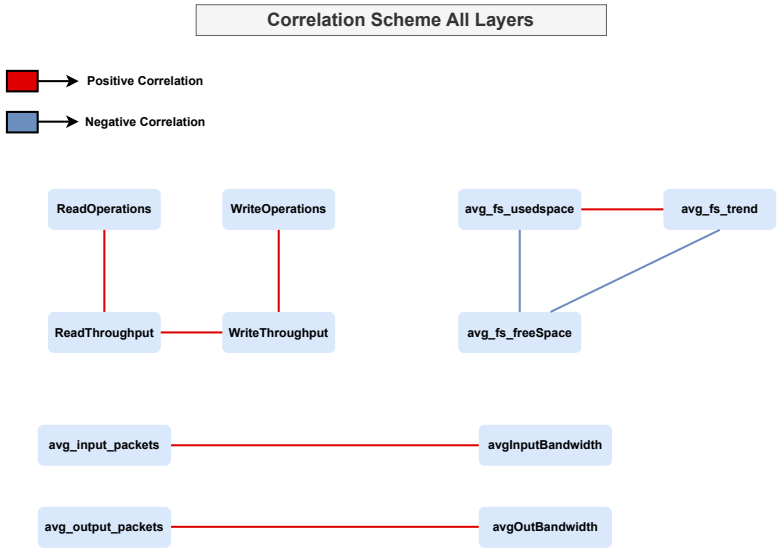



Figure 3.5: The correlated metrics across all layers

any real meaning outside the scope of a particular service or application, these metrics represented the average growth shrinkage, trend and free space of all file systems and did not represent the usage of an host, rather they were entirely dependent on the usage of the file system by the services or applications running on that host, and since each host could have multiple applications they were also dependant on the number of hosted

applications. Following that, we discarded other metrics that were not representative of the workload of the host but rather on external factors such as the round trip average and the uptime. The round trip average was dependant on the current network predicament and the uptime only described the amount of time that passed since the last machine shutdown, which also could be related to errors outside the scope of analysing the usage patterns of a host.

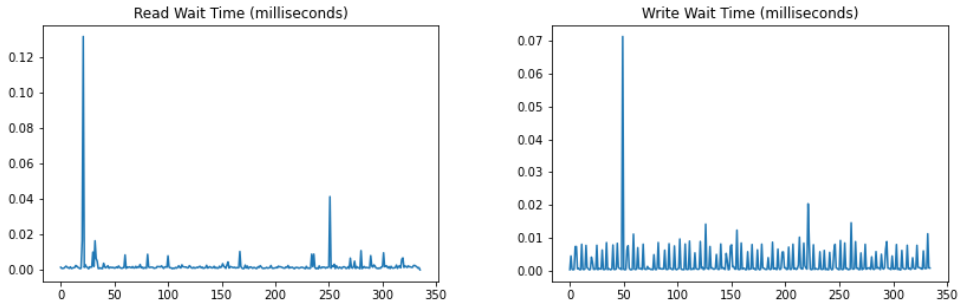



Figure 3.6: Read Wait Time and Write Wait Time monitored values of host pl26

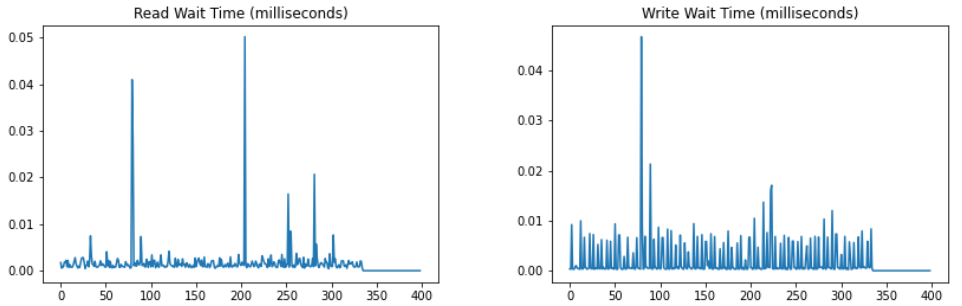



Figure 3.7: Read Wait Time and Write Wait Time monitored values of host bl22

By looking at the remaining metric plots, we noticed that the "Read wait time" and "Write Wait time" features were displaying unreliable values. Both these features, as shown in Figures 3.6 and 3.7, were spiking at the beginning and then stabilizing after a certain point with values so small that it is not possible to observe any changes, which could be related to workload. These features also did not provide a good representation of the workload of each host and were consequently discarded.

### 3.7.3 Layer Comparison

Finally, to finish our initial glance into the novobanco server infrastructure, we did a comparison analysis between the different layers of the data set. The Figures 3.8,3.9 and 3.10 represent respectively the comparison between the cpu usage, memory used and read throughput of all hosts across the four previously presented layers.

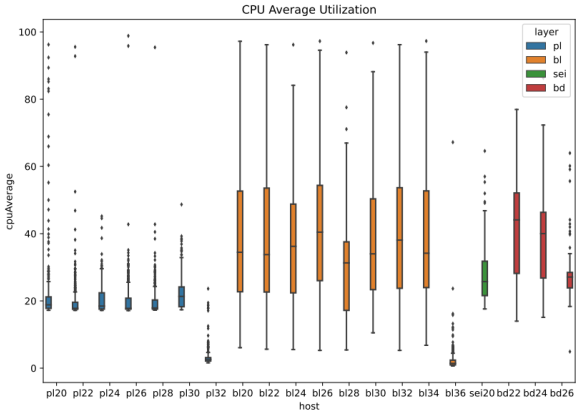



Figure 3.8: CPU Usage of each analysed server grouped by layer

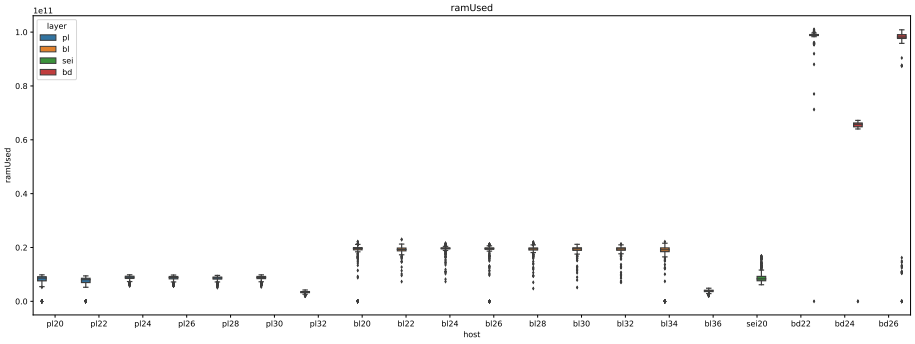



Figure 3.9: Ram Usage of each analysed server grouped by layer

As it is possible to observe there is a noticeable difference when comparing the overall workload between the different layers existent in the analysed service. The CPU usage for example is higher on the bl layer than on the pl layer, which is expected since the pl layer is used as a front-end to this particular application and the bl layer performs back-end operations which tend to be more CPU intensive. We can also observe a higher RAM usage and read throughput on the bd layer, which is expected since this is the database layer that tends to have disk-intensive operations as well as data caching (in memory).

The pl32 and bl36 servers are used only as backup servers, explaining their lack of CPU utilization and lower RAM usage. These hosts are only used when the other servers of a layer are overloaded or if one host went down, since this is a critical application to

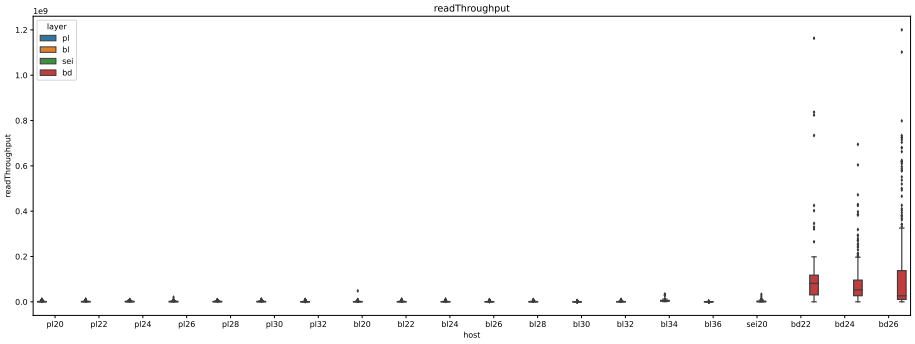



Figure 3.10: Read Throughput of each analysed server grouped by layer

the bank and it is crucial that the machines are not overloaded or have any failures, this service requires backup servers are on the main site.

By observing the behaviour of this application and its different layers, it is clear that the different services assured by multiple hosts will have distinct requirements. It is not optimal to compare, based on only one metric feature, the utilization of a host with another which might belong to a different context within the same application. We can however look at the hosts within the same layer and observe which ones have a higher or lower usage, but it is still incorrect to assert, for example, by observing that the backup hosts are not being used that they should be removed, since the machine workload metrics alone does not show the whole picture of the functioning of an application. Decisions such as improving the hardware of a host, increasing or decreasing the number of backup servers or even the consolidation of multiple hosts to minimize expenditure are taken based on multiple factors, including the needs of an application.

# 4

# Methodology

In the previous chapter, we presented an overview of the novobanco server infrastructure and provided a brief explanation on how its monitoring and managing is performed. As described in Section 3.7, we also analyzed a small subset of machines, that are related to the direct communication channels, in order to study how the services within a *novobanco* application are organized as well as investigate the importance of certain metrics to the representation of the overall workload of a machine.

With the gathered information, we also observed that some hosts that had a reduced usage level (even in the main site) were still necessary to the functioning of an important service, which indicates that the usage of a host alone is not enough to discern how crucial that host is to the overall system and what management decisions can be applied to it. Whether a machine is considered over-provisioned or not will also depend on its specifications or metadata. Often applications that are important for the company require specific hardware, and even though the machines hosting these applications might have a considerably lower usage, a resource readjustment might just not be advisable.

As novobanco shifts further towards cloud solutions, some services hosted on on-premise hosts are periodically migrated to the cloud. These service migrations are also decisions where the workload has some impact but the specifications of the machines also play a major role. The hardware itself is often not compatible with the particular cloud service making the migration not feasible without a major update to the application in question, often making the effort higher than the benefit of such changes. These are some challenges of managing on-premise hardware, where specific approaches need to be designed to deal with each event or decision.

In this chapter, we present our solution to achieve a decision support system that has the objective of easing the resource management of the *novobanco* infrastructure. We address the problem of adjusting on-premise resources, focusing on identifying machines that might be over-provisioned and recommending decisions based on domain knowledge.

## 4.1 Proposed Solution Overview

In order to identify machines that might be over-provisioned we create a knowledge base by extracting and combining host data from multiple sources and propose a model management system prototype which, by retrieving previously taken management decisions can recommend new ones using semi-supervised machine learning approaches.

