NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

GONÇALO FURTADO MATEUS

BSc in Computer Science Engineering

DATA AND COMPUTER CENTER PREDICTION OF USAGE AND COST

AN INTERPRETABLE MACHINE LEARNING APPROACH

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
November, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

DATA AND COMPUTER CENTER PREDICTION OF USAGE AND COST

AN INTERPRETABLE MACHINE LEARNING APPROACH

GONÇALO FURTADO MATEUS

BSc in Computer Science Engineering

Adviser: Cláudia Alexandra Magalhães Soares
Assistant Professor, NOVA University Lisbon

Co-advisers: João Carlos Antunes Leitão
Assistant Professor, NOVA University Lisbon

Antonio José Rodrigues
—-, novobanco

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
November, 2022

DATA AND COMPUTER CENTER PREDICTION OF USAGE AND COST

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.4).

Abstract

In recent years, Cloud computing use has considerably increased and, nowadays, is the backbone of many emerging applications. However, behind cloud structures, we have physical infrastructures (data centers) in which, managing the resources, is a difficult task due to unpredictable utilization patterns.

Accurate data center resource utilization prediction is necessary to gain efficiency, save money, and provide clients with better and faster services. It is fundamental to predict the volume of requests to improve the data center management process. Recently both the research and academic-industry have made an effort to bring intelligence to data center management mechanisms. However, solutions created are not enough to handle all patterns of real-world data center resources utilization.

The use of machine learning is fundamental to help in the management process of data center resources. However, models to predict resource utilization can increase network latency, which causes the gain of predictions to be nullified by the extra latency created in the network. Besides, data centers are critical because wrong predictions can lead to the worst cloud services, so this thesis aims to develop models that can explain their predictions. Working with real data from a national bank, we intend to build a model that uses low-level network and computation metrics as feature signals to predict global usage and cost, considering extra latency that models could create in the network. Also, in this work, we will develop a novel method in which we will explore the impact of human context on computational usage.

This thesis will be conducted in collaboration with *NOVO BANCO*.

Keywords: Data centers, Machine learning models, Interpretable models, Resource management, Natural language processing

Resumo

Nos últimos anos, a Computação em nuvem tem tido um aumento considerável e hoje em dia, é vista como a base de muitas aplicações emergentes. No entanto, por trás das conhecidas nuvens, existem infraestruturas físicas (centro de dados) nas quais, gerenciar os recursos, tem se revelado uma tarefa bastante difícil devido à imprevisibilidade de utilização dos serviços de nuvens.

Prever corretamente a utilização dos recursos dos centros de dados tornou-se numa das coisas mais importantes. Boas previsões permitem melhorar o processo de gestão dos centros de dados, o que torna possível obter mais eficiência, economizar dinheiro e fornecer rápidos serviços aos clientes. Recentemente, quer os investigadores, quer a indústria acadêmica têm feito um esforço para trazer inteligência para os mecanismos de gestão dos centros de dados. No entanto, as soluções desenvolvidas até ao momento, ainda não são suficientes para lidar com todos os tipos de padrões de utilização dos centros de dados, em situações do mundo real.

Devido a isto, a utilização de Machine Learning torna-se fundamental para auxiliar no processo de gestão dos recursos dos centros de dados. Contudo, a utilização de alguns tipos de modelos para prever a utilização de recursos pode aumentar a latência na rede, o que faz com que o ganho das previsões seja anulado pela nova latência criada na rede. Além disto, os centros de dados são uma área crítica porque más previsões levam a que haja perdas na qualidade dos serviços. Assim, este projeto visa criar modelos que expliquem as previsões que fazem. Trabalhando com dados reais de um banco português, o nosso objetivo é desenvolver um modelo que utilize métricas da rede dos centros de dados, para prever o uso e custo global, sem nunca esquecer a latência extra que os modelos podem criar na rede. Além disto, nesta tese, pretendemos desenvolver um novo método, no qual iremos explorar o impacto do contexto humano na utilização dos recursos dos centros de dados.

Esta tese será realizada em colaboração com o *novobanco*.

Palavras-chave: Centro de dados, Modelos de Machine Learning, Modelos interpretáveis, Gestão de recursos, Processamento de linguagem natural

Contents

Appendices

Annexes

List of Figures

List of Tables

Acronyms

AIC Akaike Information Criterion 14
AR Autoregressive 10
ARIMA Autoregressive integrated moving average 9, 10, 13
ARMA Autoregressive and Moving Average 10
AWS Amazon Web Services 5

BIC BIC Bayesian Information Criterion 14
BRDES Brown Double Exponential Smoothing 10

DC Data Center viii, 5–9, 11–13, 16, 20
DCN Data Center Network 7, 8
DMM Dirichlet Multinomial Mixture vi, 53, 54, 57, 58
DNN Deep Neural NetworK 11, 13

EDA Exploratory data analysis vi, 22, 35, 36
EN Elastic Net 12

G-SVM Gaussian Support Vector Machine 10
GAMs Generalized Additive Models 16
GBT Gradient boosting tree 10, 13
GLMs Generalized Linear Models 16
GRU Gated Recurrent Unit 9, 13

IaaS Infrastructure as a Service 5

KR Krigin 13

L-SVM Linear Support Vector Machine 10
LASSO Least absolute shrinkage and selection operator 12

LDA Latent Dirichlet Allocation 53
LR Linear regression 10, 12, 13, 15, 16
LSTM Long-short Term Memory 9, 13, 14

MOE Mixture of Experts 10

NB Gaussian Naive Bayes 13
NLP Natural Language Processing vi, 45, 46
NNLS Non-negative least square 12

Onprem On-premises data center 5–7

PaaS Plataform as a Service 5

RDF Random Decision Forest 10
RR Ridge regression 12
RRD Round Robin Database viii, 20, 21

SaaS Software as a Service 5
SARIMA Seasonal Autoregressive Integrated Moving Average 14
SVM Support Vector Machine 10, 12, 13

1

Overview: The growth of cloud computing

In this Chapter, we present an overview of this dissertation, highlighting the context and motivation of this work. It is also presented a description of the scope of this document. This dissertation will be conducted in collaboration with novobanco.

1.1 Context and Motivation

With the continuous growth of technology, computer system resources, such as storage, have migrated to the cloud. In the last five years, it is estimated that the number of organizations using cloud services increased from a mere 48% to 84% [1]. This is explained by the technological advancements and the scalability, 24/7 on-demand availability, flexibility, and pay-as-you-go billing mode that cloud infrastructure offers [2]. On the other hand, the SARS-CoV-2 pandemic also has increased the use of cloud computing, showing its importance in connecting and sharing for people located in different physical places [3].

Behind clouds, we have physical infrastructures distributed over different locations, being that in each location, we have a data center. All the data centers, servers, switches, and routers create a network that ensures excellent performance to programs that use them.

However, this increase in the usage of cloud computing has created some problems in data center networks related to resources management. Unpredictable resource utilization patterns on the network make it difficult to plan and split operations between machines, which leads to situations where for the number of job requests in the network, the number of resources allocated to those jobs is not optimal. When too few resources are allocated (under-provisioning), some resources have to be activated on-demand, leading to performance problems. Although when too many resources are allocated (over-provisioning), this will make some resources to be running idle [4]. Having efficient management of data center resources is vital to maximizing the performance having better operational costs [5].

Data center networks are critical areas, and the large-scale growth makes management a challenging task to be done by hand. This urges for automation to gain efficiency and provide better services to clients. It is fundamental to predict the volume of requests to improve the data center management process. Improving this process will allow for better resource allocation, better and faster services, and significant cost savings in terms of resources and energy.

In this thesis, in collaboration with *novobanco*, we will address the problem of taking low-level network and computation metrics as feature signals to be used by a machine learning model to predict global usage and cost of data center network. We will use a real dataset provided by the bank to predict computation metrics. However, more than predict metrics, we will try to provide explanations for those predictions. Providing explanations allows human decision-makers to have confidence and understand the decisions.

However, sometimes, human decisions are influenced by external factors. For example, people can do more or fewer operations in the bank depending on publicized news about the economy and the bank itself. As we will be working with real data, in this thesis, we will also try to study if there is some relation between economic/bank news and resources utilization. For this, we will use publications from Twitter to study the human context in the utilization of data centers resources. Twitter is a real-time platform where people can freely express their opinions and ideas, and in which social media channels publish news to reach people quickly and easily.

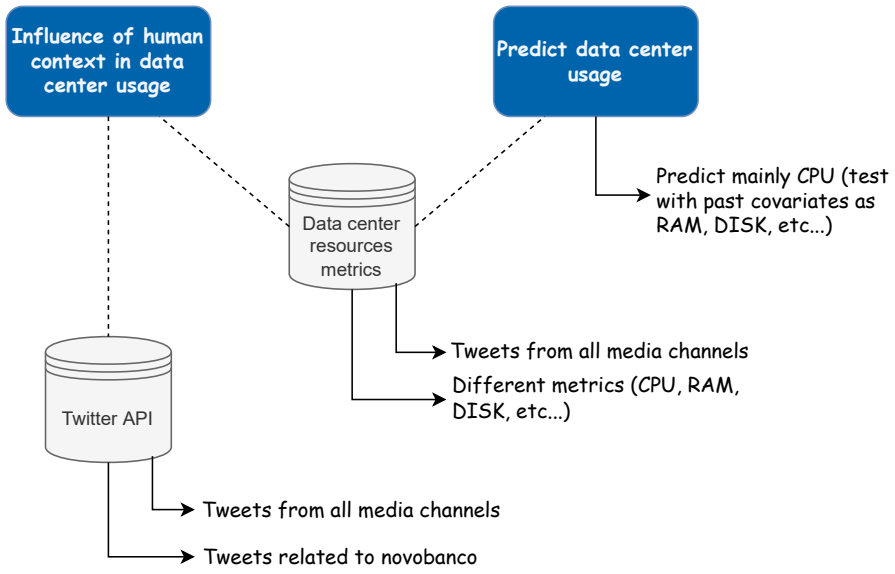


Figure 1.1: Thesis overview diagram

With the help of Figure 1.2, our objective will be to predict mainly the CPU usage, as is the most critical metric, and for the first objective of this dissertation (predict global usage and cost of data center network) we will try to predict with and without past covariates. These covariates will be other metrics of the servers that could improve our results. For the second objective of this dissertation (influence of human context in data center usage),

we will add future covariates to the models. These covariates will be the tweets related to the *novobanco* obtained from news and users.

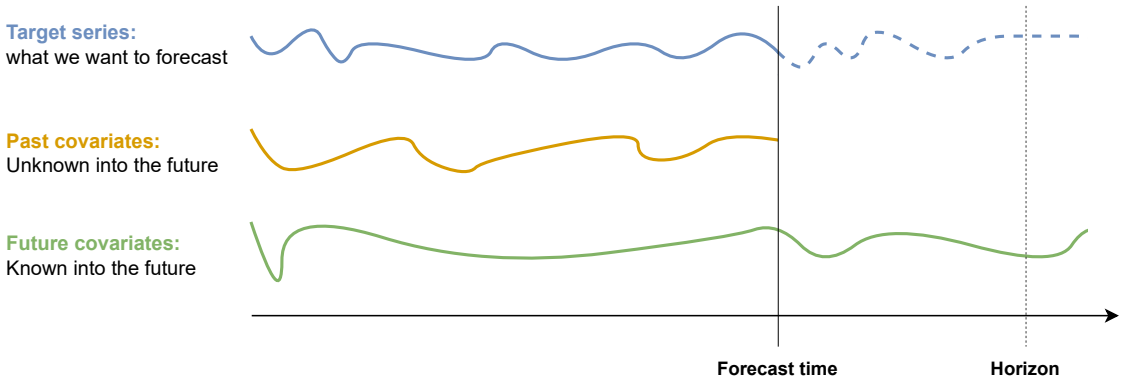


Figure 1.2: Covariates series. (Image based on [6])

1.2 Organization of the document

To better understand the organization of this document, we can think split into two parts. The first one, composed of Chapters 2, 3, and 4, we take the problem of trying to predict the usage of *novobanco* data centers through interpretable and not interpretable machine learning models. In the second part, composed of Chapters 5, 6, and 7, we address the problem of studying the relation of human context with data center utilization. In sum, a more detailed explanation of the remaining of this thesis is presented next:

1. Chapter 2 (State of the art) summarizes the required knowledge needed to understand the subsequent chapters in this dissertation. In the first part of the chapter, we present some background to frame our work and related works. The second part is dedicated to interpretable models, explaining their importance and presenting some of the models more relevant to our application.

2. Chapter 3 (*Novobanco* data center) aims to explain the *novobanco* data center dataset used in this thesis. It is presented two datasets, one referring to the metrics of data center servers (the main one) and the other related to the number of logins in *novobanco* platform. During this chapter it is presented an initially overview of this dataset, its format, and an exploration among the several features present in them.

3. Chapter 4 (Datacenter analysis and results)

4. Chapter 5 (Twitter data) describes the Twitter dataset that is used in this thesis. This Chapter explains why Twitter was chosen as our dataset source to study human context and analyzes the dataset. Note that another dataset will be used in this thesis, but this Chapter just presents the Twitter dataset.

5. Chapter 6 (Interpreting the Twitter Data) outlines two different approaches, Sentiment analysis and Topic modeling, used to interpret the content of tweets dataset. We start by presenting how we prepared the text of tweets to be analyzed, and next, the experiments and the results obtained with the two methods used.

6. Chapter 7 (Influence of human context)

7. Chapter 8 (Conclusion)

2

State of the art

In this Chapter, we summarize the required knowledge needed to understand the subsequent chapters in this dissertation. We address some of the works related to our own, providing its context and motivation for why it is relevant.

2.1 Data centers and machine learning

2.1.1 Data Centers overview

Data Center (DC) is a physical facility that houses critical computing resources as data or applications [7]. They comprise several components such as servers, switches, and routers that work together to ensure a high level of performance to applications and programs that are using it [8]. DC is a crucial point in cloud computing as it enables users to access resources and process programs on-demand [9]. Also, DC needs to be scalable and efficient, to provide a fast and secure connection between all components, to deal with the continuous evolution of the network, and the ever-growing cloud computing needs [10]. Companies regularly use them to centralize information in one point, in a way that can be accessed from everywhere.

Two DC solutions are typically adopted by the big companies: On-premises data center (Onprem) and cloud solutions. Onprem refers to private DC houses in companies' facilities and maintained by themselves. In contrast, cloud computing is defined as services provided by third-party service providers. There are different types of services provided by cloud computing, but the more used and known are Infrastructure as a Service (IaaS), Plataform as a Service (PaaS) and Software as a Service (SaaS) solutions. Cloud solutions bring advantages such as better energy costs, scalability, high levels of security, and predictable costs. With the growth of this type of computation, a lot of cloud solutions have been appearing provided by big companies as Amazon (AWS), Google (Google Cloud), Microsoft (Azure) or IBM (IBM Cloud). However, the usage of this kind of cloud solution brings some disadvantages, mainly for countries more peripheral as Portugal, because usually, the DC physical structures are located in Europe center regions. That is

why some companies do not have their resources totally in the cloud and additionally use some private data centers housed in companies' facilities and maintained by themselves (Onprem). Besides, this approach allows not to be dependent on either cloud providers or their Onprem data centers.