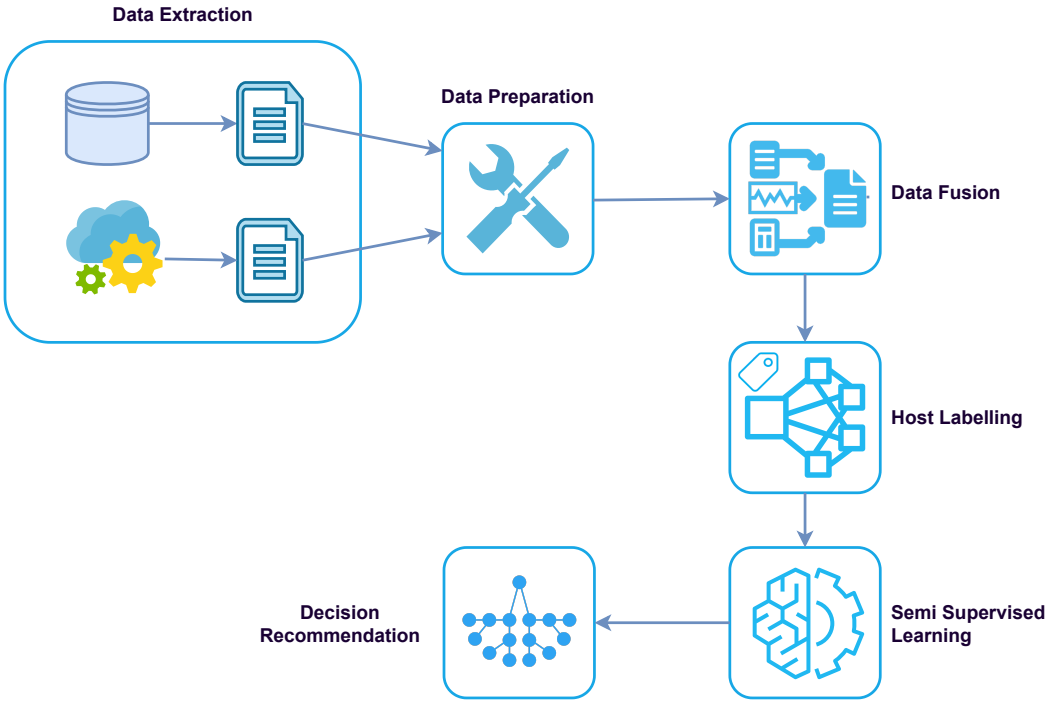



Figure 4.1: The Methodology process

The whole process is depicted in Figure 4.1 and involves the following steps:

- In the first step, it was necessary to extract both the workload and metadata of the hosts. For this, we designed a way to automatize the extraction of the data from check_mk and explored the reporting system of the *novobanco* CMDB.
- In the second step, we prepared the data for future analysis, this involved the filtering of some hosts and features as well as choosing a good time frame to represent the workload of most infrastructure. Part of this step can be mapped to the first step (**feature extraction**) of a **Decision-Level Data Fusion** framework.
- In the third step, we explored ways to combine the host's metadata with their workload metrics, this step involved an **intermediate analysis** of the extracted features which provided the results for a **data fusion** process.

- In the fourth step, with the help of the *novobanco* staff we identified hosts which were over-provisioned and therefore were in need of a resource re-adjustment as well as hosts where such adjustments were not viable. These hosts will serve as labeled data for the model management system.
- In the fifth step, utilizing the previously labeled hosts, we explored semi-supervised approaches to expand those labels to the entire data set.
- Finally, after validating the results from the fifth step, we employed a decision support tree based on heuristics that were discussed with the *novobanco* staff, which provided the output results of our decision support system in the explored on-premise data set.

## 4.2 On-premise Dataset

As previously implied, this study is designed to encompass the entire *novobanco* server infrastructure. However, due to constraints related to storage, security, and time complexity of retrieving data from thousands of machines, we decided to select a portion of all the on-premise hosts. This partial dataset was chosen in collaboration with the *novobanco* staff, where machines with older software were dominant, with the aim of maximizing the initial benefit of this project for the company.

The provided dataset, which consisted of 644 hosts, was deemed more likely to have machines with lower usage and where the reduction of hardware and consolidation of different hosts would prove more advantageous. It included hosts in all three SAP environments, hosts with various different domains as well as hosts in both site locations (Figure 3.1). It also encompassed multiple types of services being assured by the hosts, including database services, web services, application services, and infrastructure services. These services were considered legacy, and will most likely have to be updated, but the decision-making about the resource management of all hardware is done in a non-automated way. As a consequence, the hosts that are used to run legacy applications tend to have less urgency and attention by the staff, hence the benefit of this study in this particular infrastructure.

## 4.3 Data Extraction

### 4.3.1 Data Sources

As described in Chapter 3, *novobanco* stores system information about its data center hosts (or host metadata) in a configuration management database (**CMDB**). This information allows *novobanco* to consult and identify in detail the "category" of a host. Various decisions and evaluations about some hosts are often based on their specific "category". Using the previously given examples, when considering if an application service is a

viable candidate to migrate to the cloud it is important to take into account if the version of the operating system that runs that service is compatible with that specific cloud solution, or when a host starts to have a lower utilization it might be of interest to move the application services running in that host to another categorically similar underutilized host, consolidating both machines into one, in order to reduce unnecessary expenditure. Without the information about the site for example, it is not reasonable to identify hosts as over-provisioned based on their workload, since these hosts might be on the backup location and their purpose might be assuring geo-redundancy.

To retrieve the **Hosts Metadata** information, two reports were extracted from the configuration management database platform. They both had some matching information, however, one was an **applicational report** that included information such as the operating system version of each host, as well as its installed applications, and the other was a **system report** that added information such as the services that were assured by each host (for example, if the host had the capability of being a database server and/or a web server). Combined, these two reports included all the available metadata.

Besides the extraction of the hosts metadata, it was also necessary to retrieve the hosts workload information in order to study and analyze their usage patterns. For most decisions taken on the *novobanco* infrastructure, the workload pattern of a host is an important factor. For cloud migrations, it's necessary to consider if the workload of the applications which are being considered for migration are financially suitable for a cloud environment (2.1). For the resource readjustment decisions, it's also important to consider the host's workload since it is based on it that it is possible to identify overloaded or under-utilized infrastructure as well as infrastructure that could be merged for efficient hardware usage, for example, low machine workloads that peak at different times (such as machines that generate reports in different days of the week).

This workload information was extracted from check_mk using RRDTool using an iterative process that will be explained in Section 4.3.2.2.

### 4.3.2 Workload Data

#### 4.3.2.1 RRD conversion issues

Before tackling the process of extracting all the workload data from the given dataset, it is important to introduce the obstacles encountered related to the conversion of the .rrd files into an interpretable format.

As previously explained, .rrd files are stored in binary format and to interpret these files the RRDTool is necessary. By using this tool it is possible to run the command **rrd dump** [1] which can convert .rrd files to XML (Extensible Markup Language) format.

However, since the local post computer was only authorized to have installed Windows 10 as its operating system, and tools such as cygwin are also not authorized by the security norms of novobanco, it was not feasable to run the rrdtool directly on the local post computer where the .rrd files could then be converted.

To overcome this problem, privileged access management (PAM) to a linux server was given by novobanco, and the rrdtool was installed onto that server. This server was then used as a file conversion middleware between the check_mk main server and the local post.

Two issues then emerge:

- The xml files are much heavier when comparing to the original .rrd format, which meant that the given linux machine cannot hold all the converted extracted data simultaneously.
- The machines from check_mk can become overloaded when asked to pull multiple rrd files from its instances, explaining also the necessity for a second server only responsible for the conversion of the files.

To work around these obstacles, the extraction process was done in multiple partial iterations, as will be explained further.

#### 4.3.2.2 Workload Data Extraction Process

The extraction of workload data from check_mk involved multiple steps. Firstly, since the monitoring task is shared between multiple slave instances, we matched the hosts of the given dataset with the slave instances responsible for their monitoring. Subsequently, on the monitoring master instance, an internal shell script was created to pull the .rrd and .info files correspondent to each metric service and send those files to the Linux machine. The extracted metrics services included:

- The **CPU** service, which included the usage percentage of each cpu core.
- The **Memory** service, which included information such as the memory allocated and used (in kilobytes)
- The **Disk Summary** service, which encompassed the information of all the disks, included the previously mentioned disk metrics (average disk write/read throughput, number of IO operations and read write queue lenght).
- The **Interfaces** services, which had network information about the host including the bandwidth throughput of each host.

After sending all the .rrd files and .info files of each host to the linux machine, the extracted data was divided into four parts which resulted in 4 iterations of the next steps. Subsequently, a metric conversion shell script was run on each part of the extraction, which converted all the .rrd files to XML using the **rrd dump** command. This script also compressed .xml files to a zip format in order to save storage space.

Each zip file was imported to the local post computer using a PAM client, and a python script was created to convert the XML files to .csv format using the correspondent .info file

as the service metadata. In order to speed up this final conversion, we chose to only utilize the .rrd data which had 4 years as the time-frame and each timestamp corresponding to the average of the last 6 hours (last row in 3.1), allowing for the utilization of wider ranges of data while reducing the number of data points in each host and as consequence reducing the complexity of running future algorithms. This granularity of data also has the advantage of smothering momentaneous spikes of usage which was assumed as beneficial when estimating the overall usage pattern of a machine.

After this final conversion, the data could then be easily interpreted.

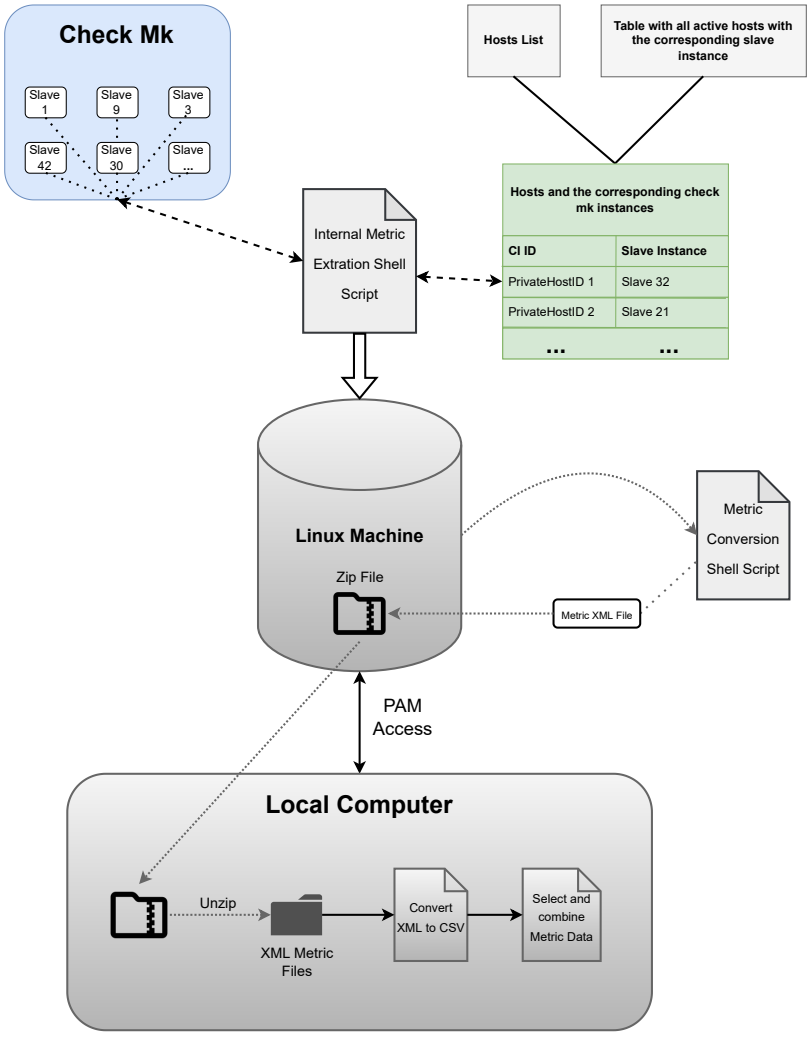



Figure 4.2: Workload Metrics Extraction process

## 4.4 Data Preparation

### 4.4.1 Data retrieval from the Service Files

After extracting and converting the .rrd files corresponding to each service to .csv format, it was necessary to select and retrieve individual features of each host. The selected features and the method used in their acquisition were the following:

- **CPU Usage (%)** - Obtained by averaging across all CPU cores available on the CPU usage service file of each machine.
- **Disk Write Throughput (bytes/s)** - Obtained by directly taking the values out of the disk write throughput column on the Disk Summary service file of each machine
- **Memory Used (bytes/s)** - Obtained by directly taking the values out of the memory used column on the memory service file of each machine.
- **Network Input bandwidth (bytes/s)** and **Network Output bandwidth (bytes/s)** - Each obtained by taking the sum of the bandwidths of all the interface service files of each machine.

All the selected features were then joined onto a single file matching the timestamps of each data point.

### 4.4.2 Data Filtering

While selecting and joining the different metric features, some issues became clear:

Firstly, some hosts did not have all the metrics available, the majority of these were just infrastructure hosts where check_mk did not record anything besides network information. There were also a few hosts that were turned off and therefore their historical data could not be pulled off. These hosts were then filtered out of the data set since machine learning models tend to not react well to that much missing data and it would be difficult to conclude anything without the majority of the metric features.

After filtering out the hosts with missing metrics, we analyzed the remaining historical metric data further and noticed that there were also periods with missing data that did not coincide on every single different host. Besides the fact that not every single host was created at the same time which did not pose an issue since we were not interested in data older than 2021, a considerable amount of hosts had missing data which ranged from days to months (Figure 4.3).

In order to maximize the number of reliable hosts (without great portions of the missing data) and to keep a time range long enough to be a reliable representation of the usage of each host, we chose to use the data from 1st of January 2022 to the 1st of march 2022. Two months of data was deemed enough to represent the workload of a machine since most novobanco usage cycles were, based on staff knowledge, at a monthly

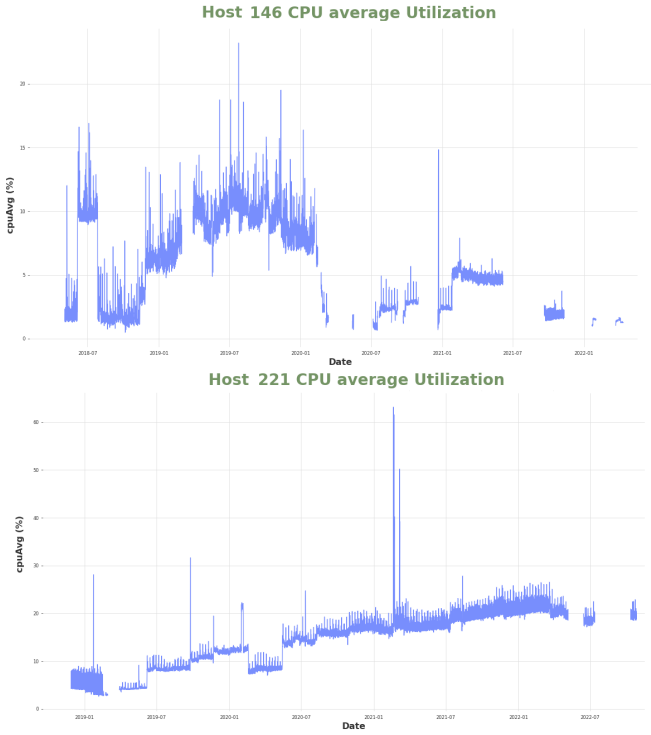



Figure 4.3: Examples of 2 hosts with missing data

basis (some at a yearly basis as well, but, as previously discussed, using an entire year of data was not a viable option). These months had also the least amount of hosts with long periods of missing data and they represented a time period recent enough to take up-to-date conclusions. Another advantage of using data from the beginning of 2022 instead of newer information was that if some machines were, in the meanwhile, turned off or had an hardware resource reduction, they could be used as labelled data for our future models.

Considering now only the detailed time-frame, we proceeded to remove the hosts which had more than 25% missing values of cpu usage, the final filtered data set of hosts included in total **455 hosts**.

Besides missing data, both the input and output bandwidth values of the sum of all the network interfaces were registering long periods with zero values. This was especially prevalent on hosts which were on development or quality environments, although it was also noticeable on some hosts on the production environment. For this reason, we chose to exclude the network bandwidth values and not use them for the future steps.

### 4.4.3 Metadata Handling

Aside from the retrieval and selection of the workload data, the metadata had to be handled as well. Using both reports from CMDB we selected the following features:

- The **Operating System** - Which corresponds to the name and specific version of the operating system which runs on the host
- The **Environment** - Which corresponds to the SAP environment of the host (which can be development, quality or production.
- The **Domain** - Which corresponds to the domain name of the host which can be within the novobanco data center or within a cloud service.
- The **Site** - Which corresponds to the location site of the host, as previously stated novobanco has two on premise sites.
- The **Service capabilities** - Which is a list of provided services by the host to its applications (most hosts were providing one single service)

After selecting and retrieving these features we proceeded to join them using the host's identifier (**CI ID**) and created our initial categorical set which had to undergo some alterations.

The machine learning models as well as the data fusion procedures which will be utilized, require all the variables to be numeric, as a consequence, in order for the categorical set to be usable, we had to encode each feature. For this, we chose to utilize **Dummy Encoding**. For each k categories in the **Operating System**, **Environment**, **Domain** and **Site** columns, we created k - 1 features. For the **Service capabilities** column we created one feature per existent service in the data set since it is possible for the same host to have multiple service capabilities. With this, we created a binary set for each host containing all categorical data, that can be used in the next step.

## 4.5 Data Fusion

As previously stated, decisions made about the hardware infrastructure are based on both the usage and the category of the hosts. Hosts with similar usage patterns might have different domains or operating systems and, consequently, cannot be analysed the same way. The opposing scenario is also true, as hosts within the same category which have drastically distinct usage patterns do not have similar behaviour and therefore will likely have different management rules applied to them.

Having now both the workload metric features and the metadata features of each host extracted and handled, the goal now was to utilize both forms of data in a way to represent a host and compare it with the rest of the infrastructure.

Recalling now we had a binary encoding of metadata for each host as well as three time series of the previously mentioned workload metrics, as shown in Figure 4.4.

As a first approach to combine these mixed features and measure similarity between the various hosts within the data set, we represented each timestamp of each workload

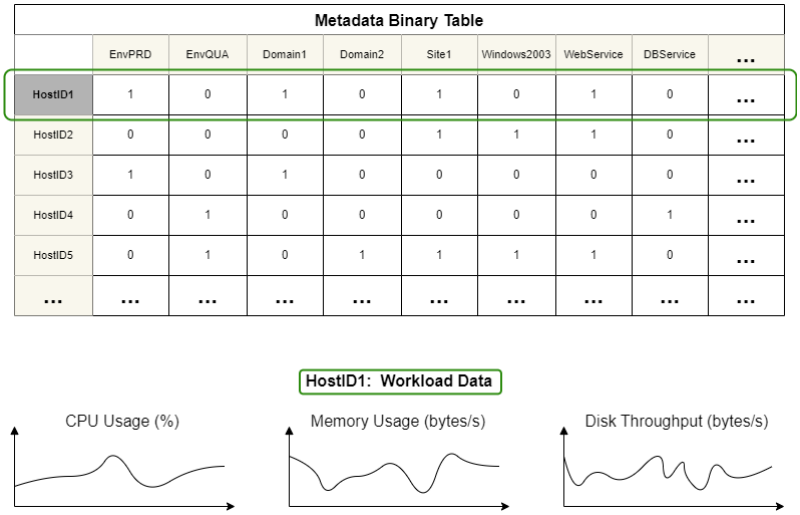



Figure 4.4: Representation of the data for each host

feature as a separate column, and combined all the workload columns with the binary columns of the hosts metadata (Figure 4.5).

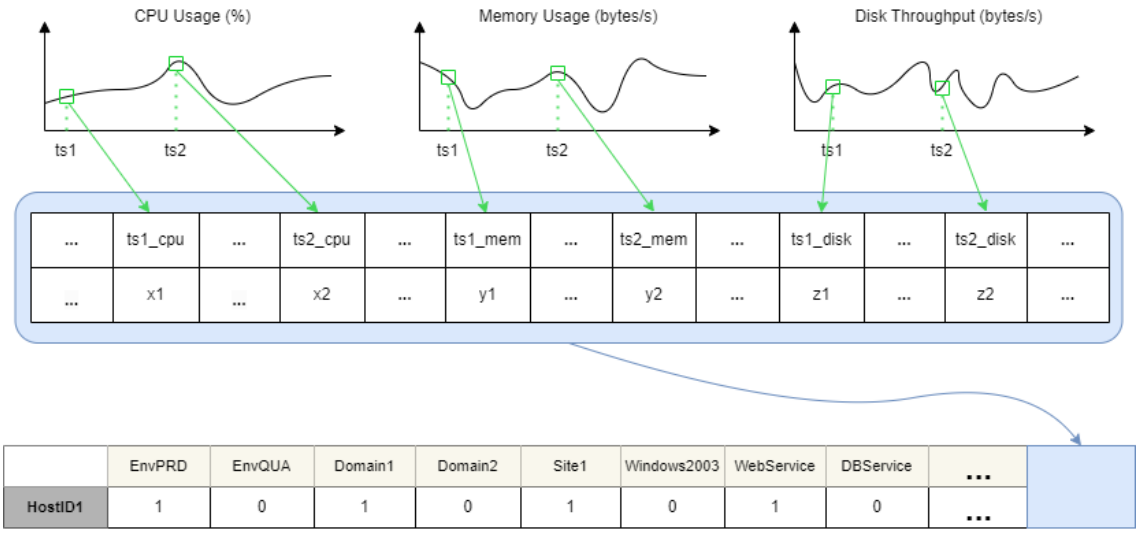



Figure 4.5: First Data Fusion approach representation

With this, each host was now represented by a entry with mixed data, which included binary values correspondent to the host metadata and numerical values correspondent to each timestamp of each host workload metric. The numerical values however, were expressed with different units and consequently had different proportions, so we had to first z-normalize these values so all features shared a similar scale. To do this we took the mean ($\mu$) and standard deviation ($\sigma$) of each workload metric ($w$) in the entire data set and applied 4.1 to each metric time series for all hosts.

$$S'_{w,h} = \frac{S_{w,h} - \mu_w}{\sigma_w} \tag{4.1}$$

$S_{w,h}$ represents the time series of the workload feature $w$ from host $h$, and the $\mu_w$ and $\sigma_w$ represent the mean and standard deviation values of feature $w$. By applying this z-normalization scaling to each metric time series we can preserve its structure and pattern [50].

## Computation of the Gower Distance Matrix

Using now the previously explained representation, we calculated a Gower distance metric for each pair of hosts, taking into account both numerical and categorical features. For the binary (categorical) values we used the Hamming distance to measure the categorical dissimilarity between each pair of hosts described in (4.2), where $n$ is total the number of features (both numerical and categorical), $x$ and $y$ are the two host categorical rows being compared, $d_i$ is the number of distinct (dummy encoded) categories for the $i$-th feature, and $1_{x_i \neq y_i}$ is the indicator function that equals 1 if $x_i \neq y_i$ and 0 otherwise.

$$d_H(x,y) = \frac{1}{n} \sum_{i=1}^{n} \frac{1_{x_i \neq y_i}}{d_i} \tag{4.2}$$

For the numerical values related to the time series of each metric, we initially used the Manhattan distance to measure the workload dissimilarity, as represented in (4.3), where $n$ is the total number of features, $x$ and $y$ are the two host numerical rows being compared, $r_i$ is the range of the $i$-th feature, and $|x_i - y_i|$ denotes the absolute value in the $i$-th column.

$$d_M(x,y) = \frac{1}{n} \sum_{i=1}^{n} \frac{|x_i - y_i|}{r_i} \tag{4.3}$$

The Gowers distance measurement between each host was then computed by adding these two distances together, resulting in a Gowers distance matrix. However, this first approach had some issues which resulted in unreliable results:

- The first issue was that we were not handling the workload data as a time series, we were instead treating each timestamp as a separate feature, which meant that two hosts that had similar workload patterns but that did not coincide in time were considered as exceedingly different.

- The second problem with this approach was the assumption that all workload features were independent from one another instead of representing the workload of the hosts as a whole.

## Incorporation of the Dynamic Time Warping algorithm

To overcome the previous issues and improve upon the former Gower distance implementation, we chose to abandon the usage of the Manhattan distance for the numeric values correspondent to each timestamp of each workload feature and instead utilized the **Dynamic Time Warping** (**DTW**) algorithm.

As explained in 2.3, by utilizing Dynamic Time Warping its possible to find an optimal alignment between two time series, solving the issue where similar patterns not matched in time were producing larger dissimilarities.

To solve the second issue explained, instead of dealing with each time stamp of each metric as an independent separate variable, all the metric features were combined into one multidimensional series where at each timestamp there is a 3-dimensional point with the CPU utilization, memory usage and disk throughput (all scaled by z-normalization) correspondent to that timestamp.

We then computed the dependent version ($DTW_d$) multidimensional variant of the dtw distance, on each pair of hosts, constructing our DTW distance matrix. Utilizing the independent variant of this algorithm ($DTW_i$) by adding the DTW distance of each time series workload feature separately would also treat each of these features independently, which would not solve our second problem described.