2.1.2 *Novobanco* data center structure

In the specific case of *novobanco*, their DC is organized as Figure 2.1 shows. *Novobanco* follows the practices of having either Onprem and cloud data centers, each of which is used for different types of operations. Knowing that *novobanco* is a bank, it is required to have a high level of security and be careful in all operations that need to access private resources of the bank so that it does not happen that the bank's customers are harmed. In the following explanation, it is going to be presented the general overview of *novobanco* data center more related to the direct channels, as these are the ones that deal with most of the critical and private information of the bank and, therefore, the most important to optimize.

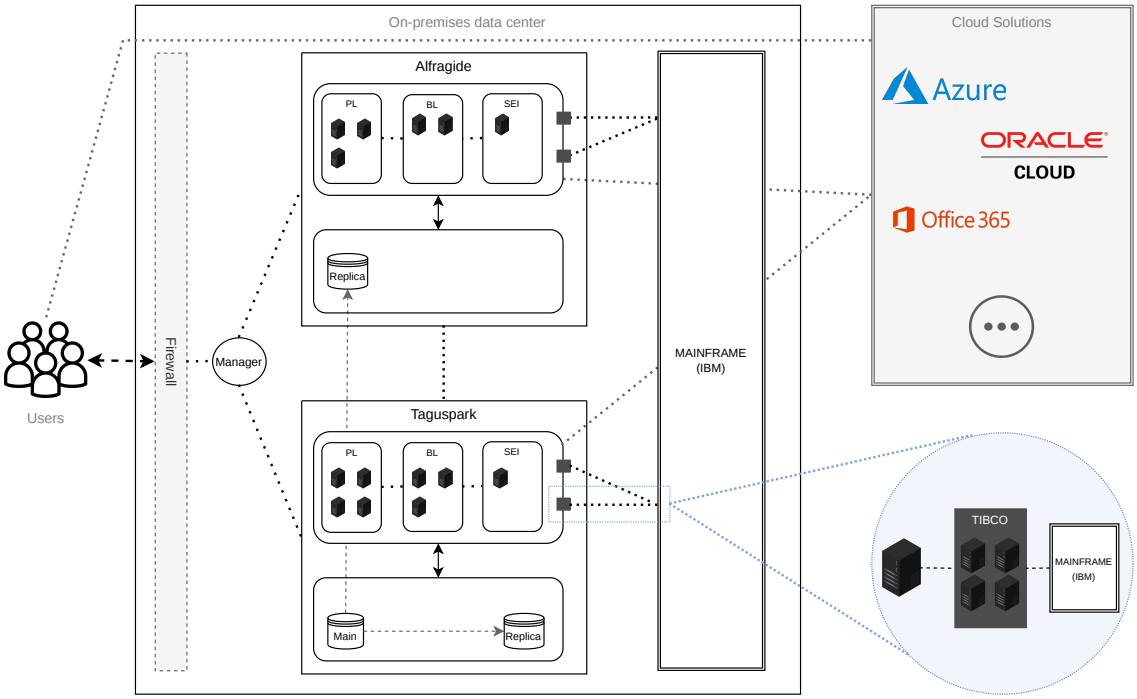


Figure 2.1: *Novobanco* DC structure.

Focusing on Onprem data centers, there are two data centers managed and maintained by *novobanco* (Taguspark and Alfragide) and an additional one (Mainframe) that IBM maintains. The data center located in Taguspark is the main one and is always active, and the one located in Alfragide works as a backup data center and is not always active. Nevertheless, the database of the Alfragide data center is always active to provide a secure and updated backup of data. The internal structure of both data centers maintained by *novobanco* is the same and is organized in layers (PL, BL, SEI, BD). The first layer works

mainly as the front end layer, and as the layers go on, the operations made in each of them require more processing skills and security. To guarantee security, firewalls exist before entering the data center and between all layers (PL, BL, SEI, BD). Note that there are different servers for different operations, and Figure 2.1 just represents the overall architecture of *novobanco* data center.

Although this architecture already guarantees a high level of security, private information such as clients' accounts is not located in either Taguspark or Alfragide data center. Instead, this information is located in a data center owned by *novobanco* but maintained by IBM (Mainframe). As can be noticed looking at Figure 2.1, the information located in Mainframe can just be accessed through the Onprem Taguspark and Alfragide data centers using a set of servers called *TIBCO*. This *TIBCO* servers process the information to the maximum so that the Mainframe requests are simple and fast, without extensive operations processing. Moreover, *TIBCO* servers add an additional layer of security and work as an API to facilitate all the connections with the Mainframe. An example is when a user wants to log in to the *novobanco* app, he has to pass through the several layers and arrive at the *TIBCO* servers, where it will be processing what is needed to contact Mainframe with a request-reply request, and return if the login was successfully or not. The same logic applies to the bank account balance and other operations.

With the growth of cloud computing, *novobanco* has also started migrating some applications to the cloud. However, even though some applications can be accessed directly to the cloud by users, when a cloud application needs to access private information located in Mainframe, it needs to make a request to the Onprem data centers. After the request passes through the multiple Firewalls, the information is returned to the cloud application.

Finally, it is also possible to analyze that before the user's request arrives at the Onprem data centers, there is an *Manager*. This *Manager* aims to coordinate operations and, in case the Main data center (Taguspark) is overloaded, distribute the requests amount to either Taguspark and Alfragide data centers, always maintaining the coherence in operations.

In sum, we can notice that the internal structure has many Firewalls, and to access private information, many steps need to be guaranteed. This makes it difficult for a cyberattack to succeed, as there are many stages that cybercriminals need to pass unnoticed.

2.1.3 Challenges and power of machine learning

Data center networks are growing fast, and managing Data Center Network (DCN) is already a tedious process. This process will become tough to improve only by humans due to random patterns/behaviors in future networks. DC urges the need for automation, which can be reached with machine learning models for automated management and optimization of the network. These models could automatically optimize data center machines and spread the work between them, avoiding having data center resources

under or over-provisioning, as depicted in Figure 2.2. A good resource estimation model will lead to better planning, schedule, and management of DC, which saves money. In the specific case of *Novobanco*, and as will be possible to observe later in Chapter 3, what happens, as a rule, is to have a provisioning mechanism superior to what is necessary (over-provisioning) so that servers never fails. Thereby, the case of under-provisioning will be rare, and it is not expected to be found very often.

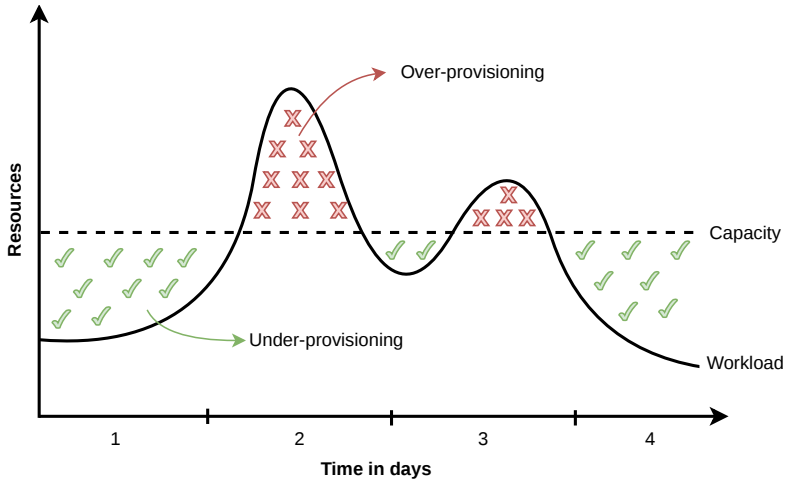


Figure 2.2: DC workload versus capacity along time. Inaccurate workload forecasting results in under and over-provisioning of resources. (Image based on [11])

However, machine learning algorithms in DC are not as simple in this field. Firstly, there are not so many real datasets that can be used, and one of the reasons is that companies cannot and do not want to expose their client's data and data center architecture mechanisms. Secondly, the traffic that passes in DCN is a mix of mice and elephant flows, but most traffic is mice flows sent before any optimization takes place. Because of this, to avoid models creating more latency on the network, it is necessary to choose machine learning model architectures that can process data and make decisions in order of microseconds. Another challenge in machine learning is that the data needed to train a machine learning model should not be shared within the network because it is too risky to let data leave its host because of privacy [11]. Studies show that some machine learning models in some points of the network are an easy target for security and privacy threats because data could be easily poisoned [12].

2.1.4 Related work

With the growth of machine learning, researchers have been developing machine learning methods to help manage DC. Several areas could help improve DC, but along this subsection, more importance will be given to critical areas that leverage machine learning and are relevant to our work (Workload forecasting and Traffic classifications). We will also, in Section 2.1.4.4, briefly talk about other approaches that have been made to optimize DCN using machine learning models.

2.1.4.1 Single model approaches

Diverse approaches have been made to try to predict DC workloads. Models have been designed based on statistical and machine learning domains, generally relying on regression, time-series, machine learning, and recurrent neural network [13, 14].

One of the most common approaches to DC resources forecasting is using time series models. Time series models use a sequence of data points over an interval of time to predict future outcomes [15]. Fang *et al.* used Autoregressive integrated moving average (ARIMA) to estimate CPU utilization [16]. Calheiros *et al.* also used ARIMA to predict the arrival rate of workload [17]. However, time-series approaches failed to make good predictions in unpredictable workload patterns because it is challenging to fit this kind of models in these situations [15, 18].

Jayakumar *et al.* developed the LoadDynamics, a machine learning model based on Long-short Term Memory (LSTM) [4]. LSTM models were used because they can efficiently learn long-term dependencies. With the right hyperparameters, these models can detect and predict various patterns in a workload [19]. The workload input is split into intervals, and each LSTM cell receives as input a workload interval to train this model. The authors used Bayesian Optimization to find the best hyperparameters for the model. BO is an iterative process that, in each iteration, searches for the best hyperparameters using a stochastic process, namely a Gaussian Process. After a fixed number of iterations, the best model is chosen and used as the final predictor.

The idea of Khan *et al.* was to create a model that could predict not only one DC resource but all resources that significantly affect the energy consumption, such as disk operations, CPU, memory, and network transmission rate [20]. They investigated several machines and deep learning algorithms and found that Gated Recurrent Unit (GRU) model was the one with lower errors values for all the estimated resources. This model allows predicting energy state in future points because the abovementioned resources significantly impact energy consumption.

Workload prediction depends a lot on the type of flows in the network. Classifying the types of flows allows better workload predictions because of the influence that large flows, which require a high transfer rate, have on the workload. Zhu *et al.* proposed SmartTrans, a system that uses deep learning methods for traffic classification (that affects the schedule in the network) and flows size rank prediction (that affects the flow completion time). [21] Estrada-Solano *et al.* also proposed a model to predict the type of flows in the network [22]. The model comprises an analyzer that analyzes and classifies the outgoing package and a learner that trains a model to help the analyzer classify the packages. In this case, the authors are dedicated to predicting elephant flows of great magnitude.

2.1.4.2 Multi-model approaches

Studies showed that a single model is not sufficient to lead with unpredictable patterns and behaviors that the network flow has [23–25]. A study conducted with 21 widely-used

prediction algorithms in four different workloads showed that the top predictors vary considerably for different workload patterns, as illustrated in Table 2.1. For different workload patterns, different models yield better estimations.

Table 2.1: Mean Absolute Percentage Error of the three best predictors depending on workload pattern. (Table from [24])

Increasing Workload		**On and Off Workload**		**Cyclic Bursty Workload**		**Random Workload**	
L-SVM	28%	**G-SVM**	22%	**ARIMA**	38%	**G-SVM**	45%
AR	29%	**ARMA**	30%	**BRDES**	41%	**LR**	46%
ARMA	30%	**L-SVM**	44%	**L-SVM**	43%	**L-SVM**	46%
Average	**51%**	**Average**	**69%**	**Average**	**75%**	**Average**	**52%**

Understanding this problem, Iqbal *et al.* proposed a novel method that adaptively and automatically identifies the most appropriate model to be used in some specific scenarios [23]. They used classical machine learning methods such as Linear regression (LR), Support Vector Machine (SVM), Gradient boosting tree (GBT), et al., to predict workload.

The proposed novel method is divided into two parts. In the first part ("Adaptive Prediction Method Learning"), the system receives historical resource utilization data divided into sliding windows as input. The system trains different machine learning models for each sliding window and predicts which model performs better. The selected features of the observed sliding window and the best prediction method are stored in the training data. In the second part, the training data created in this method's first part is inserted as input. The system prepares a Random Decision Forest (RDF) to predict the best model for a specific sliding window with training data. Once the RDF is trained, the system can predict the data center resource utilization in real-time for each sliding window, using the best model, considering the current window features. This system works by simply switching which model to use along time.

Liu *et al.* also proposed an adaptive prediction approach that categorizes the workloads into different classes based on workload features. The proposed method dynamically selects LR for slow-changing workloads and SVM for fast-changing workloads to estimate the CPU utilization of machines [25].

Differently from the two models aforementioned, Kim *et al.* proposed a method, called CloudInsight, that estimates future workload for the data centers by dynamically determining the weight of each predictor [24]. CloudInsight is inspired by the Mixture of Experts (MOE) problems [26]. The MOE's key insight is that a group decision made by all local experts is often preferable than a single expert's decision [27]. However, unlike the general MOE approach that relies on a simple linear combination of all local experts, CloudInsight creates an ensemble model with different weights for each local predictor. During the training process, CloudInsight creates and adjusts an output vector ($n \times 1$)

with each predictor's weight based on their accuracy for the current workload. CloudInsight uses this vector to create the ensemble model based on the following weighted average equation:

$$\text{Ensemble model} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}$$

where w_i is the predictor i weight and p_i is the prediction from predictor i. CloudInsight gives higher weights to potentially more accurate predictors and lower weights to potentially less accurate predictors.

2.1.4.3 Importance of window size

Developed works also noticed the significant improvement that window size selection has in the workload prediction of DC. Workloads changes are highly volatile, and fixed-size observation windows cannot always capture the local trends in the data. Studies show that, if at every estimation step, an exhaustive search to find the best observation window size is done, it is possible to significantly reduce the prediction estimation error [5, 18, 23].

Dalmazo *et al.* proposed a system to estimate the network traffic using statistical techniques. This approach adjusts the window size based on the variance of the current and previous sliding windows [28].

Baig *et al.* created a system to find the best window size for an observation [5]. In the first part of this system (Training Data Set Preparation), the system receives as input historical resource utilization logs of a data center split in sliding windows with a fixed size. For each sliding window, the system does a linear search to find the ideal length of that window that minimizes the prediction error when predicting the next interval's resource consumption. The system builds training data with the data from each sliding window and the identified corresponding optimal window size. Once all sliding windows are analyzed, the system enters the second part (Resource Utilization Prediction). Here, with the training data from the previous step, a Deep Neural NetworK (DNN) is trained to estimate the best window size for a given sliding window data. After training, the DNN model predicts the best window size for the current sliding window observations.