## Final Distance Matrix

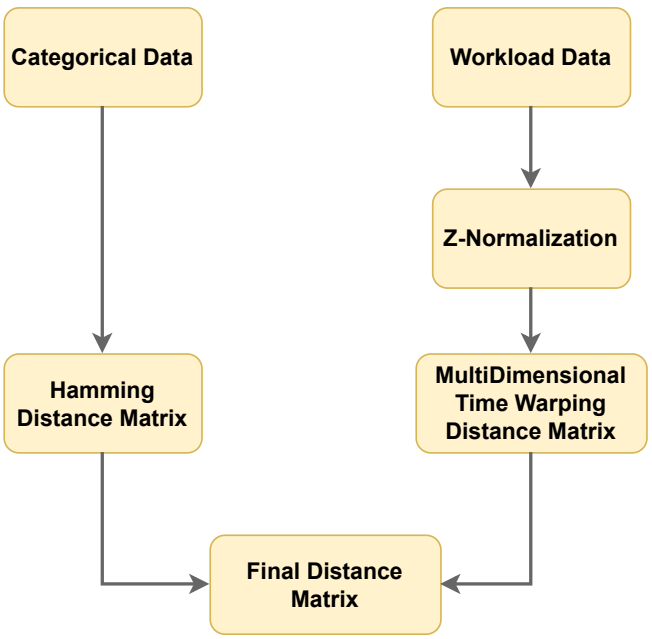



Figure 4.6: The Final Distance Matrix Computation Process

To compute the final distance matrix which we will use for the semi-supervised step, we combined the previously computed hamming distance matrix with the now computed DTW distance matrix (Figure 4.6). This computation is described in (4.4) and (4.5).

$$GowersDistanceMatrix = \frac{\mu}{\rho} HammingDistanceMatrix + \frac{\nu}{\rho} DTWDistanceMatrix \quad (4.4)$$

The $\rho$ corresponds to the total number of features in the data set, and the $\mu$ and $\nu$ correspond to the number of features represented in each matrix.

By multiplying each distance matrix by the number of features represented by that matrix and then dividing by the total number of features we assure that all individual features are given the same importance. Applying now this equation to our case study:

$$GowersDistanceMatrix = \frac{5}{8} HammingDistanceMatrix + \frac{3}{8} DTWDistanceMatrix \quad (4.5)$$

We considered each individual metadata column before the dummy encoding process as well as each workload metric a as an individual feature. As we can observe this method assigned more weight to the distance matrix that was representing more dimensions, in this case the hamming distance matrix, which consequently prioritizes the metadata of the hosts to the final distance computation. We are assuming all features to be equally important, which we can not confirm to be the case, but it is reasonable to assume that the categories of a pair of hosts contribute more to their similarity (or dissimilarity) than their workload. To observe the results (from both approaches), we utilized the **DBScan** clustering algorithm, adapting the distance ($\epsilon$) to observe if the hosts being grouped were reasonable.

## 4.6 Host Labelling

After the data fusion step, we had to find labels in the data set so that future machine learning models could be applied. Since the time of extraction, we noticed that five hosts had been turned off and had their workload assured by other machines, these hosts could then be utilized as labeled data, but five labels were not enough to viably employ a semi-supervised model. So, to get more labeled data, we had to request the *novobanco* staff to identify hosts which could have their hardware reduced or be consolidated with other hosts within the same category. We also needed labels of hosts, which, due to a higher workload and/or their category, could not suffer such reductions.

As a way to assist the staff in identifying these hosts, we grouped all machines by their category while displaying all their metadata details. This helped the staff immediately identify hosts that could not suffer any alteration because they were critical to the company, belonged to a particular portion of the data center that could not be altered, or provided unique periodical services that could not be substituted easily. We also asked

for pairs of hosts within the same category, where one could not suffer any reduction due to a higher workload and another which had a lower workload could be considered as over-provisioned and therefore could be marked for such reductions.

A total of 30 labels were manually identified in collaboration with the *novobanco* staff, 15 hosts were labeled as over-provisioned (label 1), and the other 15 were labeled for the opposite, where the hosts could not suffer any alteration, or were in fact under-provisioned (label 0). These labels were spread across 19 categorical groups, each group having at most two labeled hosts. By doing this, we achieved some variety in the data set, and it will be possible to test the algorithms with a balanced distribution of both labels. These labels will serve to simulate user interaction with our model management system, where the users input (either manually or automatically) their made assumptions about a portion of the infrastructure.

## 4.7 Semi-Supervised Learning

Having now our initial labeled subset of hosts, we began to explore ways to expand the labels onto the entire data set. In this section, we describe the implementation of a semi-supervised procedure that was adapted from [14], and that will function as our model management system.

Firstly, we took the computed distance matrix and applied the radial basis function (RBF) kernel described in (4.6), where $\mathbf{x}$ and $\mathbf{x}'$ represent two data points in the distance matrix, and $\sigma$ is the standard deviation of the distribution of all pairwise distances.

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{4.6}$$

This function converted the distance matrix to an affinity matrix (Figure 4.7).

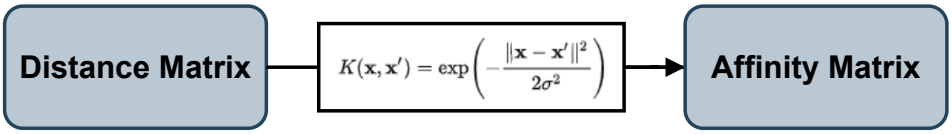



Figure 4.7: Conversion of the distance matrix to an affinity matrix

After this conversion, we replaced the diagonal values of the matrix from 1 to 0 and computed the sum of all rows within the affinity matrix, which we represented in a diagonal matrix.

We computed the difference between the diagonal matrix (D) and the affinity matrix (A) and obtained a Laplacian matrix (L) ($L = D - A$). Then, by eigen decomposing the Laplacian matrix, we obtained its eigenvalues and eigenvectors. Subsequently, we sorted

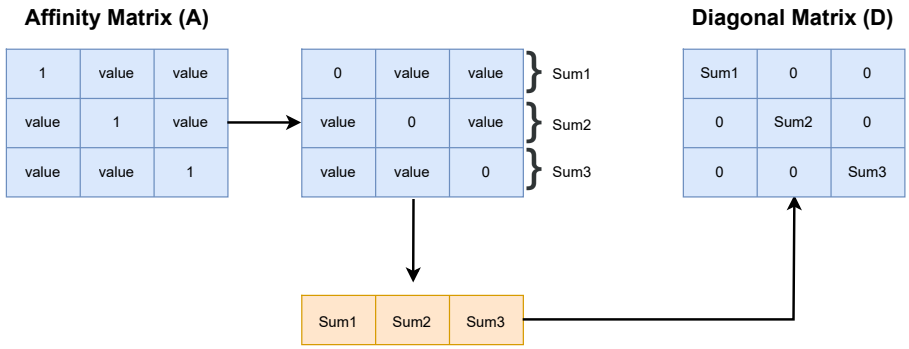



Figure 4.8: Computation of the diagonal Matrix

the eigenvalues in ascending order and applied the same order to the corresponding eigenvectors (as represented in Figure 4.9).

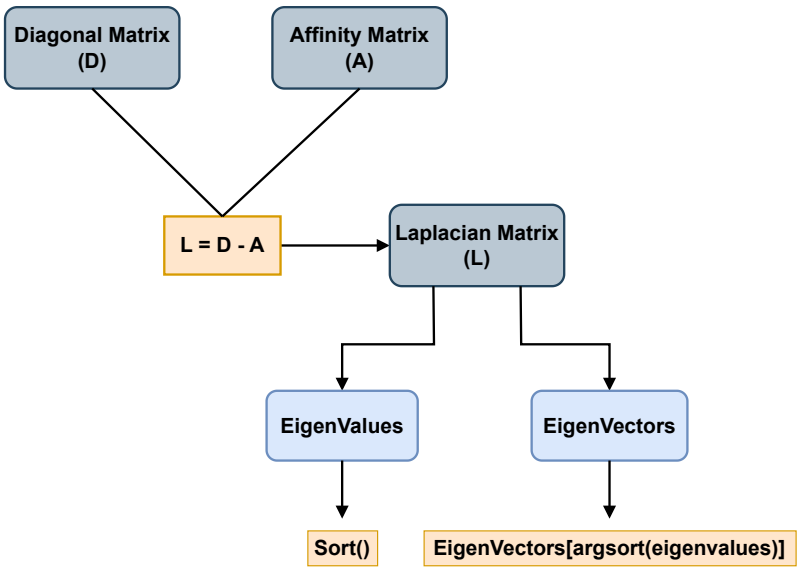



Figure 4.9: Laplacian Matrix Computation and Eigen Decomposition

As suggested in [14], we can use the eigenvectors to capture the structure of the data and identify its patterns in a more flexible way. In Figure 4.10, we present the ordered eigenvalues (in a linear scale).

In [14], it is suggested to use the first k eigenvectors (correspondent to the first non-zero eigenvalues) in order to represent a lower dimensional embedding of the data, which can be used to cluster the data points based on similarity.

However, looking at figure 4.10, we notice that a significant portion of the eigenvalues are very close to zero, indicating that some of these first eigenvalues might represent

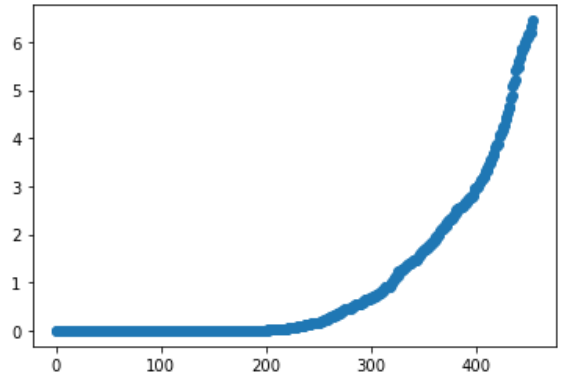



Figure 4.10: Ordered Eigenvalues in Linear scale

noise. By finding the eigengap, it is possible to separate the noise eigenvalues (or the zero eigenvalues) from the first eigenvalues, which provide the best separation of the data.

In order to find the eigengap, we represented the eigenvalues in a logarithmic scale (Figure 4.11).

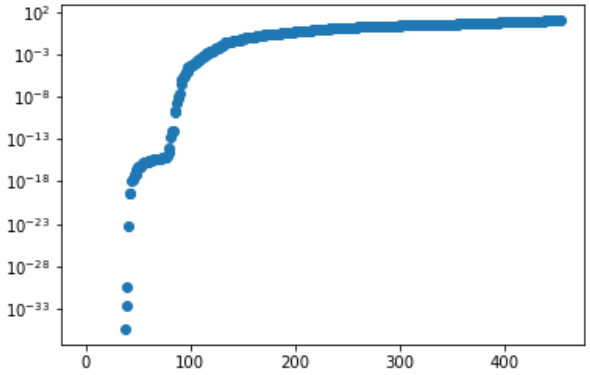



Figure 4.11: Ordered Eigenvalues in a Logarithmic scale

By observing Figure 4.11, we identified two possible eigengaps (represented in figure 4.12), from which we needed to choose the one that provided the best separation between the noise and the first eigenvalues.

To then make the choice between the first and second eigen gaps, we took the first two eigenvectors correspondent to the first two eigenvalues after each gap, and, by mapping those eigenvectors onto the distance matrix, we made two separate bi-dimensional representations of the data, which are displayed in Figure 5.2

As we can observe, the first two eigenvectors corresponding to the two eigenvalues after the second gap produced a clearly better separation of the data. Consequently, the k vectors after this gap will be the ones taken into account for future methods. We decided to test 2 solutions to propagate the labels onto the remaining hosts, the **Label propagation** algorithm and a custom approach using the **K-Means ++** clustering algorithm where

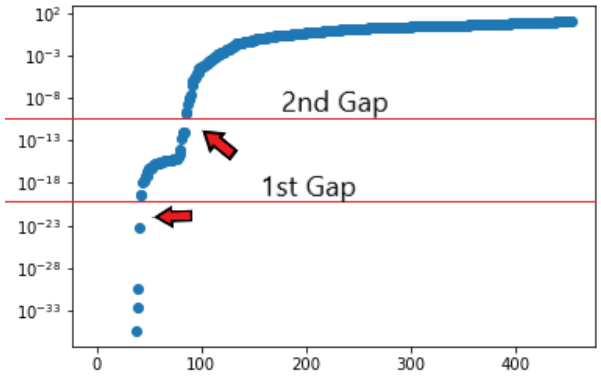
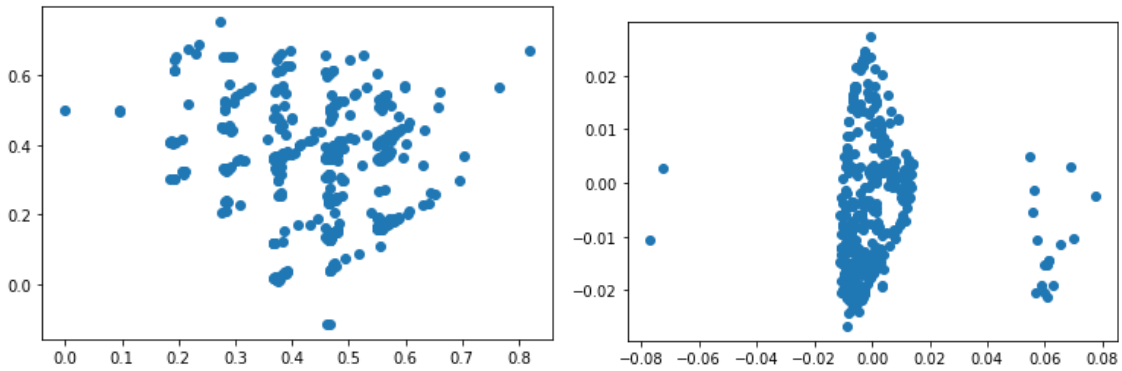



Figure 4.12: The two identified possible eigengaps



(a) Bi-dimensional representation after the 1st gap (b) Bi-dimensional representation after the 2nd gap

Figure 4.13: Bi-dimensional representation of the distance matrix using both possible eigen gaps

we induce the labels based on a majority vote on the clustered representation. For both solutions, we tested using a different number of dimensions (**ndimensions**), by varying the number of k vectors after the 2nd gap, we experimented using between 2 to 30 dimensions. For each number of dimensions, we utilized the precision and recall measures to discern which parameter values of each function provided the best F1 score. Per each **ndimensions** value, for the **Label Propagation** algorithm, we tested 100 values for the $\gamma$ parameter spaced evenly in a logarithmic scale between 0.001 and 100 and chose the $\gamma$ value that provided the best F1 score.

For the **Kmeans ++** clustering algorithm, we performed a similar action but for the *n_clusters* parameter where we tested using between 2 to 60 clusters. We utilized the K-Means ++ algorithm to improve the initialization of the centroids, however, the algorithm still has a varying nature and will produce different results on different runs. To try to mitigate this issue and discern the number of clusters that provided the best F1 Score consistently, we ran the K-means algorithm ten times for each *n_clusters* parameter and computed the average F1 Score of those 10 runs.

The precision and recall measures correspondent to the previously mentioned chosen parameters are represented in Table 4.1.

Table 4.1: Best F1Scores for the Label propagation and K-Means algorithms per number of dimensions

| nDimensions | **Label Propagation** | **K-Means ++** |
|---|---|---|
| 2 | 0.6666 | 0.7866 |
| 3 | 0.6428 | 0.7855 |
| 4 | 0.7200 | 0.7582 |
| 5 | 0.7200 | 0.7813 |
| 6 | 0.7200 | 0.8020 |
| 7 | 0.7200 | 0.8208 |
| 8 | 0.7200 | 0.8471 |
| 9 | 0.7692 | 0.8374 |
| 10 | 0.7857 | 0.8513 |
| 11 | 0.8666 | 0.8212 |
| 12 | 0.8750 | 0.8130 |
| 13 | 0.8965 | 0.7931 |
| 14 | 0.8965 | 0.7964 |
| 15 | 0.9285 | 0.8072 |
| 16 | 0.9285 | 0.8372 |
| 17 | 0.9285 | 0.8536 |
| 18 | 0.9655 | 0.8529 |
| 19 | 0.9655 | 0.8427 |
| 20 | 0.9655 | 0.8327 |
| 21 | 0.9655 | 0.8401 |
| 22 | 0.9655 | 0.8505 |
| 23 | 0.9655 | 0.8558 |
| 24 | 0.9655 | 0.8448 |
| 25 | 0.9655 | 0.8587 |

As we can observe, the label propagation F1 measures stagnate after more than 18 dimensions and the K-Means algorithm F1 measures stay fluctuating making them unreliable to choose the best number of dimensions. This occurs due to the nature of this approach, besides the variance introduced by the different centroid initializations, as we increase the value of the *n_clusters* parameter the F1 score tends to be higher. However by increasing this parameter the problem of over fitting also arises and the algorithm might not expand the labels onto the whole data set in a meaningful way.

We chose to utilize 18 dimensions (correspondent to 18 vectors after the chosen eigen gap), since this is the minimum number of dimensions which provide the best F1 score for the label propagation algorithm, which as previously mentioned was producing stable results. And now we must choose a value for the *n_clusters* parameter in the K-Means clustering that provides good results while avoiding over fitting. To do this we utilized the purity measure. Using the 18 dimensional representation of the distance matrix we measured the average purity (over 10 runs) of the labeled data for each *n_clusters*

as represented in Figure 4.14. We observed that the purity measure starts to stabilize around the 25 to 30 *n_clusters* values. We chose to use the *n_clusters* value that had the best average purity measure and that was lower than 35 in order to avoid over fitting.

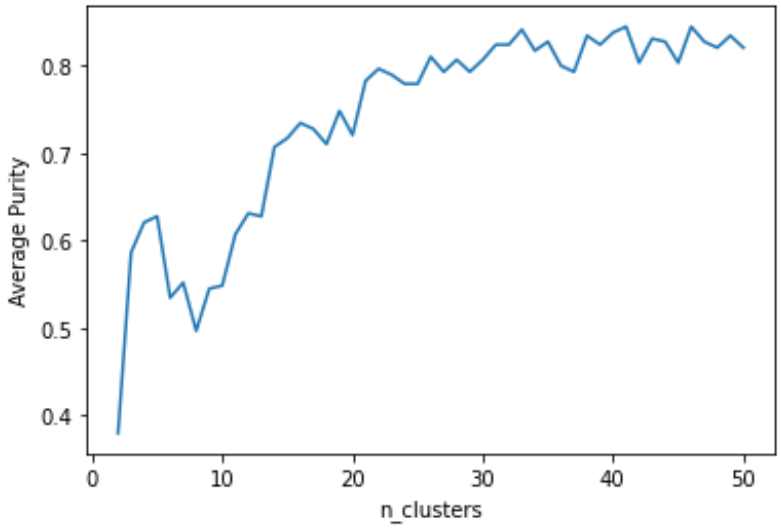



Figure 4.14: Average purity of the K-Means clusters

### 4.7.1 Analysis of the Semi Supervised Step Results

In Tables 4.2 and 4.3 we can observe the resulting label distribution from both models. One initial observation is that the Label propagation algorithm ensures that all data points have a label assigned to them at the end of the process, not leaving any hosts unlabeled. The K-Means approach on the other hand leaves some data unlabeled, if a cluster does not possess a single label or an equal number of both labels, all its similar hosts will remain unlabeled. The Label Propagation algorithm propagates the labels to all data until convergence, the results therefore simply indicate which labels were closer to each host in comparison to the other.

Considering the context at hand, leaving some hosts unlabeled can be considered reasonable if there is a high degree of uncertainty about the correct label for those hosts. On this point the K-Means approach is a more attractive solution since it keeps some data up for future decisions or interpretations. On the other hand, the Label propagation approach produces stable results, is easier to automate and it is more versatile to eventual increases in the data set size. This is due to the fact that the amount of data does not affect the underlying logic of the algorithm as it relies on the similarity of its data points, computed based on their feature vectors. The K-Means clustering approach needs a step of carefully choosing the appropriate number of clusters.

Table 4.2: Results using Label Propagation

| **Label Propagation** | |
|---|---|
| Hosts labeled 1 | Hosts Labeled 0 |
| 137 | 318 |

Table 4.3: Results using the K-Means clustering approach

| **K-Means Clustering** | | |
|---|---|---|
| Hosts labeled 1 | Hosts Labeled 0 | Hosts Unlabeled |
| 111 | 126 | 218 |

## 4.8 Decision Recommendation

The last step to complete our decision support system was to recommend decisions based on the labeled data. To accomplish this, we had to rely on the domain knowledge of the *novobanco* staff to gather information on how infrastructure management decisions are often made so that we can emulate and map those decisions on our set of hosts labeled as over-provisioned by each of the previously described approaches. Figure 4.15 represents the heuristics used to recommend a decision for each host with label 1.

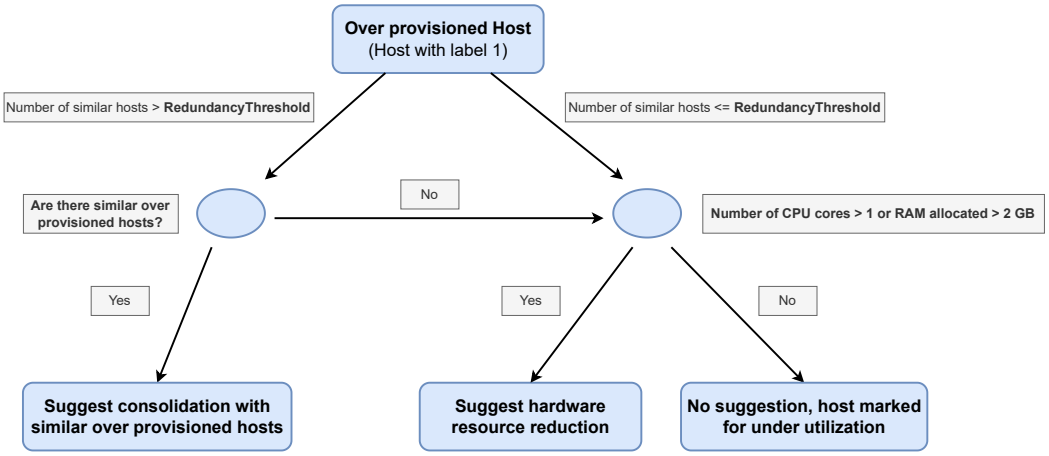



Figure 4.15: Decision Support Tree

For each host labeled as over-provisioned, we check for similar hosts which, in this context, are hosts with the same metadata specifications. We check if the number of similar hosts is higher than the redundancy threshold, which is a constant that defines the minimum level of redundancy that *novobanco* requires for its infrastructure. If the number of similar hosts is higher than the redundancy threshold we check for other hosts within that category group that were also labeled by each of the previously explained approaches. If there are other hosts with the same metadata specification that were labeled by the current approach as over-provisioned we suggest consolidation with those hosts, meaning an overall reduction of the number of hosts from that metadata group.

If there are no other similar hosts with label 1 or if the number of similar hosts is lower than the redundancy threshold defined, it's not possible to suggest consolidation with different hosts. However, it might still be feasible to scale down the resources of the host. To verify this, we chose to utilize the number of CPU cores and ram allocated as heuristics to discern if the host is eligible for a hardware resource reduction. For a server to be functional it needs at least 1 CPU core and 2 GB of RAM allocated, therefore we utilize this minimum threshold to assert if a hardware scale-down is possible. If a server is running with minimal hardware and does not have sufficient similar over-provisioned infrastructure (or similar infrastructure at all) we are unable to make any suggestion since we do not possess any domain knowledge about what distinctive functionalities or services this underutilized host is providing. If the services are indeed necessary we cannot reduce the number of hosts capable of running them under the redundancy threshold and we cannot scale down the hardware of hosts that already have such low hardware specifications.