Using different machine learning models and comparing the *Baig*, *Dalmazo*, and fixed-window-size (FixedW) method revealed that the system proposed by *Baig* could improve the prediction accuracy from 16% to 54% compared to other models. (A comparison of these three methods made in [5] is presented in Table 2.1).

2.1.4.4 Other works

Beyond workload forecast and classification, machine learning is being applied in other DC problems to optimize and automate networks.

Table 2.2: Comparison of resource estimation error of the four experiments using different models and datasets. (Table from [5])

Data set	Experiment	LR	SVM	EN	LASSO	RR	NNLS	Average
Alibaba	FixedW	0.61	0.57	0.53	0.56	0.61	0.68	0.59
	Dalmazo	0.83	0.91	0.82	0.82	0.83	1.00	0.87
	Baig	**0.40**	**0.38**	**0.36**	**0.39**	**0.37**	**0.41**	**0.39**
Materna	FixedW	0.55	0.54	0.50	0.53	0.55	0.58	0.54
	Dalmazo	0.61	0.75	0.60	0.60	0.61	0.65	0.64
	Baig	**0.49**	**0.49**	**0.46**	**0.47**	**0.48**	**0.42**	**0.47**
Bitbrains	FixedW	0.23	0.32	0.19	0.22	0.21	0.24	0.24
	Dalmazo	0.56	1.00	0.55	0.55	0.56	0.64	0.64
	Baig	**0.20**	**0.22**	**0.17**	**0.20**	**0.19**	**0.20**	**0.20**

Chen *et al.* proposed a model to make traffic optimization decisions locally (i.e., application server), named *AuTO* [29]. This model uses deep reinforcement learning to optimize traffic of the long flows, adjusting the sending rate at end-hosts and avoiding congestion in the receiver queue. Topology management, which consists in finding good links connection in the network to provide good connectivity, is also a big problem in DC that requires automation. Salman *et al.* proposed a model that uses Deep Reinforcement Learning, and according to the network state, selects the links to activate [30]. Different kinds of failures, such as server, link, switch failures, happen in DC. Arzani *et al.* created a system based on random forest models to identify causes of failures and anomalies in the network [31]. With the growth of DC, it is also essential to have security in mind. Tongaonkar *et al.* proposed SANTaClass, an unsupervised machine learning model that uses network traffic to generate protocol signatures [32]. This model can also identify encrypted communications and distinguish between VPN and non-VPN networks.

2.1.4.5 Conclusions

Several different approaches have been made to try to predict workload in DC. In this Section (2.1.4), we started by presenting some single models that predict workload. Even though these models present promising results, studies showed that a single model is not enough to adapt to all unpredictable changes that the network flow has. Adaptive models can address variable-nature workloads and adapt to the stochastic changes of workloads. Experimental results prove that adaptive methods that choose the best model for each sliding window outperform all other models that use a single model to predict. However, a multi-model strategy bring the overhead problem to the discussion. On average, these approaches are computationally expensive and take more time to train and predict. Knowing that DC decisions have to be made fast, we must not disregard the single model approaches as these can lose a bit of accuracy but gain in computation and time.

Table 2.3: Comparison of techniques described in Related work.

Work	Observed Metric	Algorithms	Objective	Periodically re-trains
Fang [16]	CPU	ARIMA	Workload prediction	X
Calheiros [17]	Job requests	ARIMA	Workload prediction	X
Jayakumar [4]	Job arrival rate	LSTM	Workload prediction based on long-term dependencies	X
Khan [20]	Disk operations, CPU, Memory, Network	GRU	Predict DC resources that affects the energy consumption	X
Zhu [21]	Packets transmitted	Deep Learning	Flow type Prediction	X
Estrada-Solano [22]	Packets transmitted	Incremental learning algorithms	Large (elephant) flows Prediction	YES
Iqbal [23]	CPU	LR, SVM, KR, GBT, NB	Workload prediction by selecting the best predictor for the observed sliding window	X
Liu [25]	CPU	LR, SVM	Workload prediction by using (LR) for slow-changing workloads and (SVM) for fast-changing workloads	X
Kim [24]	Job arrival rate	21 prediction algorithms	Workload prediction by assigning a importance weight to each predictor	X
Dalmazo [28]	Network traffic	Statistical techniques	Adjusts window size based on the variance of the current and previous sliding windows	X
Baig [5]	CPU	DNN	Predicts the ideal length for the current observed sliding window	X

We also have to consider that machine learning needs continuous monitoring to learn new patterns over time, which can create latency in the network [23]. Excepting the Estrada-Solano *et al.* method, none of the works presented consider this aspect.

In the major part of these models, authors use CPU resource estimations, but models can be adapted to predict other resources. Though, studies show that CPU consumes 58% of the total host's energy, which makes it one of the most critical resources to be optimized [33]. Our research did not find any proposed system that uses interpretable models to explain its predictions (this will be discussed in subsequent Section 2.3). Table 2.3 summarizes and compares the main and more relevant works presented in this section.

2.2 Baseline machine learning models

To obtain baseline results as a base to the datacenter dataset used in this work, five models were chosen (Exponential Smoothing, Seasonal Autoregressive Integrated Moving Average (SARIMA), Prophet, Long-short Term Memory, Transformer).

2.2.1 Exponential Smoothing

2.2.2 SARIMA

SARIMA

-Augmented Dickey-Fuller unit root test.

Akaike Information Criterion (AIC) BIC Bayesian Information Criterion (BIC)

$$AIC = 2k - 2\log(\widehat{A}) \tag{2.1}$$

$$BIC = k\log(n) - 2\log(\widehat{A}) \tag{2.2}$$

2.2.3 Prophet

2.2.4 Long short-term memory

2.2.5 Transformer

2.3 Interpretable methods for machine learning models

2.3.1 Why to provide explanations

It is essential to explain predictions more than just present predictions because, in many situations, having a correct prediction is not enough to solve the original problem. In essence, an explanation is a "bridge"between people and a decision-maker that allows humans to understand the decisions [34]. For example, in a model that predicts a drug for a patient, it is essential to know why the model predicted that because it is necessary to provide explanations so the patient can trust and accept the decision. Explanations can be done by adding interpretability to our machine learning problems. The greater the degree of interpretability of a model, the easier it is to understand and accept certain decisions/predictions.

However, interpretability is not always needed in all machine problems because models are used in a low-risk environment in some situations, making predictions mistakes not significant. Moreover, interpretability also loses importance when the problem is well studied or if interpretability enables people to manipulate the system.

2.3.2 How to achieve interpretability

Interpretability can be done in several ways, and due to that, we can classify it using different criteria. For example, it is possible to apply interpretability methods that analyze the model after training (post hoc) or by analyzing the machine learning model itself (intrinsic). Intrinsic interpretability relates to models, such as short decision trees, that are considered interpretable due to their basic structure [35].

Besides that, we can interpret machine learning models by analyzing feature input and output pairs after training (model agnostic) or by analyzing internal model parameters and interpretation tools (model specific). Interpretability is also possible to be analyzed in a global way that tries to explain how the model works using features and all the learned components (weights and other parameters), or in a local way that consists of using a single or a group of predictions to analyze the model.

In the end, it is possible to observe that interpretability can be achieved in several ways. The most important is choosing methods that provide reasonable explanations for the problem being studied. This work will use mainly intrinsic interpretable models, analyzing them globally. In the next Section 2.3.3, we are going to talk in more detail about some of these models.

2.3.3 Interpretable Models

This Section will present some algorithms that create interpretable machine learning models. In Table 2.4 we present some of the main characteristics of the models that will be explained in more detail in this Section. To better understand this table, a model is considered linear if the relationships between features are modeled linearly. A model is monotone if an increase in feature values invariably leads to a decrease or to an increase in the target outcome, over all the model. Interactions refer to models that can automatically detect interactions between features. Finally, different models handle different tasks. Models can handle classification, regression, or both of these tasks.

Table 2.4: Comparison of different interpretable models. (Table from [35])

Algorithm	**Linear**	**Monotone**	**Interaction**	**Task**
Linear regression	Yes	Yes	No	Regression
Decision trees	No	Some	Yes	Regression, Classification
RuleFit	Yes	No	Yes	Regression, Classification

2.3.3.1 Linear Regression (LR)

Linear regression (LR) is one of the most used and well-understood algorithms in machine learning that allows features to be numerical and categorical. LR models produce transparent results, and mathematically it is not very complicated to estimate the weights.

Interpretability in these models is very simple and easy to understand/describe due to the linearity of the learned relationship. It is also possible to create visualizations with models weights to have an analysis that is quick and easy. Medicine, Sociology, and other fields use these models to understand why a particular prediction was made.

In LR, the target is predicted through the weighted sum of each feature input. This means that there are no interactions between features, and a change in a feature by just one unit directly influences the prediction outcome, making it possible to observe the impact of a single feature in the model. However, this can lead to mistakes because, for example, in a house price prediction, increasing the number of rooms without increasing the house size is, in some parts, unrealistic. In these cases, interpretation becomes more complex in LR models.

LR models impose predictions as a linear combination of features, which can be seen either as a strength or a limitation. Also, good LR models need the prediction outcome to follow a Gaussian distribution and assume feature independence. However, accomplishing all these requirements in many real-world problems is practically impossible. Because of this, the statistical and machine learning community has developed variations that allow improving LR models.

Generalized Linear Models (GLMs) were created to deal with LR models where predictions don't follow a Gaussian distribution. The idea behind GLMs is to keep the weighted sum and allow a different outcome distribution function linked to the weighted sum through a possibly nonlinear function. For example, predicting the amount of workload of a DC using a linear model can lead to negative predictions because the outcome assumes a Gaussian distribution. In these cases, we could choose a different distribution outcome by having a link function that always leads to positive predictions.

To deal with the non-linearity in the data, Generalized Additive Models (GAMs) were created. GAMs change the way targets are predicted. Instead of having a simple weighted sum, GAMs have a sum of functions for each feature. For example, in a DC, if we are predicting energy cost based on CPU consumption, the effect of increasing one unit when CPU consumption is low is different from growing one unit when the CPU is high [20].

When interactions between features are detected, we can create a new column that multiplies the two interacting features and solve this LR restriction. However, finding interactions by hand is not always easy, and some approaches automatically detect interactions as RuleFit. We will talk in more detail about this model below.

2.3.3.2 Decision Tree

Inspired in human decision-making, Decision trees are a technique that uses a tree-like model of decisions and can be used for classification and regression mainly in situations where models fail because of feature relationships in data, features interaction, or the model outcome is non-linear. Decision trees allow features to be numerical and categorical. [36]

Decision trees are constructed upside down, with the root at the top and the leaves (where decisions are made) at the bottom. So, these models' natural visualization makes it easy to understand the feature importance and its relations. Construction of decision trees involves deciding which features to use, when to split features and when to stop the tree's growth. If not, we could end up with complex, unstable, and biased trees, resulting in overfitting and bad decisions. Analogously, real-world trees need to be treated carefully to look beautiful and healthy or can end up growing randomly and looking bad.

2.3.3.3 RuleFit

Friedman and Popescu developed RuleFit to find feature interactions automatically, maintaining interpretability simple [37]. RuleFit starts by learning a sparse linear model with the original features and creates new features where interactions in the data are captured. These new features are called decision rules, which are binary *IF-THEN* statements:

IF the number of rooms > 2 *AND* the age of the house < 15
THEN 1 ELSE 0 [36]

However, these decision rules are generated through different Decision Trees, so it is necessary to have an algorithm that generates a lot of different trees to be used in RuleFit (the random forest method is one possibility). Several trees like the one depicted in Figure 2.3 are generated, and from these trees, we can get the decision rules.

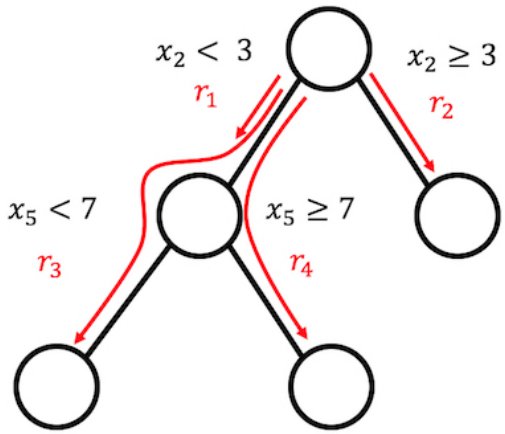


Figure 2.3: One decision tree generated in RuleFit algorithm. A simple tree with 3 leaf nodes can generate 4 rules (identified with r_1, r_2, r_3 and r_4) [37].

Many different rules are generated from decision trees, leading to many new data features (each rule becomes a new feature). Although, as can be imagined, not all new features are informative. In the next step, RuleFit trains a sparse linear model that selects

the best features of the data with the original and newly created features, resulting in a linear model.

Interpretation in these models is the same as interpreting a linear regression model with the difference that the RuleFit model has some binary rules features. As each decision rule is a feature, through the model weights, we can observe which decision rules have the greatest and lower importance in the model.

3

Novobanco data center

This chapter aims to explain the novobanco data center datasets used in this thesis. It is presented two datasets, one referring to the metrics of data center servers (the main one) and the other related to the number of logins in novobanco platform. We present an initial overview and an exploratory data analysis to understand the dataset's content better.

3.1 Data Overview

Recalling the diagram 2.1 explained in Chapter 2, we could observe that there a lot of servers and channels in *novobanco* data centers. As it is difficult to investigate all servers individually, we had to restrict our research to some servers with major importance for the problem we are dealing with. Consequently, we chose to continue our investigation with a set of servers located in Taguspark and used as direct channels. It was decided not to use data from the Alfragide data center, as this is not the main one. Figure 3.1 highlights the localization of the servers used. These servers were chosen because they are used for processing user requests (bank balance, credits, transfers, etc.) and are the ones with major importance in the bank. Recalling the structure of this data center, there are four main layers. The first layer deals mostly with the front-end part, the second layer processes most of the operations needed, the third layer processes mostly enterprise operations, and the last layer is the database layer. Considering this and as will be further justified, we will use the first and second layers as these are the ones that can be optimized and the ones that process the most important information. Contrarily, the third layer does not process so many processes, and the last layer is the databases and is not correct to turn on and turn off databases as this could create problems related to the database reliability and consistency.

Moreover it is also highlighted in Figure 3.1 the servers called *TIBCO* located in Taguspark data center. Even though the servers we will investigate more deeply correspond to direct channels, we will also use the number of logins in the different applications of *novobanco* to study the relation with Twitter.

In sum, for the servers in the first and second layers of direct channels, we will first create prediction models to predict the metrics of the servers, and after this, we will study the human-context impact on the usage of these servers. This will be the main data center dataset used during this work. Afterward, for the data related to the number of logins, the objective will be to investigate if the human-context influences the number of times customers and employees log in to the applications and sites of novobanco.