We take this decision support tree and apply it to the data labeled by the label propagation and K-Means approaches. For simplicity, we set the redundancy threshold value to 1 (meaning there needs to be at least 2 hosts for each metadata category). As stated in Section 1.4, due to security constraints we cannot present any information about the metadata of the hosts and these results only mean anything inside the scope of the company, we can, however, present a count of the decisions provided by applying the decision support tree to both approaches, which corresponds to Tables 4.4 and 4.5 as well as provide an illustration of how the outputs are presented to the *novobanco* staff which is shown in Table 4.8.

Table 4.4: Recommended decisions for the Label Propagation approach

| **Decision Recommended** | **Decision Count** |
|---|---|
| Consolidation with similar over-provisioned hosts | 90 |
| Hardware Resource Reduction | 40 |
| No suggestion, host marked for under-utilization | 7 |

Table 4.5: Recommended decisions for the K-Means clustering approach

| **Decision Recommended** | **Decision Count** |
|---|---|
| Consolidation with similar over-provisioned hosts | 63 |
| Hardware Resource Reduction | 41 |
| No suggestion, host marked for under-utilization | 7 |

As we can observe, the number of hardware resource reduction suggestions is fairly similar in both approaches and the number of hosts without any possible suggestions is the same. The number of consolidation recommendations is higher on the label propagation approach which indicates that this approach identified more metadata categories, with some hosts that might be prone for consolidation.

Table 4.6: Representation of the output results

| **Host ID** | **Decision Recommended** | **Similar Over-provisioned Hosts** | **Applications Affected** |
|---|---|---|---|
| Host1 | Hardware Resource Reduction | [ - ] | [App1,App2,App3] |
| Host2 | Consolidation with similar over-provisioned hosts | [Host3, Host4] | [App4, App5, App6, App7] |
| Host3 | Consolidation with similar over-provisioned hosts | [Host2, Host4] | [App4, App5, App6, App7, App8] |
| Host4 | Consolidation with similar over-provisioned hosts | [Host2, Host3] | [App5, App8, App9] |
| Host5 | No suggestion, host marked for under-utilization | [ - ] | [App10] |

# 5

# Cost-Benefit Analysis

In this chapter, we will estimate what the economic impact would be if the decisions generated by the previously described models were followed. Due to security constraints, we are not able to reveal any contract information about the costs of the *novobanco* infrastructure, which also vary depending on different factors such as the external supplier that is providing the specific infrastructure, the time of the contract, discounts associated with having a large cluster of machines from a specific category as well as another multitude of contract details. In order to work around this issue, we have to map the costs of the *novobanco* hardware onto the prices of similar hardware from a public external provider. There are different available tools from cloud providers that provide a way to estimate expenditure before using a specified service. For our analysis, we will use the **Azure Pricing Calculator** to estimate the monthly costs of the studied data set.

The **Azure Pricing Calculator** [5] is a tool provided by *Microsoft* that can be used to estimate the cost of using the Azure Cloud to host specific workloads. It displays pricing information for different configurations, which account for CPU, memory, storage, location, and hours of usage. It can also display cost information for using different services such as databases, virtual machines, containers, and Azure functions. However, we will only use this tool to configure and simulate the costs of different virtual machine instances.

Since we are also simply not authorized to detail all hardware specifications related to the studied infrastructure, we will separate the hosts into different **hardware specification tiers**. The criteria to separate these categories will use the same variables as our decision support tree (Section 4.8), the number of CPU cores ($nCores$) of each host, and their allocated RAM ($allocatedRAM$). We consider four hardware tiers:

- Tier 1: Which contains all hosts with either less than 2 GB of RAM allocated or less than two cores, making this the tier with minimal specifications

$$nCores < 2 \; or \; allocatedRAM < 2GB \tag{5.1}$$

- Tier 2: Which contains all hosts with more than 1 CPU core but less than 8 CPU cores. All hosts must have more than 2 GB of RAM allocated to separate them from

the minimal tier.

$$2 <= nCores < 8 \; and \; allocatedRAM > 2GB \tag{5.2}$$

- Tier 3: Which contains all hosts with 8 or more CPU cores and less than 16 CPU cores.

$$8 <= nCores < 16 \tag{5.3}$$

- Tier 4: Which contains all hosts with 16 or more CPU cores.

$$nCores >= 16 \tag{5.4}$$

Each tier will be mapped onto a monthly cost derived from the Azure Pricing Calculator. We will only consider the standard prices for the West Europe region, and each price will be represented as an average between the Linux and Windows configuration options (Figure 5.1).

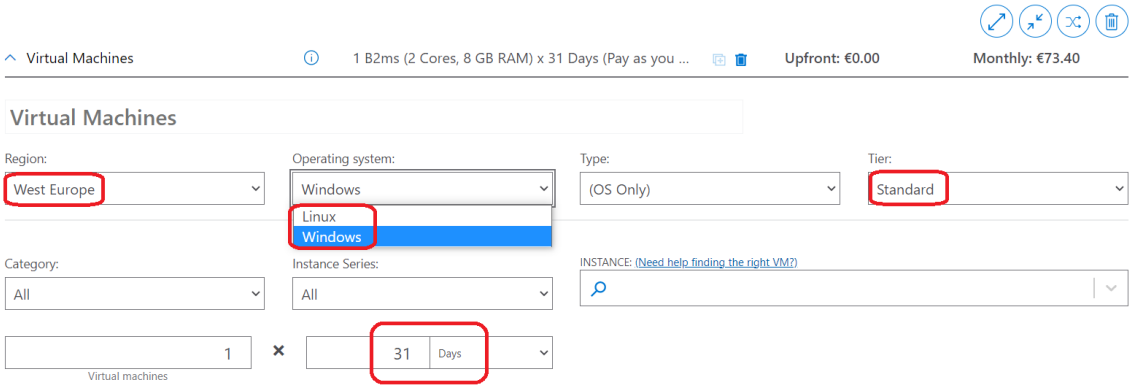



Figure 5.1: Fixed Configurations used on the Azure Pricing Calculator

Besides the baseline configurations, we also only consider Virtual Machines from the B-series. These machines offer a variety of hardware types and are optimal for workloads that do not need full performance continuously. By utilizing the VMs from the same series, we get a more rigorous comparison between the previously mentioned hardware tiers. We consider the VM instances shown in Figure 5.2. For **Tier 1** hosts, we consider the prices (for Windows and Linux VMs) of the B1ms instance, for **Tier 2** hosts, we consider the average of the costs of the **B2ms** and **B4ms** instances, for the **Tier 3** hosts we consider the average of the prices of the **B8ms** and **B12ms** instances and for the final **Tier 4** hosts we consider the average of the prices of the **B16ms** and **B20ms** instances.

Considering $CL$ the cost of the Linux (Ubuntu) instances and $CW$ the cost of the windows instances:

The monthly cost for each host of **Tier 1** will be:

$$\frac{CW_{B1ms} + CL_{B1ms}}{2} => \frac{19.55 + 16.94}{2} = 18.245 \tag{5.5}$$

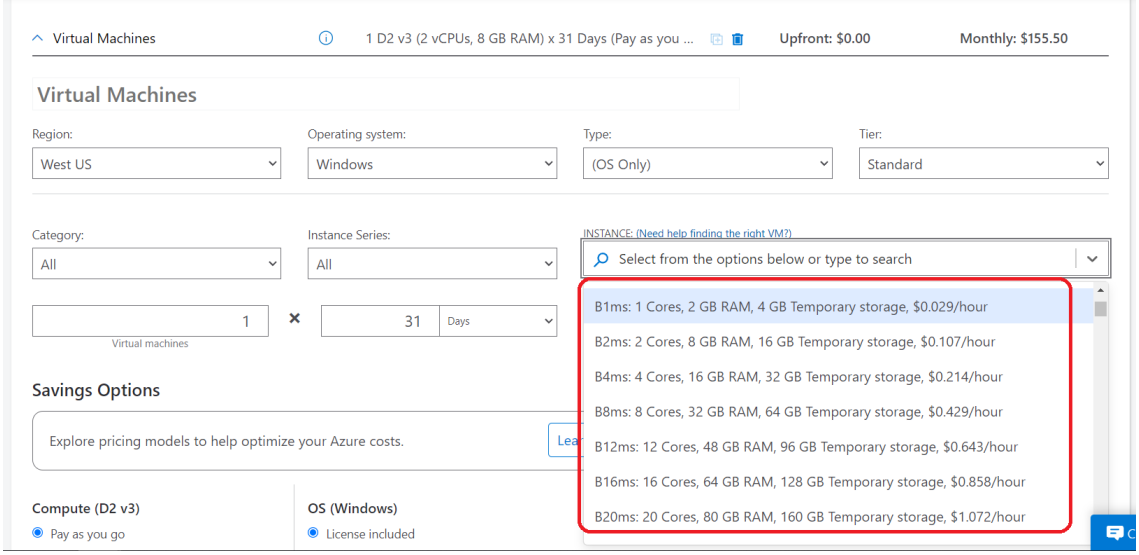



Figure 5.2: The VM instances considered

The monthly cost for each host of **Tier 2** will be:

$$\frac{CW_{B2ms} + CL_{B2ms} + CW_{B4ms} + CL_{B4ms}}{4} => \frac{73.40 + 67.76 + 146.80 + 135.51}{4} = 105.968 \tag{5.6}$$

The monthly cost for each host of **Tier 3** will be:

$$\frac{CW_{B8ms} + CL_{B8ms} + CW_{B12ms} + CL_{B12ms}}{4} => \frac{293.61 + 271.02 + 440.41 + 406.53}{4} = 352.893 \tag{5.7}$$

The monthly cost for each host of **Tier 4** will be:

$$\frac{CW_{B16ms} + CL_{B16ms} + CW_{B20ms} + CL_{B20ms}}{4} => \frac{587.21 + 542.04 + 734.01 + 677.55}{4} = 635,203 \tag{5.8}$$

Table 5.1: Tier Distribution of the Hosts

| **Tier 1** | **Tier 2** | **Tier 3** | **Tier 4** |
|---|---|---|---|
| 77 | 249 | 102 | 27 |

The studied data set had a hardware tier distribution shown in Table 5.1. By taking the previously calculated monthly costs of each tier, we estimate that maintaining the studied data set would cost 80,936.56 € per month (5.9).

$$77 * 18.245 + 249 * 105.968 + 102 * 352.893 + 27 * 635.203 = 80936.464 \tag{5.9}$$

We will now measure the impact of the choices recommended using both the **Label Propagation** and **K-Means** approaches. To do this we need to make assumptions.

Firstly, a "consolidation with similar over-provisioned hosts" decision, does not recommend any particular number of hosts that should remain, since this is also entirely dependent on the cluster of machines where that recommendation was made. For the purpose of this study, we will try to simulate different scenarios, by assuming the number of remaining machines using different consolidation ratios: a 2 to 1 ratio, a 3 to 2 ratio, and a 4 to 3 ratio.

For the "Hardware Resource Reduction" decision, we will assume that the hardware is "downgraded" to the tier under the one which corresponds to the hosts where this recommendation was applied, for example, a tier 3 host with this recommendation will be assumed to be reduced to a price equivalent of a tier 2 host.

Upon analysis of the resulting recommended decisions of both algorithms, we retrieved the tier distribution per decision for both algorithms represented in Tables 5.2 and 5.3

Table 5.2: Label Propagation approach

| | **Tier 1** | **Tier 2** | **Tier 3** | **Tier 4** |
|---|---|---|---|---|
| Hardware Resource Reduction | 0 | 25 | 11 | 4 |
| Consolidation with similar over-provisioned hosts | 7 | 55 | 24 | 4 |
| **Total** | 14 | 80 | 35 | 8 |

Table 5.3: K-Means approach

| | **Tier 1** | **Tier 2** | **Tier 3** | **Tier 4** |
|---|---|---|---|---|
| Hardware Resource Reduction | 0 | 22 | 14 | 4 |
| Consolidation with similar over-provisioned hosts | 12 | 35 | 12 | 5 |
| **Total** | 19 | 57 | 26 | 9 |

On the hosts which belong in **Tier 1** and have no suggestion applied, we consider that no changes are made. We can now calculate the impact of the changes by looking at the total cost of the labeled subset of each approach.

For the **Label Propagation** approach we have 14 **Tier 1** hosts, 80 **Tier 2** hosts, 35 **Tier 3** hosts and 8 **Tier 4** hosts, which results in a monthly cost of 26,165.75 € (5.10).

$$14 * 18.245 + 80 * 105.968 + 35 * 352.893 + 8 * 635.203 = 26165.749 \tag{5.10}$$

Applying the tier readjustments to the hosts marked for hardware reduction, we have 25 hosts which are moved from **Tier 2** to **Tier 1**, 11 hosts which are moved from **Tier 3** to **Tier 2** and four hosts which are moved from **Tier 4** to **Tier 3**. The following transformations are then applied:

- Number of Tier 1 Hosts ——> Number of Tier 1 Hosts + 25
- Number of Tier 2 Hosts ——> Number of Tier 2 Hosts - 25 + 11 = Number of Tier 2 Hosts - 14

- Number of Tier 3 Hosts ——> Number of Tier 3 Hosts - 11 + 4 = Number of Tier 3 Hosts - 7
- Number of Tier 4 Hosts ——> Number of Tier 4 Hosts - 4

We now consider the previously mentioned different consolidation ratios for hosts marked for consolidation. For the 2 to 1 ratio, we simply "cut" the number of hosts marked for consolidation in half, so the number of hosts has the following transformations applied:

- Number of Tier 1 Hosts ——> Number of Tier 1 Hosts - $\lceil \frac{7}{2} \rceil$
- Number of Tier 2 Hosts ——> Number of Tier 2 Hosts - $\lceil \frac{55}{2} \rceil$
- Number of Tier 3 Hosts ——> Number of Tier 3 Hosts - $\lceil \frac{24}{2} \rceil$
- Number of Tier 4 Hosts ——> Number of Tier 4 Hosts - $\lceil \frac{8}{2} \rceil$

By applying both transformations, we get the following results:

- Number of Tier 1 Hosts ——> $14 + 25 - \lceil \frac{7}{2} \rceil = 35$
- Number of Tier 2 Hosts ——> $80 - 14 - \lceil \frac{55}{2} \rceil = 39$
- Number of Tier 3 Hosts ——> $35 - 7 - \lceil \frac{24}{2} \rceil = 16$
- Number of Tier 4 Hosts ——> $8 - 4 - \lceil \frac{4}{2} \rceil = 2$

For the 3 to 2 ratio, we simply take a third of the number of hosts marked for consolidation (and round the number up), so the number of hosts has the following transformations applied:

- Number of Tier 1 Hosts ——> Number of Tier 1 Hosts - $\lceil \frac{7}{3} \rceil$
- Number of Tier 2 Hosts ——> Number of Tier 2 Hosts - $\lceil \frac{55}{3} \rceil$
- Number of Tier 3 Hosts ——> Number of Tier 3 Hosts - $\lceil \frac{24}{3} \rceil$
- Number of Tier 4 Hosts ——> Number of Tier 4 Hosts - $\lceil \frac{8}{3} \rceil$

By applying both transformations, we get the following results:

- Number of Tier 1 Hosts ——> $14 + 25 - \lceil \frac{7}{3} \rceil = 36$
- Number of Tier 2 Hosts ——> $80 - 14 - \lceil \frac{55}{3} \rceil = 47$
- Number of Tier 3 Hosts ——> $35 - 7 - \lceil \frac{24}{3} \rceil = 20$
- Number of Tier 4 Hosts ——> $8 - 4 - \lceil \frac{4}{3} \rceil = 2$

For the 4 to 3 ratio, we simply take a third of the number of hosts marked for consolidation (and round the number up), so the number of hosts has the following transformations applied:

- Number of Tier 1 Hosts ——> Number of Tier 1 Hosts - $\lceil \frac{7}{4} \rceil$
- Number of Tier 2 Hosts ——> Number of Tier 2 Hosts - $\lceil \frac{55}{4} \rceil$
- Number of Tier 3 Hosts ——> Number of Tier 3 Hosts - $\lceil \frac{24}{4} \rceil$
- Number of Tier 4 Hosts ——> Number of Tier 4 Hosts - $\lceil \frac{8}{4} \rceil$

By applying both transformations, we get the following results:

- Number of Tier 1 Hosts ——> $14 + 25 - \lceil \frac{7}{4} \rceil = 37$
- Number of Tier 2 Hosts ——> $80 - 14 - \lceil \frac{55}{4} \rceil = 52$
- Number of Tier 3 Hosts ——> $35 - 7 - \lceil \frac{24}{4} \rceil = 22$
- Number of Tier 4 Hosts ——> $8 - 4 - \lceil \frac{4}{4} \rceil = 3$

Table 5.4: Resulting Tier Distributions per consolidation ratios using the Label Propagation approach

| | **Tier 1** | **Tier 2** | **Tier 3** | **Tier 4** | Final monthly cost |
|---|---|---|---|---|---|
| **2 to 1 ratio** | 35 | 39 | 16 | 2 | 11688.02 € |
| **3 to 2 ratio** | 36 | 47 | 20 | 2 | 13965.58 € |
| **4 to 3 ratio** | 37 | 52 | 22 | 3 | 15854.66 € |

The resulting tier distributions per consolidation ratio, as well as the resulting monthly costs for the **Label Propagation** approach, are represented in Table 5.4. We can now calculate the reduction of expenditure by taking the initial monthly costs of the hosts identified by the **label propagation** approach by taking their initial monthly cost and subtracting the now resulting costs for each ratio (Table 5.5).

Table 5.5: Monthly cost savings using the different consolidation ratios for the Label Propagation approach

| | **Cost saving** | **Relative saving** |
|---|---|---|
| **2 to 1 ratio** | 14477.73 € | 17.89 % |
| **3 to 2 ratio** | 12200.17 € | 15.07% |
| **4 to 3 ratio** | 10311.09 € | 12.74% |

For the **K-Means** approach we have 12 **Tier 1** hosts, 57 **Tier 2** hosts, 26 **Tier 3** hosts and 9 **Tier 4** hosts, which results in a monthly cost of 21,278.88 € (5.11).

$$19 * 18.245 + 57 * 105.968 + 26 * 352.893 + 9 * 635.203 = 21278.876 \tag{5.11}$$

Applying the tier readjustments to the hosts marked for hardware reduction, we have 22 hosts which are moved from **Tier 2** to **Tier 1**, 14 hosts which are moved from **Tier 3** to **Tier 2** and four hosts which are moved from **Tier 4** to **Tier 3**. We must consider then the following transformations:

- Number of Tier 1 Hosts ——> Number of Tier 1 Hosts + 22
- Number of Tier 2 Hosts ——> Number of Tier 2 Hosts - 22 + 14 = Number of Tier 2 Hosts - 8
- Number of Tier 3 Hosts ——> Number of Tier 3 Hosts - 14 + 4 = Number of Tier 3 Hosts - 10
- Number of Tier 4 Hosts ——> Number of Tier 4 Hosts - 4

By applying the same calculation process previously applied for the **label propagation** approach, we get the following transformations for each consolidation ratio.

**For the 2 to 1 ratio:**

- Number of Tier 1 Hosts ——> $19 + 22 - \lceil \frac{12}{2} \rceil = 35$
- Number of Tier 2 Hosts ——> $57 - 8 - \lceil \frac{35}{2} \rceil = 31$
- Number of Tier 3 Hosts ——> $26 - 10 - \lceil \frac{12}{2} \rceil = 10$
- Number of Tier 4 Hosts ——> $9 - 4 - \lceil \frac{5}{2} \rceil = 2$

**For the 3 to 2 ratio:**

- Number of Tier 1 Hosts ——> $19 + 22 - \lceil \frac{12}{3} \rceil = 37$
- Number of Tier 2 Hosts ——> $57 - 8 - \lceil \frac{35}{3} \rceil = 37$
- Number of Tier 3 Hosts ——> $26 - 10 - \lceil \frac{12}{3} \rceil = 12$
- Number of Tier 4 Hosts ——> $9 - 4 - \lceil \frac{5}{3} \rceil = 3$

**For the 4 to 3 ratio:**

- Number of Tier 1 Hosts ——> $19 + 22 - \lceil \frac{12}{4} \rceil = 38$
- Number of Tier 2 Hosts ——> $57 - 8 - \lceil \frac{35}{4} \rceil = 40$
- Number of Tier 3 Hosts ——> $26 - 10 - \lceil \frac{12}{4} \rceil = 13$
- Number of Tier 4 Hosts ——> $9 - 4 - \lceil \frac{5}{4} \rceil = 3$

Table 5.6: Resulting Tier Distributions per consolidation ratios using the K-Means approach

| | **Tier 1** | **Tier 2** | **Tier 3** | **Tier 4** | Final monthly cost |
|---|---|---|---|---|---|
| **2 to 1 ratio** | 35 | 31 | 10 | 2 | 8722.92€ |
| **3 to 2 ratio** | 37 | 37 | 12 | 3 | 10736.21 € |
| **4 to 3 ratio** | 38 | 40 | 13 | 3 | 11425.25 € |

Table 5.7: Monthly cost savings using the different consolidation ratios for the K-Means approach

| | **Cost saving** | **Relative saving** |
|---|---|---|
| **2 to 1 ratio** | 12555.96 € | 15.51 % |
| **3 to 2 ratio** | 10542.67 € | 13.03% |
| **4 to 3 ratio** | 9853.63 € | 12.17% |

The resulting tier distributions per consolidation ratio, as well as the resulting monthly costs for the **K-Means** approach, are described in Table 5.6, and the cost savings of using this approach are shown in Table 5.7.

We observe an apparent cost reduction in the studied data set, which is considerably small compared to the whole *novobanco* infrastructure. Although this cost estimation is not as perfectly accurate as a direct mapping with the real infrastructure costs, it can still give a perspective of relative savings for each approach. As expected, since the **label propagation** approach has more hosts labeled as over-provisioned, it will result in higher cost savings if, of course, we consider all its decisions viable, although this difference becomes less visible when we reduce fewer hosts in the consolidation process (lower consolidation ratio).

The costs studied are monthly costs, even assuming discounted prices by special contracts with external providers, the delay in taking action about over-provisioned infrastructure significantly increases the costs. By looking at the relative cost-saving percentage, we can estimate that 6 to 8 months of no actions on over-provisioned infrastructure would add another month of costs for the whole data set of hosts (assuming the infrastructure does not change).

What is not considered is the indirect cost of managing infrastructure in a non-automated way. The larger the data center, the higher the difficulty for the staff to supervise all the heterogeneous hosts, and the higher the management costs will be. It might even be less costly to leave some over-provisioned infrastructure than pay for continuous decision-making and readjustment, hence the benefit of automating tasks to identify and recommend new candidates for certain decisions.

# 6

# Conclusion

Like other computation infrastructures, the *novobanco* data center is constantly growing. This constant growth entails an increasing need to continuously take infrastructure management decisions. Since *novobanco* hosts a great portion of its applications on on-premise infrastructure, there is a constant need to scale resources in order to fit the needs of all services and avoid unnecessary costs. However, due to the high diversity of applications, often running on the same machines, it becomes difficult to keep track of the changes that need to be made constantly.

As the company grows, so do its services, some of which start to get a higher usage, and the infrastructure where those services are hosted might need some adjustments to increase its capacity. The opposite is also frequent, where services become less used over time (often becoming legacy software) and the amount of resources assigned to those services might no longer be necessary. Identifying these issues and what should be done about them requires a deep knowledge of the data center, increasing the burden for the *novobanco* staff which holds the responsibility of monitoring and adjusting the bank's critical infrastructure. There is a great interest in automating the process of identifying infrastructure issues in order to ease the management process, however, this is not a simple task, since it is not possible to define a set of rules that apply to all the hosts in the novobanco data center. The hosts have their behavior affected by their overall purpose and functionalities, this behavior is reflected by their workload but it is also highly dependent on their environment, service capabilities, operating system, site, domain, and even on the very nature of the applications hosted.

Considering the obstacles mentioned, this thesis work had the purpose of aiding the management process of the data center infrastructure by adding mechanisms to extract data from different sources, combine this data in a way to measure similarity between the different hosts, and use input from the staff about identified hosts with a specified issue, in order to train a model which will try to identify other hosts where the same issue might be visible. In order to facilitate decision-making, the results from the machine learning algorithms are also used to recommend decisions using simple heuristics recommended by the staff.