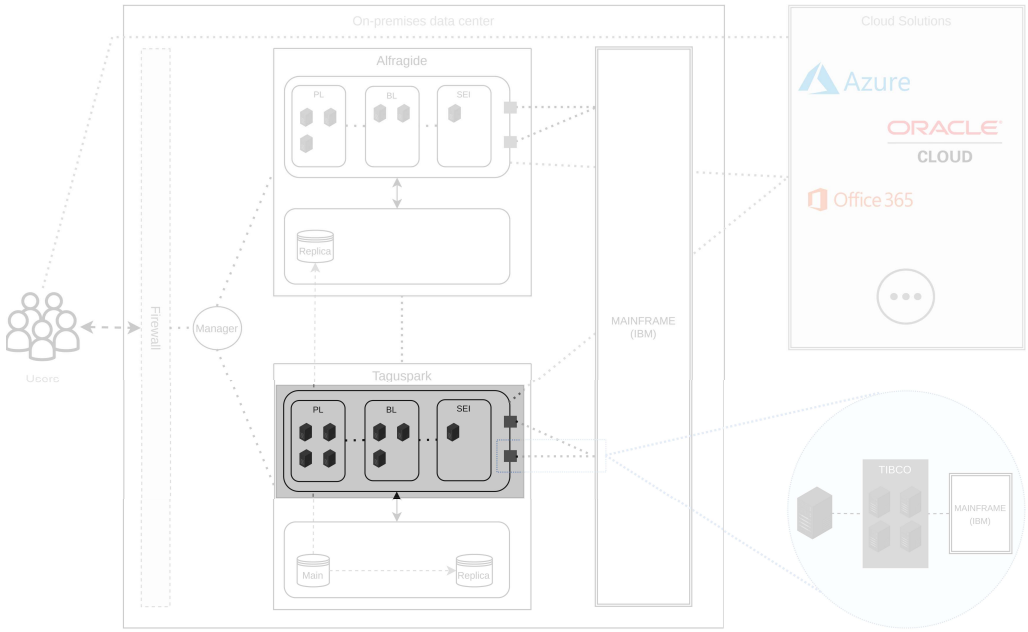


Figure 3.1: *Novobanco* DC structure with highlight in the servers used in this work.

3.2 Direct channels dataset

3.2.1 Data source, format and pre-processing steps

The metrics monitoring mechanism implemented in *novobanco* servers is `Nagios`[1]. `Nagios` is a monitoring application of all mission-critical infrastructure elements such as services, operating systems, network protocols, systems metrics, and network infrastructure. It is a system developed in `Nagios`, but can run in other `Unix-like` environments. In the specific case of server monitoring in *novobanco*, each server has a file for each metric (CPU, RAM, read and write operations, etc.) and can be exported in an RRD file format.

RRD is a format commonly used to handle time series data related to metrics monitoring, and the data is kept in a database with circular buffers. Thus, the time range is constant over time (in the case of *novobanco* the last record is from 1460 days ago, counting at the time it is observed). Figure 3.2 illustrate how data is stored in this kind of

[1] `https://www.nagios.org/`

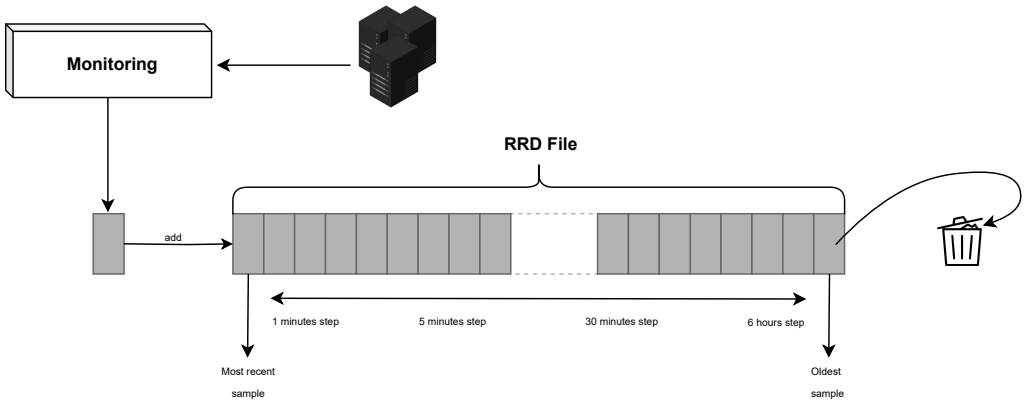


Figure 3.2: RRD file schema.

files. As can be observed, there is a monitoring node that assumes a time-variable range called step (in case of *novobanco* this step is of one minute), and in each step, it saves the metric value and erases the last sample in the RRD file. However, despite adding a new sample to the file every minute, the step is not one minute in all RRD file. Instead, as the time goes far away, the step increases firstly to 5 minutes, after to 30 and finally to 6 hours. In sum, a new sample is added every minute. As time goes away, it starts to compress samples into bigger time ranges (note that these intervals can be changed in the setup phase, and the available time ranges and the number of records available can be defined manually). Table 3.1 Shows the number of days available for every time range. As observed, the number of available days increases as the range increases. For the rest of this work, we will restrict to the 30 minutes and 6 hours ranges as the 1 and 5 minutes ranges do not have sufficient days to observe trends and seasonality, and so on to have power models. Also, the day the data were taken was 9 August 2022, so the beginning range of days will vary depending on the time step in observation. For example, for the time step of 30 minutes, we will have records from 11 May to 9 August 2022, but for the time step of 6 hours, we will have records from 10 August 2018 to 9 August 2022.

Table 3.1: Dataset overview ranges (Data was taken in 9 August 2022).

Time Step	Number of days	Starting day	Number of records
1 minute	Last 2 days	07-08-2022	2880
5 minute	Last 10 days	30-07-2022	2880
30 minute	Last 90 days	11-05-2022	4320
6 hours	Last 1460 days	10-08-2018	5840

The problem with files in RRD format is are not human readable and cannot be understood with normal and commonly used editors. Because of this, we had to make use of `rrdtool`[2], a tool made for handle, manage and visualize time-series RRD files over Linux, and was later adapt for the PHP and Node.js languages. Over the different methods that

[2] https://oss.oetiker.ch/rrdtool/index.en.html

`rrdtool` provides, we made use of `dump` that convert the content of an `RRD` file into an human readable `XML` format. With the files in `XML` format, we create a python script that reads and organized the content of each file in an `csv` file to be easy to work and explore in the next phases.

3.2.2 Exploratory data analysis (EDA)

Several metrics are being monitored for each server, and as we deal with multiple servers, we have multiple files to organize and study. Table 3.2 does an overview of the dataset, and in Table A.3 is provided a more detailed description of features, their type, and which features were discarded (with the respective reason in Table A.4). Note that for all the graphs presented during this section, the data has a time step of 6 hours, except for the one involving the hours (for that, it was used a step of 30 minutes).

Table 3.2: Data center dataset overview.

Number of servers	Number of layers	Number of metrics per server (processed data)	Obtained on
21	4	29 (8)	9 August 2022

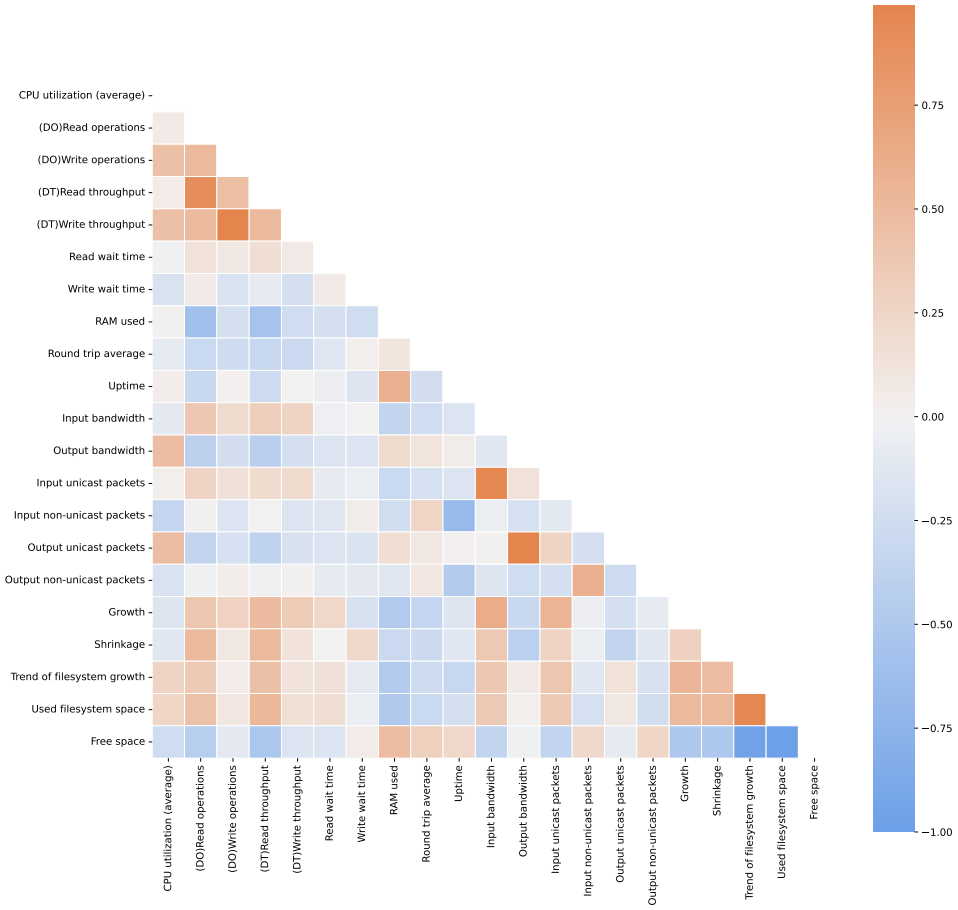


Figure 3.3: Heatmap with correlated features degree.

Observing that the amount of features available is high, we started by exploring correlations between features to reduce the feature space. For this, we calculate the correlation of all features available for every server individually and calculate the average of individual servers to understand the correlations common to all servers (note that the features available in each server are the same). Figure 3.3 shows the main positive and negative correlated features. Observing and studying this graph, we decided to discard features with a correlation bigger or smaller than 0,65. As they are correlated, these features will not bring additional information to our study and will increase the complexity of this study. Diagram of Figure 3.4 shows the features that correlate positively (in red) and negatively (in blue) with a correlation of more or less than 0.7, respectively. Also, from the correlated features, the features that were maintained are highlighted. From the 16 correlated features, 10 of them were discarded, reducing the feature space from 21 to 11.

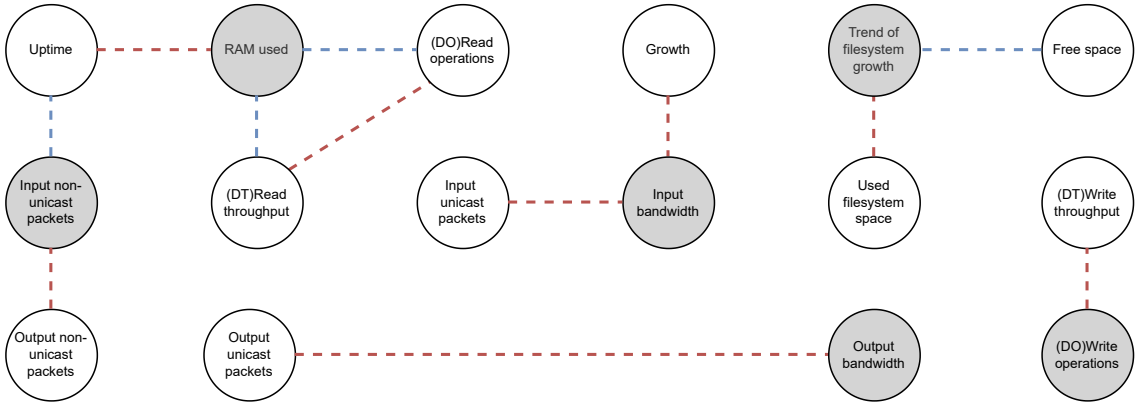
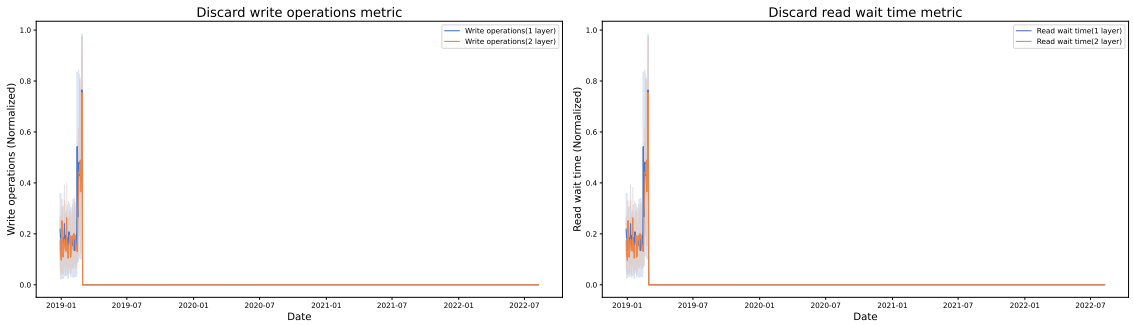


Figure 3.4: Correlated features diagram. (Red correspond to features positively correlated, and blue correspond to features negatively correlated)

Moreover, we decided to do a more internal analysis with the remaining features. With the remaining ten features, we investigated if there was missing data. Regarding the first layer, there was no missing data in any server concerning the 30 minutes and 6 hours step data. Regarding the second layer, in the 30 minutes step data, there were no missing values in any server feature. In contrast, in the 6 hours step data, we found that between 10 August 2018 and 29 December 2018, in six of the ten features maintained, there were more than 80% of null values in almost all servers. This suggests that during that time, there was a failure in monitoring data, or simply the monitoring was turned off. Because of this, and to have matching records in all servers of both layers, we discarded all the values between 10 August 2018 and 29 December 2018 of both layers, which resulted in the loss of 560 rows. After this, we normalized all the data and did some basic statistics and visualizations to understand the dispersion of these features. With this, we notice that features "Write operations"and "Read wait time"were producing strange values as they are significantly changing in the beginning and stabilized near zero since 2019-02-28 as can be seen in Figure 3.5. As the values stabilized, and we did not find an apparent reason for this, we decided to discard both features. The most likely cause is an error in

monitoring these two features, as this happens in all servers.



(a) Write operations metric over time.

(b) Read wait time metric over time.

Figure 3.5: Reasons to discard Write operations and Read wait time metrics.

The next analysis consisted in analyzing the servers through the different layers of *novobanco* data center. Figure 3.6 shows this analysis. As we can see, the CPU utilization varies a lot for different layers, which indicates that each layer should be studied individually to have better and more accurate results. Moreover, it is possible to observe that in the first and second layers, there is a server that is the major part of the time very near to 0. These two servers act as backup servers in case one server fails or where the requests increase a lot, and the other servers cannot deal with that amount of requests. Regarding the third layer, it is just composed of one server, and there is not a big possibility to optimize this layer. For the database layer, there is the main database, and the other two act as a backup of the main database, so the number of operations made in the ones that are not the main one decrease as was expected. With this analysis, we decided for the next phases to focus mainly on analyzing the first and second layers as these are the ones with more interest to study.

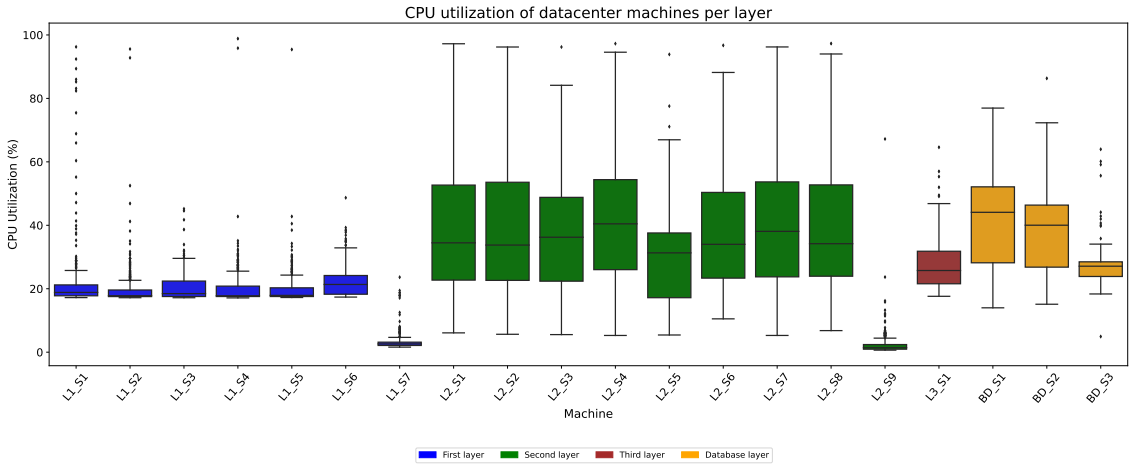
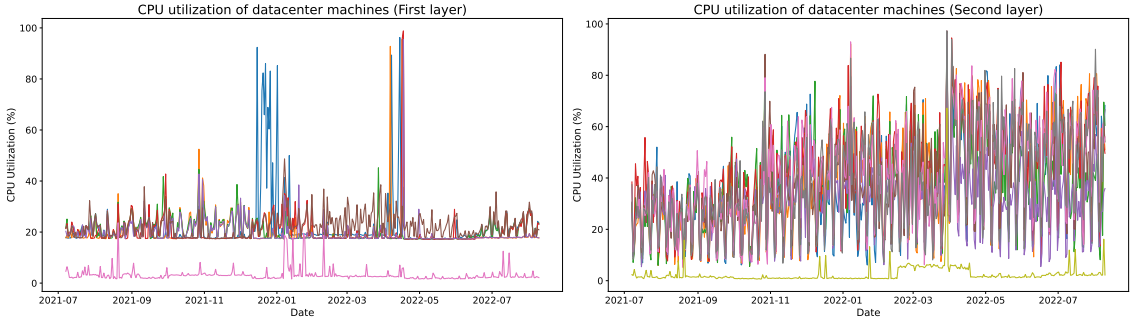


Figure 3.6: CPU Average over servers and layers.

Analyzing the CPU utilization over time of the first layer and second layer (Figure 3.7),

the difference in patterns can be noticed. As the first layer is mainly used for the front-end, the processing in this layer is not too big, but instead, in the second layer, where many operations are processed, we could notice a very high CPU usage. Also, as expected from the Figure 3.6 analysis, we have a server with almost no usage and just an increase in cases where the usage increases in other servers. It is also interesting to observe that in the first layer (3.7(a)), there was an increase at the end of 2021 and beginning of 2022 in one of the servers, and around April 2022, three servers had a peak of utilization. Analyzing this graph together with the previous one, and knowing that there is a server that acts as a backup in the first and second layer, we decided to discard these two servers (L1_S7 and L2_S9) of optimization because independent of any optimization that we want to do, knowing that we are dealing with direct channels of a bank, it will always be necessary to have a server to act immediately in case of any failure.



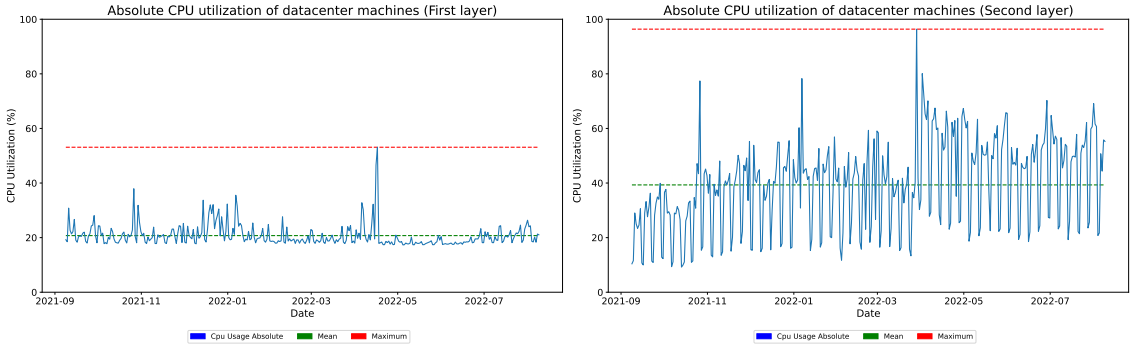
(a) CPU utilization of the servers of first layer.

(b) CPU utilization of the servers of second layer.

Figure 3.7: Cpu usage layers

The following observation (present in Figure 3.8) was to analyze the CPU utilization in absolute terms of the first and second layer (excluding the servers that serve as backup). With these graphs, we can observe that in the first layer, the CPU utilization never was superior to 53% and the mean value was around 21%, while in the second layer, it arrived at values near 98% with a mean of around 40%. Also, Figure 3.8(a) emphasizes the peak of April 2022 as the bigger one of the two noticed in the previous Figure (3.7), and this peak also corresponds to the time the CPU has the more prominent absolute peak in the second layer. The final remarks of this graph reinforce that while there is not a considerable variance in the first layer and the CPU used is not very high, the second layer is the opposite.

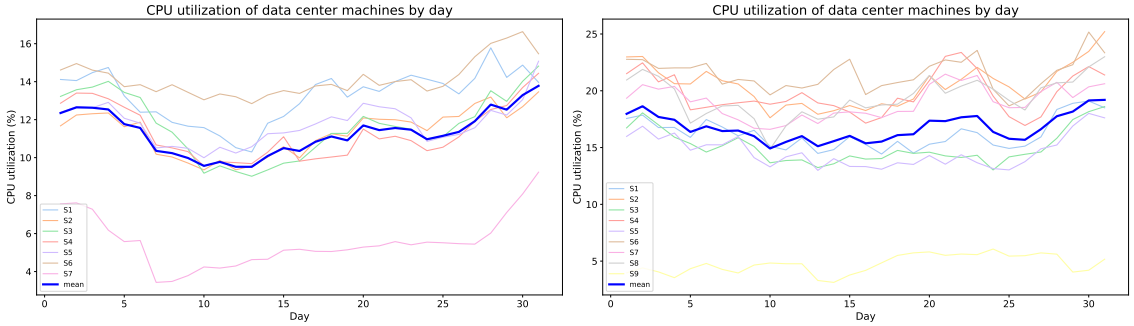
With a better understanding of the general usage and variance of servers of each layer, we decided to investigate trends related to the hour of the day, the month or the day of the week. Behind this is that users will probably have time zones that use more frequently and other time zones that use more infrequent applications of *novobanco*. In the graphs that are going to be explained next, each line corresponds to a trend of a individual server, as in charged dark blue color it is presented the mean of all servers.



(a) Absolute CPU utilization of first layer. (b) Absolute CPU utilization of second layer.

Figure 3.8: Absolute CPU utilization of servers (excluding the ones that serve as backup).

Starting by observing the CPU utilization over the month days (Figure 3.9), we can see that in both layers, there is a slight increase in the beginning and ending days of the month. This can be related to days when people receive their salary (in Portugal is generally at the end of each month), have to pail their house bills, or organize their finances for the next month.

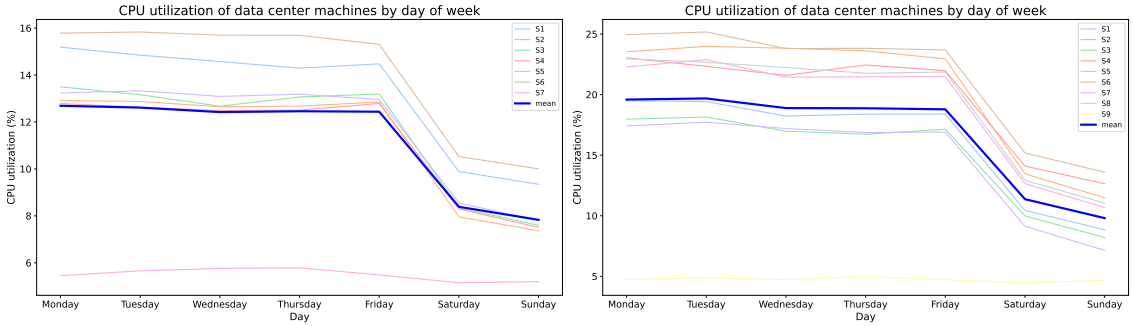


(a) CPU utilization of first layer by day of month. (b) CPU utilization of second layer by day of month.

Figure 3.9: CPU utilization of data center machines by day of month

Regarding CPU utilization by day of the week (Figure 3.10), we could notice that in both layers, there is constant CPU usage during business days. Contrarily, on the weekends, there is a significant decrease in CPU utilization. This was a curious observation as we expect that these servers' usage could increase as some people like to organize their economies on weekends, but these graphs did not show this.

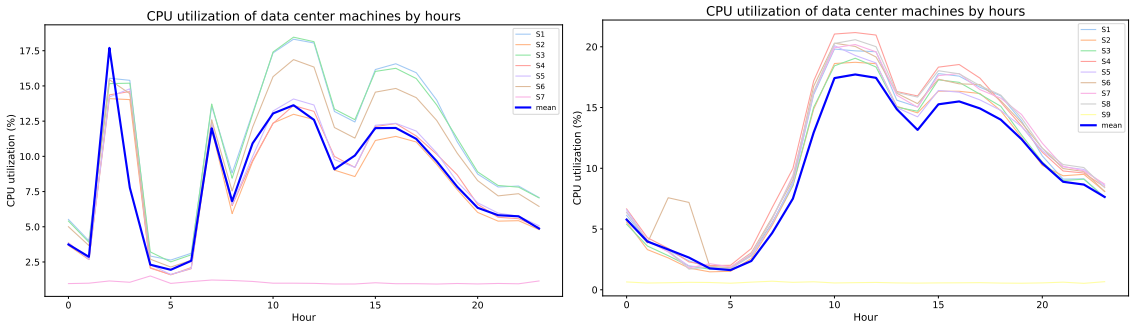
After analyzing the CPU utilization by day of the week, we analyzed the CPU utilization by the hours of the day in the first and second layer, depicted in Figure 3.11(a), 3.11(b) respectively. Analyzing these graphs, we could analyze some clear trends in data. Regarding both graphs, it is possible to observe a growth in usage in the morning and starts to decrease in the late afternoon. During this time, there is a slight decrease in usage around 1-2 pm, which may be correlated with the usual lunchtime. Moreover, in



(a) CPU utilization of first layer by day of week. (b) CPU utilization of second layer by day of week.

Figure 3.10: CPU utilization of data center machines by day of week

the first layer, two curious peaks happen over all servers at daybreak (around 3 and 7 am), and we did not find any apparent reason.

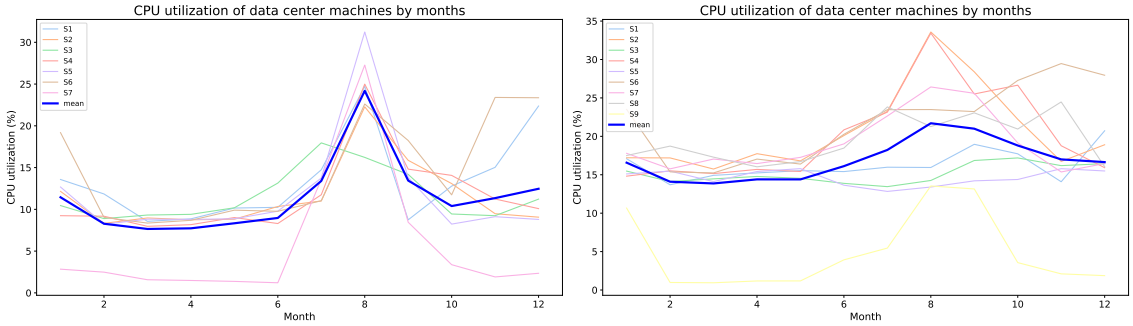


(a) CPU utilization of first layer by hours. (b) CPU utilization of second layer by hours.

Figure 3.11: CPU utilization of data center machines by hours

Some interesting trends happen regarding the CPU utilization by months (Figure 3.12). It can be observed that August is where there is a peak in both layers, which can be explained because it is most people's vacation month, so more operations and bank balance inquiries are made. Moreover, a peak could be observed in the final and beginning months of the year (December-January). This probably happens because of the festive seasons like Christmas and the new year, and it is known that this season involves a considerable cost of money.

In short, during the analysis in this section, we strict a lot to CPU usage despite having many other features available. We did this because the CPU is the most crucial resource to predict and is the most related to server usage. With this analysis, we first concluded to discard optimization in the third and database layers as in the case of the third layer, there is only a server, so it cannot be optimized. In the case of the database layer, it is also tricky because replicas need to be synchronized. Turning on and off some databases is not viable because it would create data consistency problems. That is why we restrict



(a) CPU utilization of first layer by months.

(b) CPU utilization of second layer by months.

Figure 3.12: CPU utilization of data center machines by months

to the first and second layers, and in both these layers, we also observe that each layer has a server with a meager CPU consumption that serves only for emergency cases (as a backup). It is required to have these servers always there in case of an unexpected failure of some other server. To complement this analysis, in Annex II, we present some plots similar to the ones presented in this section but for other metrics.

3.3 Number of logins (*TIBCO*) dataset

The other dataset used in this work corresponds to the number of logins in the different applications of *novobanco*. This one will not suffer such deep research as the last one but will be interesting to investigate the correlation with the human context. This dataset was easily obtained through a *novobanco* platform through an SQL request, and the result was not required to preprocess like the other one. The data corresponds to logins between 1 January 2022 and 30 June 2022, aggregated in a time step of 1 hour, and it was not found any camps with missing values. This dataset corresponds to the logins of 4 applications (2 regarding *novobanco* "continental Portugal"and will further be referred to as the main application, and the other two regarding *novobanco* "açores"). Moreover, there are three different media channels (1 regarding the site, the other to the old app, and the other regarding the new app). Table 3.3 do an overview about the dataset.