## 6.1 Main Contributions

The main contributions of this thesis work were:

- A method to extract a large amount of host workload data from the check_mk monitoring system
- A method to combine data from the hosts using their workload patterns and metadata features in order to measure host similarity
- A process to capture the structure of data and identify its patterns by using its eigenvalues and eigenvectors
- Two semi-supervised learning approaches that use the similarity measure of the hosts to expand user-labeled data to the entire data set
- A simple decision support tree that recommends decisions based on the results from the semi-supervised approaches with the purpose of easing the decision-making process

In the conducted work, we developed methods to facilitate decision-making about infrastructure management. We focused specifically on identifying over-provisioned infrastructure and recommending scaling decisions on its different hosts, however, the designed methods can be used transversely to other decision labels.

Most decisions regarding infrastructure management on a large scale, likely require a process of extracting (or observing) a considerable amount of data and identifying the groups of hosts where the decisions might apply using their workload patterns and/or metadata. In migrating resources to a cloud environment for example, by using our data fusion processes it can be possible to match workload patterns that are considered beneficial to be migrated to the cloud (Section 2.1), as well as identify hosts with specifications (metadata) that allow a cloud migration.

Simply accelerating the decision-making process for resource scaling decisions can, as described in Chapter 5, considerably reduce expenditure (around 12% to 17% of relative savings) as well as ease the burden on the staff responsible for the management of the data center.

## 6.2 Limitations and Future Work

There were some limitations on the work that was presented that should be taken into account. Firstly, due to the lack of labeled data, the semi-supervised algorithms could not be fully tested in a train, validation, and test split, it is, therefore, necessary to further test the efficacy of these approaches (as well as try new ones) and compare their results. Secondly, some decisions might be based on application characteristics rather than on

infrastructure details, however, we did not possess any data regarding application usage or other application details, these require generating a report where an application is monitored for a time frame, since applicational data is not monitored passively by check_mk. By extending this thesis work for other decisions, such as cloud migrations, analyzing the workload and the characteristics of the hosts is essential but not sufficient. The migration of workloads entails the migration of applications or services responsible for such workloads and, therefore, it would be interesting to design a process to monitor or extract the data from the applications, to predict if they are compatible to be migrated.

This work focused on developing processes to automate tasks that tend to be done when making management decisions about data center infrastructure, however, further mechanisms are necessary to integrate these processes in an intuitive way. It would be beneficial to develop a process where hosts marked by the staff for any change or decision would be registered, for example, using a user interface (which was not developed in this work) to continuously train the models further. Further automation would be also favorable for the semi-supervised approaches, which although fairly simple to implement, required a step of choosing different parameters as well as identifying the eigengap in the data set manually.

# Bibliography


[1] https://oss.oetiker.ch/rrdtool/doc/rrddump.en.html (cit. on p. 36).

[2] V. Andrikopoulos, S. Strauch, and F. Leymann. "Decision support for application migration to the cloud". In: *Proceedings of CLOSER* 13 (2013), pp. 149–155 (cit. on pp. 2, 5, 9).

[3] *Art. 3 GDPR*. URL: https://gdpr.eu/article-3-requirements-of-handling-personal-data-of-subjects-in-the-union/ (visited on 03/16/2023) (cit. on p. 9).

[4] D. Arthur and S. Vassilvitskii. *k-means++: The advantages of careful seeding*. Tech. rep. Stanford, 2006 (cit. on p. 19).

[5] *Azure Pricing Calculator*. URL: https://azure.microsoft.com/en-us/pricing/calculator/ (visited on 01/22/2023) (cit. on p. 55).

[6] M. F. Bari et al. "Data Center Network Virtualization: A Survey". In: *IEEE Communications Surveys Tutorials* 15.2 (2013), pp. 909–928. DOI: 10.1109/SURV.2012.090512.00043 (cit. on p. 5).

[7] D. Berthelot et al. "Mixmatch: A holistic approach to semi-supervised learning". In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 17).

[8] J. Bleiholder and F. Naumann. "Data Fusion". In: *ACM Comput. Surv.* 41.1 (Jan. 2009). ISSN: 0360-0300. DOI: 10.1145/1456650.1456651. URL: https://doi.org/10.1145/1456650.1456651 (cit. on p. 10).

[9] *Categorical Encoding*. URL: https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/ (visited on 03/17/2023) (cit. on p. 12).

[10] S. Chaudhuri, U. Dayal, and V. Ganti. "Database technology for decision support systems". In: *Computer* 34.12 (2001), pp. 48–55. DOI: 10.1109/2.970575 (cit. on p. 9).

[11] *Checkmk as an alternative to Nagios*. URL: https://checkmk.com/product/nagios-alternative (visited on 01/15/2023) (cit. on p. 24).

[12] *Checkmk everything monitored*. URL: https://checkmk.com/ (visited on 01/15/2023) (cit. on p. 24).

[13] *Checkmk Product Features*. URL: https://checkmk.com/product/features (visited on 01/15/2023) (cit. on p. 24).

[14] "Cluster kernels for semi-supervised learning". In: *Advances in neural information processing systems* 15 (2002) (cit. on pp. 18, 46, 47).

[15] M. S. Coelho. "Patterns in financial markets: Dynamic time warping". PhD thesis. NSBE-UNL, 2012 (cit. on p. 13).

[16] J. C. Gower. "A general coefficient of similarity and some of its properties". In: *Biometrics* (1971), pp. 857–871 (cit. on pp. 12, 13).

[17] M. Johnson et al. "Evolving standards for IT service management". In: *IBM Systems Journal* 46 (Feb. 2007), pp. 583–597. DOI: 10.1147/sj.463.0583 (cit. on p. 23).

[18] *K-Means Clustering Algorithm*. URL: https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation (visited on 03/07/2023) (cit. on p. 19).

[19] S. Liu et al. "Integration of decision support systems to improve decision support performance". In: *Knowledge and Information Systems* 22 (2010), pp. 261–286 (cit. on p. 8).

[20] Y. Luo, K. Liu, and D. Davis. "A multi-agent decision support system for stock trading". In: *IEEE Network* 16.1 (2002), pp. 20–27. DOI: 10.1109/65.980541 (cit. on p. 8).

[21] H. Madduri et al. "A configuration management database architecture in support of IBM Service Management". In: *IBM Systems Journal* 46 (Feb. 2007), pp. 441–457. DOI: 10.1147/sj.463.0441 (cit. on p. 23).

[22] *Maximizing efficiency and expenditure on the cloud*. URL: https://learn.microsoft.com/pt-pt/training/modules/azure-well-architected-cost-optimization/5-maximize-efficiency-of-cloud-spend (visited on 03/05/2023) (cit. on p. 6).

[23] M. W. L. Moreira et al. "A Comprehensive Review on Smart Decision Support Systems for Health Care". In: *IEEE Systems Journal* 13.3 (2019), pp. 3536–3545. DOI: 10.1109/JSYST.2018.2890121 (cit. on p. 8).

[24] M. Müller. "Dynamic time warping". In: *Information retrieval for music and motion* (2007), pp. 69–84 (cit. on p. 13).

[25] *Nagios Extension Library*. URL: https://exchange.nagios.org (visited on 01/15/2023) (cit. on p. 24).

[26] *Nagios Main Page*. URL: https://www.nagios.org/ (visited on 01/15/2023) (cit. on p. 23).

[27] *Nagios product Features*. URL: https://www.nagios.com/products/nagios-xi/features (visited on 01/15/2023) (cit. on p. 24).

[28] *Nearest neighbors*. URL: https://scikit-learn.org/stable/modules/neighbors.html (visited on 03/08/2023) (cit. on p. 18).

[29] M. Pawlish, A. S. Varde, and S. Robila. "Decision support in data centers for sustainability". In: *2013 IEEE 13th International Conference on Data Mining Workshops*. IEEE. 2013, pp. 613–620 (cit. on p. 9).

[30] Y. Permanasari, E. H. Harahap, and E. P. Ali. "Speech recognition using dynamic time warping (DTW)". In: *Journal of physics: Conference series*. Vol. 1366. 1. IOP Publishing. 2019, p. 012091 (cit. on p. 13).

[31] A. Pikrakis, S. Theodoridis, and D. Kamarotos. "Recognition of isolated musical patterns using context dependent dynamic time warping". In: *IEEE Transactions on Speech and Audio Processing* 11.3 (2003), pp. 175–183 (cit. on p. 13).

[32] D. Power. *Decision Support Systems: Concepts and Resources for Managers*. Mar. 2002. ISBN: 156720497X (cit. on p. 8).

[33] *Quem somos | novobanco*. URL: https://www.novobanco.pt/investidores/sobre-nos/quem-somos (visited on 02/17/2022) (cit. on p. 3).

[34] V. Rajaraman. "Cloud computing". In: *Resonance* 19.3 (2014), pp. 242–258 (cit. on p. 1).

[35] *RBF SVM parameters*. URL: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html (visited on 03/08/2023) (cit. on p. 18).

[36] *Reading and analysing log files in the RRD database format*. URL: https://rviews.rstudio.com/2018/06/20/reading-rrd-files/ (visited on 01/20/2023) (cit. on pp. 25, 26).

[37] H. Rong et al. "Optimizing energy consumption for data centers". In: *Renewable and Sustainable Energy Reviews* 58 (2016), pp. 674–691. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2015.12.283. URL: https://www.sciencedirect.com/science/article/pii/S1364032115016664 (cit. on p. 5).

[38] *Round-Robin Business Intelligence*. URL: https://sureshjoshi.com/development/round-robin-bi-rrd-explained (visited on 01/20/2023) (cit. on p. 25).

[39] *RRD Format*. URL: https://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html (visited on 01/20/2023) (cit. on p. 25).

[40] *RRDtool logging graphing*. URL: https://oss.oetiker.ch/rrdtool/ (visited on 01/22/2023) (cit. on p. 26).

[41] *Sap environment landscape*. URL: https://www.tutorialspoint.com/sap_basis/sap_basis_system_landscape_architecture.htm (visited on 02/15/2023) (cit. on p. 22).

[42] M. Schmitt and X. Zhu. "Data Fusion and Remote Sensing – An Ever-Growing Relationship". In: *IEEE Geoscience and Remote Sensing Magazine* 4 (Dec. 2016), pp. 6–23. DOI: 10.1109/MGRS.2016.2561021 (cit. on pp. 10, 11).

[43] *Semi-Supervised: Label Propagation*. URL: https://scikit-learn.org/0.15/modules/label_propagation.html (visited on 03/08/2023) (cit. on p. 18).

[44] P. Senin. "Dynamic time warping algorithm review". In: *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA* 855.1-23 (2008), p. 40 (cit. on pp. 14, 15).

[45] M. J. Shaw et al. "Knowledge management and data mining for marketing". In: *Decision Support Systems* 31.1 (2001). Knowledge Management Support of Decision Making, pp. 127–137. ISSN: 0167-9236. DOI: https://doi.org/10.1016/S0167-9236(00)00123-8. URL: https://www.sciencedirect.com/science/article/pii/S0167923600001238 (cit. on p. 8).

[46] M. Shokoohi-Yekta et al. "Generalizing DTW to the multi-dimensional case requires an adaptive approach". In: *Data mining and knowledge discovery* 31 (2017), pp. 1–31 (cit. on p. 16).

[47] J. E. Van Engelen and H. H. Hoos. "A survey on semi-supervised learning". In: *Machine learning* 109.2 (2020), pp. 373–440 (cit. on p. 17).

[48] W. Wang et al. "Time series clustering based on dynamic time warping". In: *2018 IEEE 9th international conference on software engineering and service science (ICSESS)*. IEEE. 2018, pp. 487–490 (cit. on p. 13).

[49] *What is Cloud Computing?* en-us. URL: https://www.ibm.com/cloud/learn/cloud-computing (visited on 02/17/2022) (cit. on p. 1).

[50] R. Wu, S. Sukhanov, and C. Debes. "Real time pattern matching with dynamic normalization". In: *arXiv preprint arXiv:1912.11977* (2019) (cit. on p. 43).

[51] X. Zhu and Z. Ghahramani. "Learning from labeled and unlabeled data with label propagation". In: (2002) (cit. on p. 17).