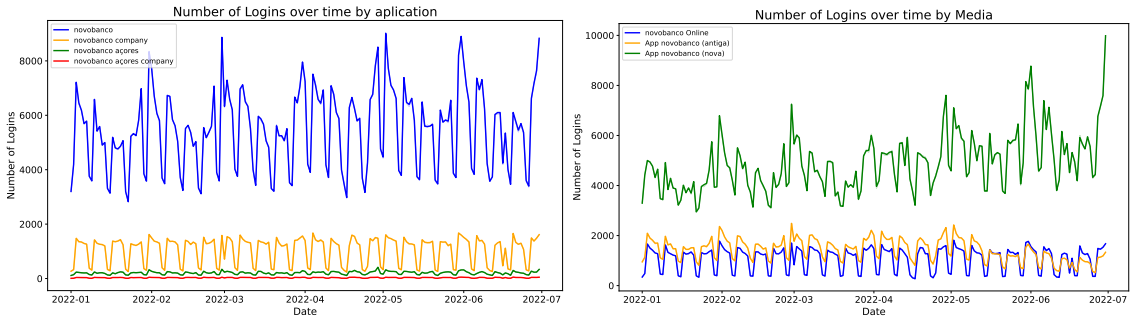
To understand this data better, we decided to investigate trends related to the hour of the day, the month, and the day of the week. In all Figures that will be presented next, there are two graphs, one regarding the number of logins according to applications and the other regarding the number of logins according to the media channel.

The first analysis was to study the number of logins over the days, as depicted in Figure 3.13. Observing this Figure, we can conclude that the *novobanco* main application has much more logins than the applications of *novobanco* "açores". Also, by observing Figure 3.13(b), we could see that the number of logins related to the new *novobanco*

Table 3.3: Logins dataset overview.

Application	Media	Description	Number of records
1	2	novobanco Online	4343
1	18	App novobanco (old)	4343
1	21	App novobanco (new)	4341
3	2	novobanco Online Company	4343
3	18	App novobanco Company (old)	4343
3	21	App novobanco Company (new)	3166
27	2	novobanco Online Açores	4324
27	18	App novobanco Açores (old)	4343
27	21	App novobanco Açores (new)	1180
29	2	novobanco Açores Online Company	3938
29	18	App novobanco Açores Company (old)	4269
29	21	App novobanco Açores Company (new)	85

applications is much bigger and has a more considerable variance related to the other media channels. Finally, with both graphs, it is possible to notice a clear seasonality trend that appears to be weekly.



(a) Number of logins per application.

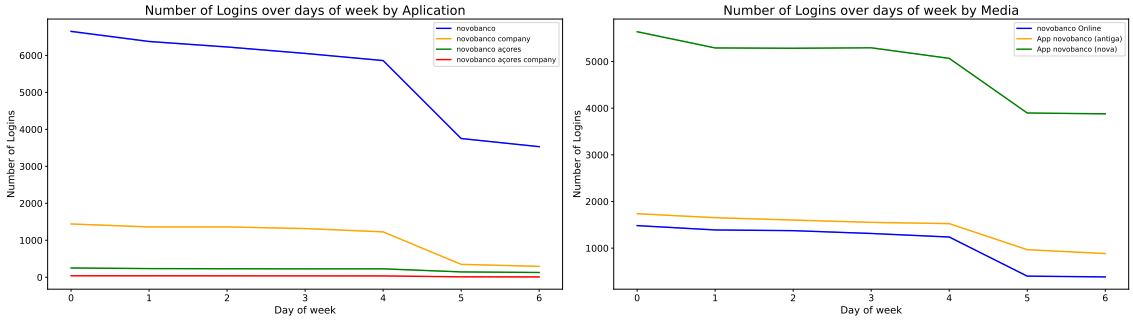
(b) Number of logins per media channel.

Figure 3.13: Number of logins in the last 6 months.

In the sequence of the previous Figure study, we decided to investigate the existence of any trends related to week. Graphs of Figure 3.14 show a decrease in weekend usage. It is also curious to observe that the number of logins generally decreases a little bit during business days when analyzing the graphs that refer to applications and media channels. This end to confirm the trend that has been initially noticed during the Figure 3.13 analysis.

Next, we decided to observe if the user login more in apps on some days of the month, depicted in Figure 3.15(a) and Figure 3.15(b). In general, the usage is constant all the days of the month with a general increase on the last and first days of the month, mainly in the *novobanco* new application.

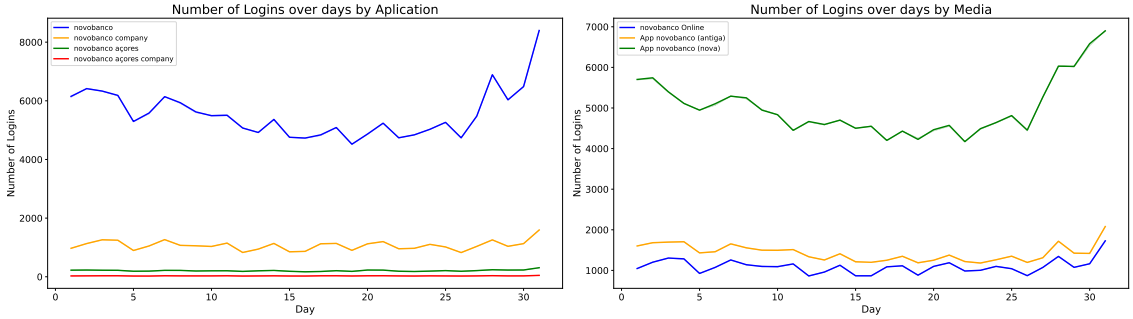
The last investigation done with this data was related to the hours of the day, as is shown in Figure 3.16. Similar to what would be expected, we notice an apparent decrease



(a) Number of logins per application.

(b) Number of logins per media channel.

Figure 3.14: Number of logins by day of week.

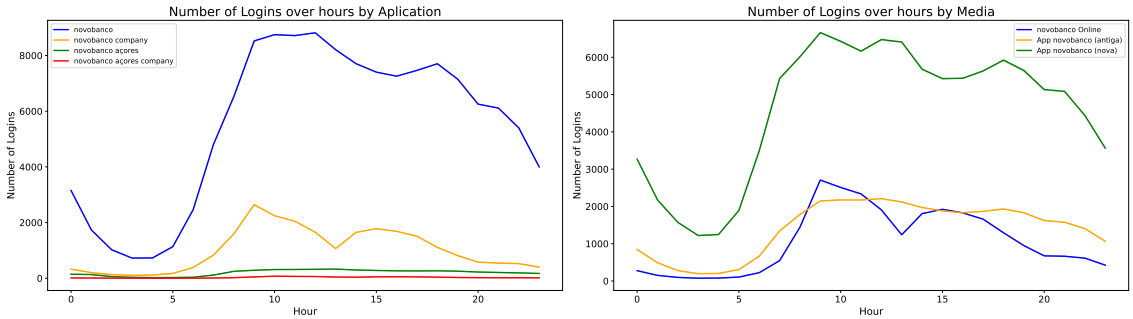


(a) Number of logins per application.

(b) Number of logins per media channel.

Figure 3.15: Number of logins by day of month.

in the number of logins during the night and the peak of usage in the morning. Regarding Figure 3.11(a) of the previous section (3.2.2) we noticed no apparent reason peak at 2 am. With the logins graph, we can conclude that the peak found previously has nothing to do with users' usage.



(a) Number of logins per application.

(b) Number of logins per media channel.

Figure 3.16: Number of logins by hours of the day.

Looking at graphs of trends of data center servers presented in the previous section

(3.2.2) and this one, we could notice apparent similarities in some graphs. Regarding the hours of the days (Figure 3.11 and 3.16), the trend is very similar but also simple to understand that it is correlated to the hours people wake and concerning the days of the week (Figure 3.10 and 3.14), we could notice that in both cases ate weekends there is a decrease of usage. Moreover, in the graphs with the day of the month (Figure 3.12 and 3.15), we can notice that the trend of having an increase of usage at the end and beginning of the month exists in both graphs, and mainly via main novobanco application. It makes sense to correlate with logins and CPU usage as users to log in need to pass through the servers of direct channels to arrive to *TIBCO*. However, servers of direct channels process much more information than just the logins, so it is normal not to have completely equal relations.

4

Datacenter analysis and results

This chapter describes the novobanco data center analysis with the specific results for models run. Firstly there is presented some baseline results having in account MAE and MSE metrics, and posterior, the results of the interpretables and final models...

4.1 Baseline Models Accuracy

Regarding the data center exploratory data analysis of Chapter 3, specifically, Figure 3.7, we could understand that each server has a specific pattern and besides the existence of some relations between servers of the same layer, along time the patterns change for each server. An example of this is Figure 4.1 that shows the CPU usage of one server of the first layer over time and highlights in grey different patterns in different regions. Having this as motivation, and knowing that this phenomenon occurs among all servers, instead of analyzing each server individually, we try to find examples of the main patterns that occur among the servers in analysis. This is, we tried to create slots of 400 days, in a way that each slot contains a pattern. To better understand this, we can look at each grey box of Figure 4.1 as one slot with a specific pattern (note that in this figure boxes are not in slots of 400 days, but the slots created posterior to analyze are).

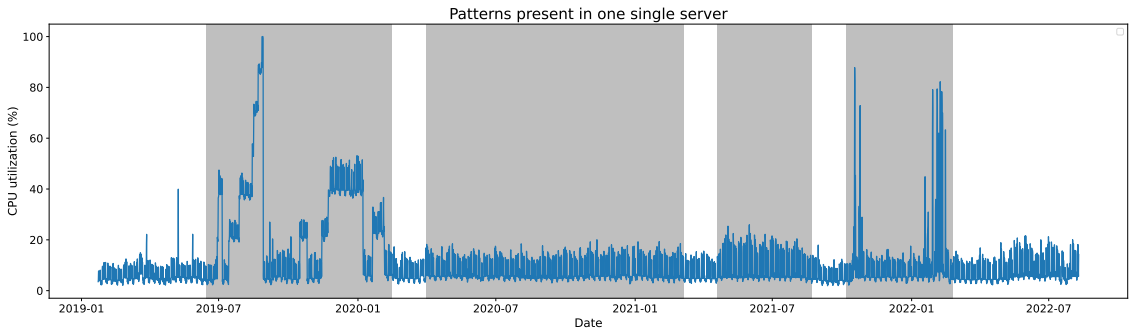


Figure 4.1: Patterns present in one single server. For each server, different patterns are found over time.

A 30-fold cross-validation approach was used to evaluate the baseline models' accuracy. We chose a such big cross validation to try to have accuracy's that reflect and covers different patterns of dataset.

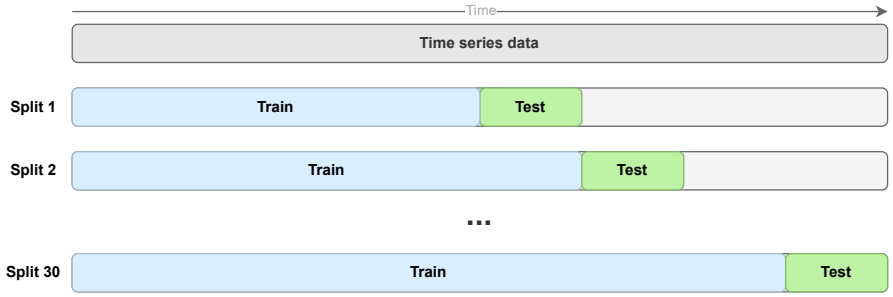


Figure 4.2: cross validation time series.

4.1.1 Seasonal autoregressive integrated moving average

4.1.2 Prophet

4.1.3 Exponential Smoothing

4.1.4 Long short-term memory

4.1.5 Transformer

4.2 Interpretability versus accuracy

5

Twitter data

This chapter aims to explain the Twitter dataset used in this thesis to understand the relation of tweets and novobanco data center resources utilization. We present how we obtained this data, describing and analyzing it.

5.1 Why is Twitter important

In today's society, social media started to be part of regular daily routine tasks. It is a valuable communication tool that can reach everyone, regardless of where people are. There are different kinds of social media, but they are generally used to interact, have fun, and access news and information [38].

Twitter is a real-time platform where people can freely express their opinions and ideas. Besides that, social media channels publish all the news on this platform to reach people quickly and easily. Data from Twitter is used to conduct several studies because Twitter is used by more than 320 million active users, which provides enough data on almost any significant trend or shift (for example, rise and fall of stock market prices) [39].

5.2 Tweets Extraction

Twitter's Developer Platform enables programmatic access to Twitter through *Twitter API*[1]. Through a set of programmatic endpoints, *Twitter API* allows developers to find and retrieve tweets, user information, trends, and so on.

To use *Twitter API* we created a Python program that accesses this API using authentication keys and tokens obtained from an *Twitter API* application. With these keys and tokens, header access to the API is created. The request is built through a function that receives the header and creates a request with the query we want to get and the remaining query parameters. *Twitter API* just allow making 300 requests per 15-minute window,

[1] https://developer.twitter.com/en/docs/twitter-api

and in each request, it is just possible to get 100 tweets. To deal with this limit, we insert a loop in our program to keep making requests until there are no more tweets to retrieve but using the Python function `sleep()` to delay the time between requests. *Twitter API* returns tweets in `JSON` format, that we convert to `CSV` format after search. This results in a `CSV` file with 17 features (detailed in Table A.2).

5.3 Tweets from all media channels

With the Python program prepared, we created a list with the leading Portuguese media accounts to extract tweets from those accounts. We defined the search since the beginning of the year 2014 because *novobanco* was created in that year, and in this way, we could investigate tweets since the creation of the bank. We collected data from 16 media sources, which resulted in 3 175 369 different tweets. Table A.1 list the media sources used presenting some characteristics of them (number of publications and number of followers).

5.3.1 Data preprocessing and cleaning

In this section, we will present the preprocessing and cleaning process to ensure that the dataset is well prepared to be analyzed. We started by checking if there were duplicate tweets in the dataset. We used the unique id that each tweet has, which resulted in zero repeated tweets. We explored data using features individually to understand whether errors, missing values, or special values exist. We also did some basic statistics, using some `pandas`[2] functions like `value_counts()`, `info()`, `unique()`, `describe()`, and performed calculations such as variance, mean, and standard deviation to correctly understand and interpret each feature. We conclude that the data in this dataset was well structured, and all empty/missing camps were filled with `NaN` values.

We noticed that the feature with the date was in `String` format, so we transformed this feature into a python `DateTime`[3] object, to make it easier to manipulate dates and times during Exploratory data analysis (EDA). Analyzing tweets language, we discover that 94,4% of tweets are in Portuguese, 3,3% in Spanish, and 2,3% in other languages, depicted in Figure 5.1. However, when we investigated tweets individually to understand why Portuguese media accounts have tweets in other languages, we found out that some tweets' text has some expression or word, not in Portuguese. In these cases, the automatic algorithm of Twitter that classifies languages is fooled and makes some mistakes. Because of that, we did not exclude the tweets that Twitter did not classify as not being Portuguese.

Some of the features of this dataset were discarded during the data cleaning process. Table A.2 presents which features were discarded explaining the reason in Table A.4 .

[2] https://pandas.pydata.org/docs/index.html
[3] https://docs.python.org/3/library/datetime.html

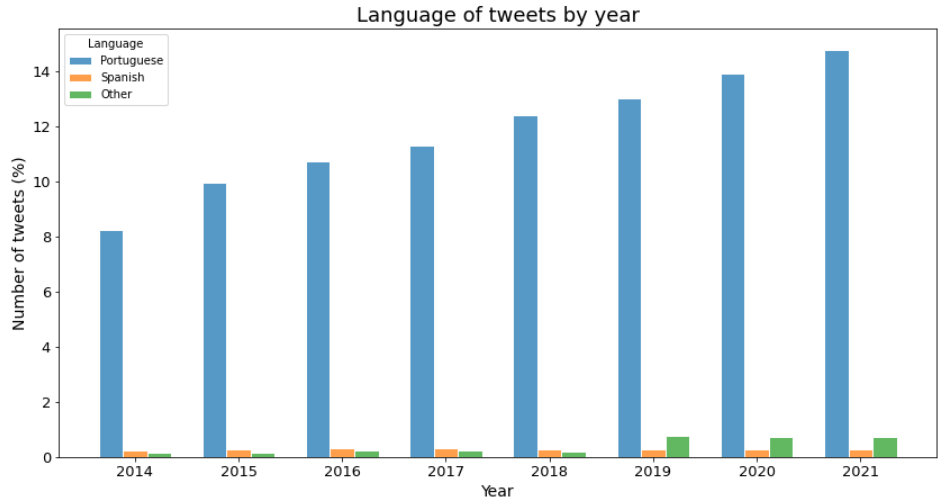


Figure 5.1: Distribution of tweets of the main languages between 2014 and 2021. Data extracted from mass media tweets search.

5.3.2 Exploratory data analysis (EDA)

Exploratory data analysis (EDA) is an approach to analyzing data, where it is possible to discover patterns, identify errors in data, detect and understand patterns in the data, and find interesting relations between variables. Exploratory data analysis (EDA) uses visualizations and statistical graphics to better and easier understand the dataset. To help us in EDA plots, we made use of `Matplotlib`[4] and `Seaborn`[5] libraries.

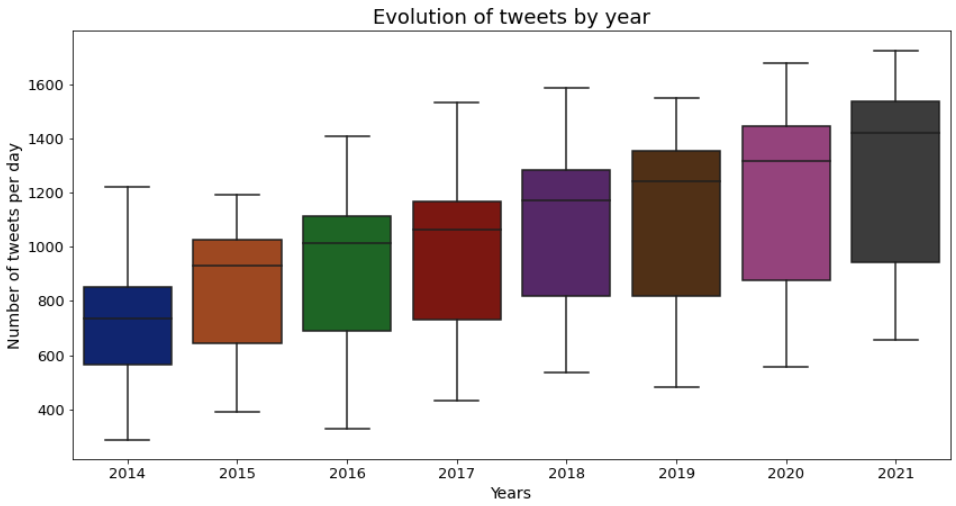
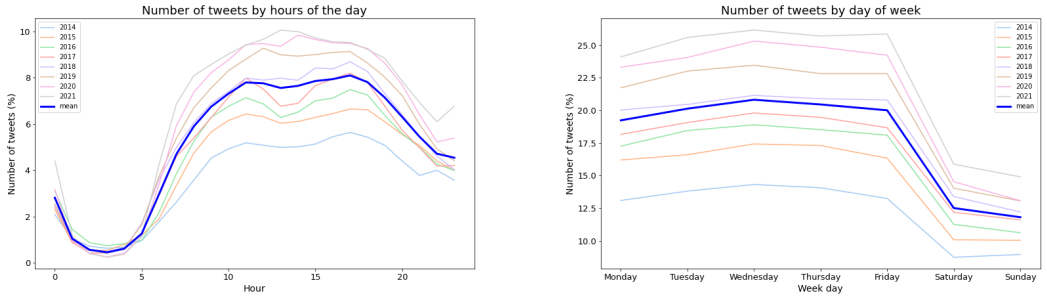


Figure 5.2: Tweets count per year between 2014 and 2021.

[4] https://matplotlib.org/
[5] https://seaborn.pydata.org/

Analyzing the tweets over the years showed that the number of tweets per year has been increasing since 2014, as depicted in Figure 5.2. Besides, we could observe that, on average, the number of tweets per day has also been increasing since 2014.

After analyzing the tweets over the years, we analyzed the evolution of tweets per hour of the day, days of the week, and months of the year, depicted in Figure 5.3(a), 5.3(b) and 5.4 respectively. These graphs allow us to understand if there are any correlations of tweets with the time they are published. In these graphs, we added a line with mean percentage because it helps analyze the graph as a whole, having a better estimation of how the percentage of tweets fluctuates along all years, without being biased by years with more or fewer tweets. These graphs showed us that tweets from mass media accounts are mainly published between 8 am and 7 pm from Monday to Friday. This can be explained because these hours correspond to the Portuguese working hours. Referring to Figure 5.3(a), we can notice that around 1-2 pm exists a slight decrease in the number of tweets released, which can be because of lunchtime. As Twitter has a feature to schedule publications hours, we cannot be sure if the reasons presented are correct, although the probability is high.



(a) Relation of tweets with hours of the day.

(b) Relation of tweets with days of the week.

Figure 5.3: Relation between the number of tweets per year, in percentage.

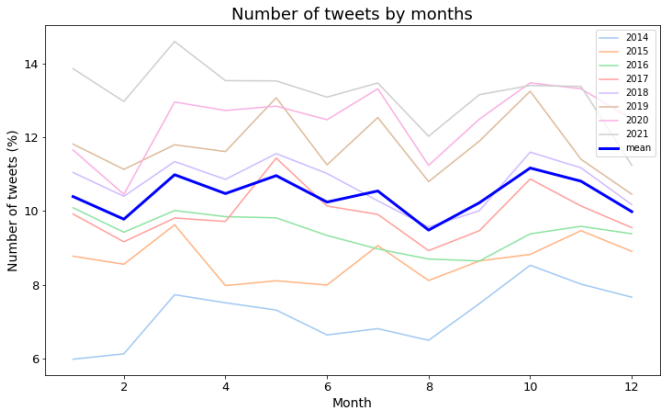


Figure 5.4: Tweets count per year between 2014 and 2021.

Contrarily, concerning Figure 5.4, we conclude that any significant correlations exist between the number of tweets and the month of the year. Just in August, on average, can be noticed a very slight decrease in the number of tweets that can be explained because this is the vacation month in Portugal.

This dataset also has a feature called `public_metrics` that has information about the number of tweet replies, retweets, and likes. Figure 5.5 shows the relation between news retweets and replies, with color change depending on the year. In this graph, we exclude some outliers to understand better the relation of these two variables in the more concentrated zone. By looking at this graph, we can see that both variables do not have a clear correlation. The more concentrated zone shows that most tweets in this dataset have less than 80 replies and less than 180 retweets. Also, by looking at colors, the number of replies increases with the year (tweets from recent years have more replies). For the number of retweets, any clear conclusion can be taken from the color change of this graph.

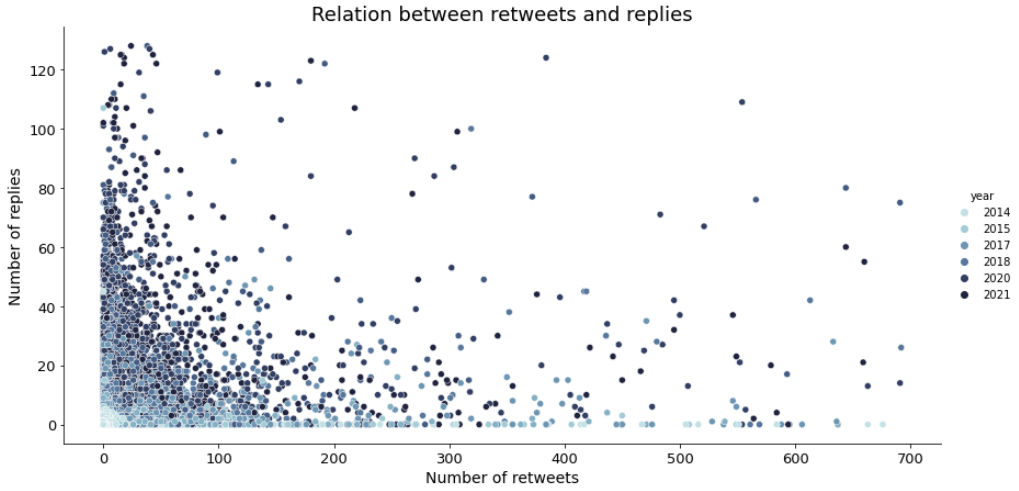


Figure 5.5: Relation between the number of news retweets and replies, grouping by year.

Another relation to be studied is between the number of retweets and the number of likes. Figure 5.6 shows this relation, having a color change depending on the tweet year. As in the previous graph, some outliers were excluded in this one. This graph shows that the oldest tweets have significantly fewer likes than the most recent ones. When the number of retweets increases, it is possible to notice a slight correlation with the number of likes, which means that if the number of retweets increases, the number of likes will increase as well.

Contrarily to what we thought and having in account the Figures 5.5 and 5.6, there is no clear relations between number of likes, retweets and replies. The only small correlation is between the number of tweets and the number of likes. Moreover, to better understand the variations of likes, retweets, and replies along the years, Figure 5.7 presents a graph that has the number of each variable individually along years. This graph support what was said before about the most recent tweets having more likes than the older ones (green line). Another interesting point is the tweet's replies line (orange) that is increasing

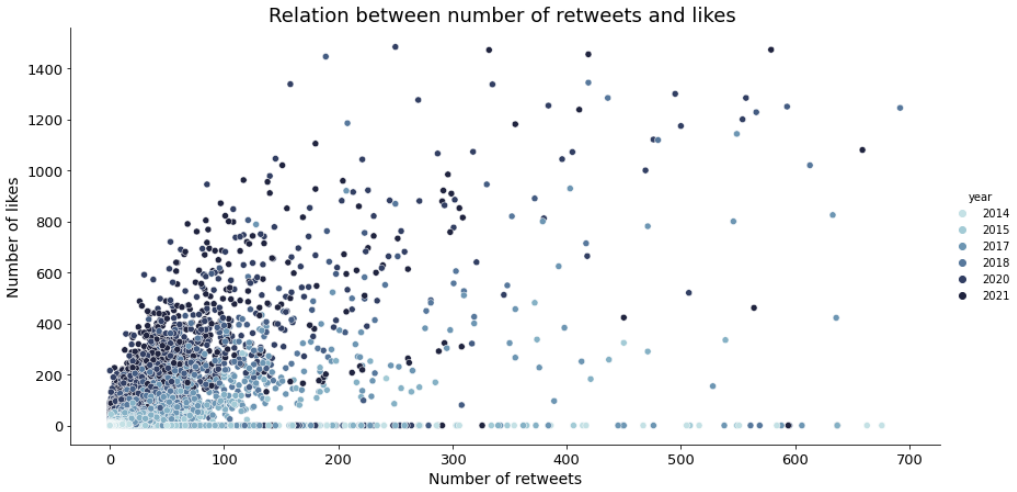


Figure 5.6: Relation between the number of news retweets and likes, grouping by year.

over the years but not so much compared to the increase in the number of likes. The same for the number of retweets (blue line), as this one has some variations in 2014 and 2017. Looking at these three tweets metrics (likes, retweets, and replies), it is clear that the metric gaining more prominence over the years is the tweet's likes. Like is one of the most straightforward metrics used in Twitter, explaining this increase.

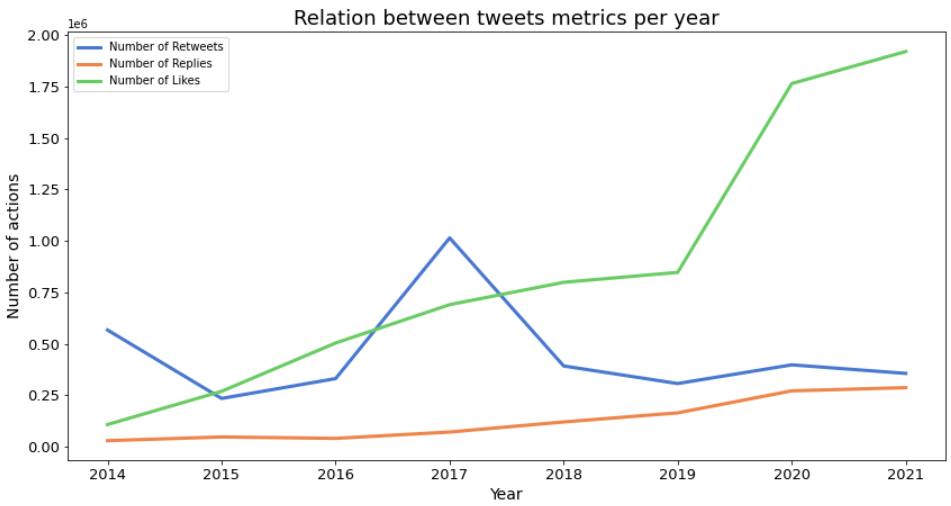


Figure 5.7: Relation between tweet's metrics (likes, retweets, and replies) over the years.

5.4 Tweets related to *novobanco*

It is possible to use different ways to obtain tweets related to *novobanco*. The first one, and one that has more reliable information, is to filter the tweets obtained by the mass media search presented in the last Section and filter tweets based on the content. Another possibility is to directly to a search query through *Twitter API* with keywords related to

novobanco. In these cases, we could select one or more keywords, and the search query returns all tweets that contain at least one of those keywords. Both of these ways will be presented in the following sections, auxiliary by some exploratory data analysis to understand each method better.

For both methods, it is necessary to select the keywords to get *novobanco* related tweets. From our search, we found out that the bank is manily refered in twitter as *"Novo Banco"*, *"novobanco"* and *"novobancopt"*. Since tweets with *"novobancopt"* word are already obtained with *"novobanco"* word, we selected *"Novo Banco"* and *"novobanco"* as our final search keywords.

5.4.1 Filtering Tweets from mass media search

Our first objective is to filter the data from mass media search presented in the last Section (5.3) to get *novobanco* related tweets, based on the keywords mentioned above. Filter tweets by keywords result in 12 743 different tweets representing around 0.004% of tweets in the initial dataset.

Analyzing the amount of media news related to *novobanco* per year, as depicted in Figure 5.8, we can see that the *novobanco* has been more referenced by the Portuguese media in some years than others. This bar chart shows that the number of tweets is not constant over the years, but 2014, 2018, and 2019 were years where the number of news related to the bank was smaller. *novobanco* was founded in August of 2014, so the lowest number of news in this year can be explained by that. During 2018 and 2019, we did not find any apparent reason for the number to be inferior, but during this time *novobanco* appeared to be in a stable situation. However, posterior years increase can be explained by the enormous losses that the bank presented and some polemics related to the offer of remuneration and bonuses for its managers despite the enormous losses that the bank was having [40]. To analyze twitter patterns, we created similar graphs to the ones shown

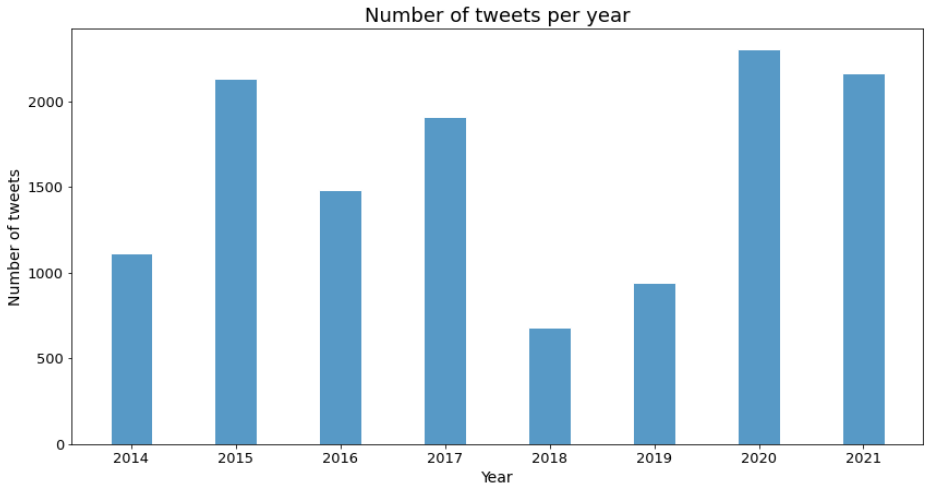
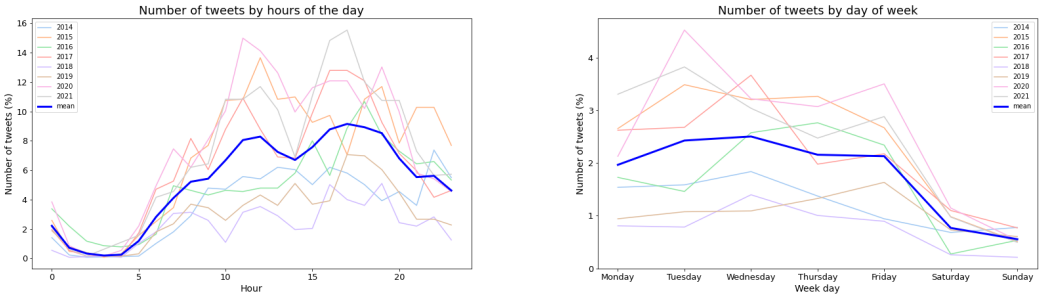


Figure 5.8: Number of tweets per year between 2014 and 2021.

before to study the evolution of tweets per hour of the day, days of the week, and months of the year, depicted in Figure 5.9(a), 5.9(b), and 5.10 respectively. For Figure 5.9(a) and 5.9(b), we can see that the mean line pattern is similar to the pattern obtained with the all media dataset (Figure 5.3). However, when observing the individual lines of each year, we can notice considerable variations between years that did not happen before. With the whole mass media dataset, the number of tweets was following a growing linear trend over the years, but when we filtered, we saw that the number of tweets related to *novobanco* changed from year to year. As it happened before, Figure 5.10 shows that does not exist a clear correlation between months and the number of tweets. We noticed that number of tweets increased in 2014, 2015, and 2020 during August and September, and on May 2021.

Another aspect of interest is the line corresponding to 2014 in this graph that only starts having tweets from July. This is because *novobanco* was founded just in August of 2014, and probably until the official foundation, few people knew about this new bank. This also proves the reason why in 2014, the number of news related to *novobanco* was lower than in other years.



(a) Relation of tweets with hours of the day.

(b) Relation of tweets with days of the week.

Figure 5.9: Relation between the number of tweets per year, in percentage.

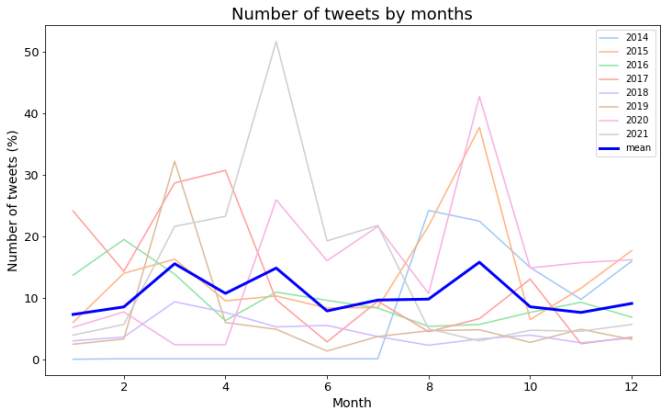


Figure 5.10: Tweets count per year between 2014 and 2021.

5.4.2 Search tweets directly with keywords

The other possibility to retrieve tweets from *Twitter API* is searching directly in Twitter for tweets that contain one or more keywords. This search resulted in 158 959 tweets from different sources (verified and non-verified ones). In this search, more than the news found in the last section, tweets from all users will be in this dataset. Because of this, tweets can have both reliable and non-reliable content. Besides that, this search results in multiple tweets that use slang expressions to address the bank. Although, we should not discard tweets from this search because what people write, even if they use slang, can influence the bank resources usage.

By analyzing the number of tweets of this search per year, we can conclude that there is no clear pattern evidence (Figure 5.11). Moreover, comparing the number of tweets from this search with the number of released news related to *novobanco* (presented before in Figure 5.8) showed that similar patterns were formed, which reveals that in times where there is more released news from some media sources about the bank, there are also more tweets in all Twitter network.

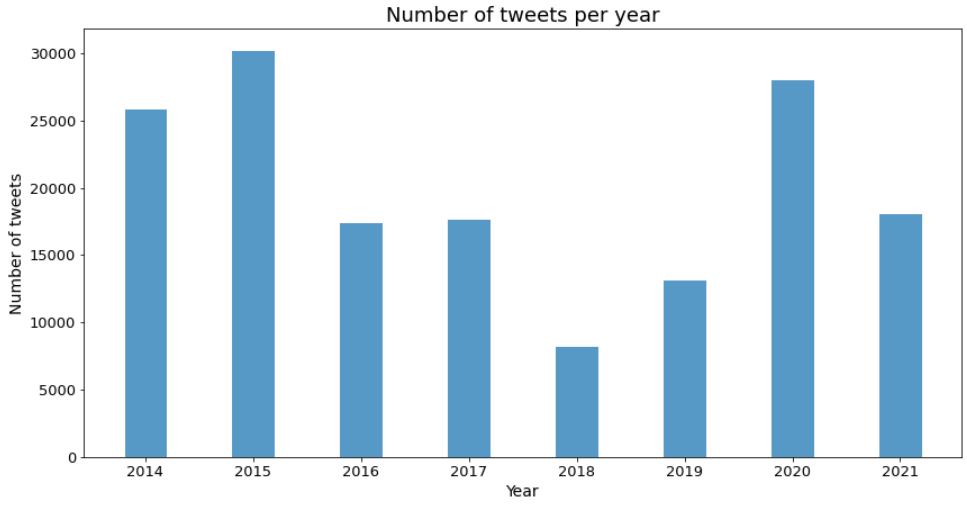


Figure 5.11: Number of tweets per year between 2014 and 2021.

This search resulted in only 75% of tweets in the Portuguese language. Figure 5.12 shows the number of tweets by languages over the years. Compared to graph 5.1, we can observe that this search significantly increased tweets of foreign origin. This happened because tweets from all world that contain the keywords inserted are retrieved.

An analysis of the evolution of tweets per hour of the day, days of the week, and months of the year is illustrated in Figure 5.13(a), 5.13(b), and 5.14, respectively. Figure 5.13(a) and 5.13(b) show that tweets are published mainly during working hours and days. However, it is possible to notice that in the 2014 year, this pattern was not followed because Figure 5.9(b) shows an increasing number of tweets during the weekend.

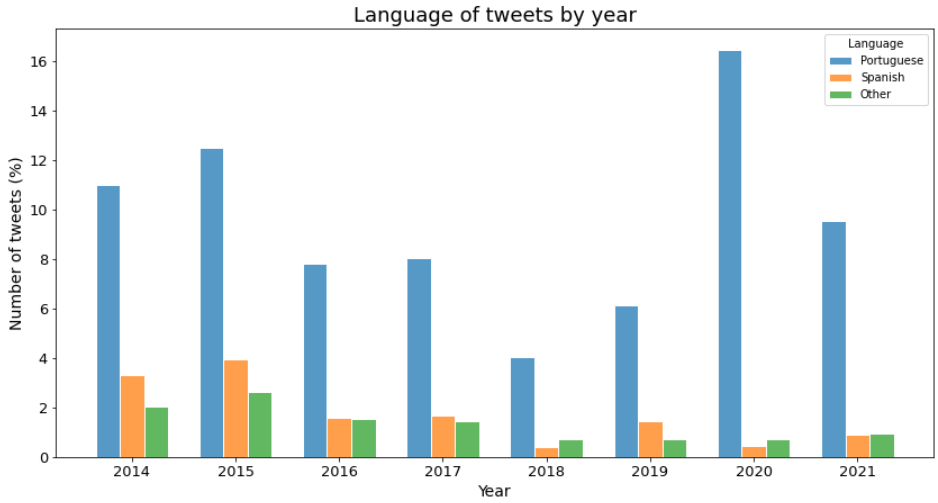
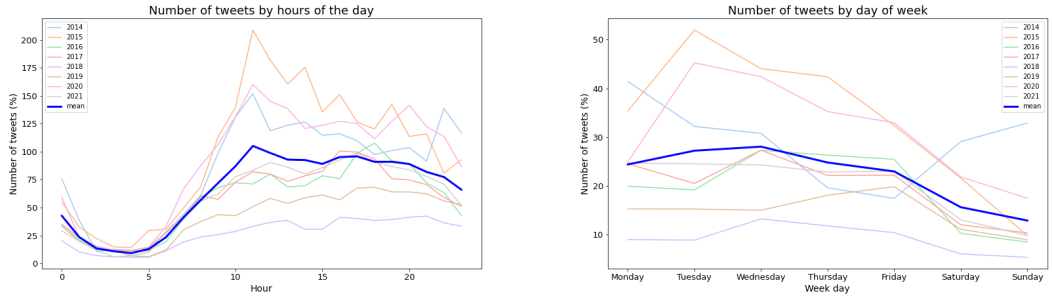


Figure 5.12: Number of tweets in percentage of the main languages between 2014 and 2021. Data extracted from twitter keyword search.



(a) Relation of tweets with hours of the day.

(b) Relation of tweets with days of the week.

Figure 5.13: Relation between the number of tweets per year, in percentage.

Looking at Figure 5.14, we can see a constant volume of tweets throughout all the months, except for August and September in 2014, 2015, and 2020. As mentioned before, the bank was founded in August of 2014, which decreased the total tweets number that year. The 2015 and 2020 years increase can be because some financial reports are published during September. This also happened in the dataset with media news filtered by keywords (Figure 5.10).

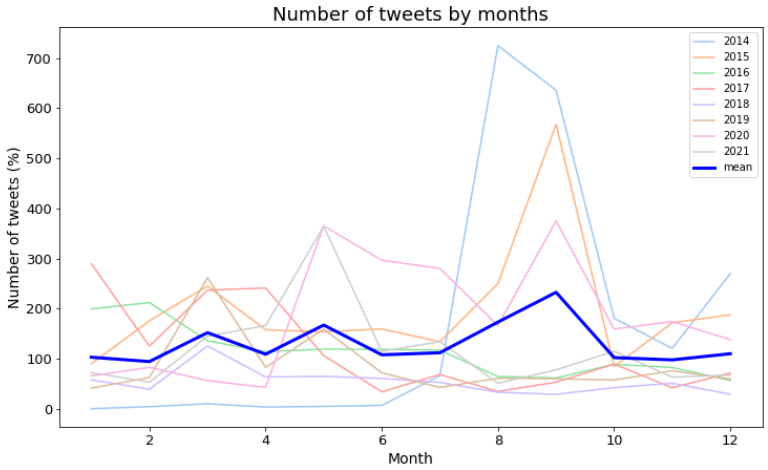


Figure 5.14: Tweets count per year between 2014 and 2021.

6

Interpreting the Twitter Data

This chapter outlines two different approaches, Sentiment analysis and Topic modeling, used to interpret the content of tweets related to novobanco. We start by presenting how we prepared the text of tweets to be analyzed, followed by the experiments with the methods used.

6.1 Natural Language Processing (NLP)

The previous chapter explained how we obtained tweets and presented some data exploration to understand the potential of their features. Although the insights this already gives us, it is also essential to look at the content of each tweet and understand if it has a positive or negative sentiment and what is the main topic present in them. For example, if we have 100 tweets and 90 of those tweets are positive, this can influence the model differently if those 90 tweets are negative.

Since this kind of data is typically too vast for a simple user to analyze manually, it is necessary to find ways to automate this process [41]. Computational linguistics, or simply natural language processing, is the subfield of computer science concerned with making a computer capable of learning, understanding, and generating human language content [42]. In this chapter, we used two natural language processing techniques, Sentiment analysis and Topic modeling. Concerning this analysis, it has just been used tweets related to *novobanco* from the mass media search (presented in 5.4.1). Using this type of tweet allows reliable and trustable information that can provide better conclusions and, therefore, better results for the models. Tweets from all users were not used in this chapter because they can sometimes negatively influence the results as some are baseless opinions and have many slang expressions.

6.2 Pre-processing of the dataset

Before we start analyzing the content of tweets, some pre-process steps need to be done. As shown in Figure 6.1, there are five main steps: *Cleaning the text*, *Tokenization*, *Reducing*

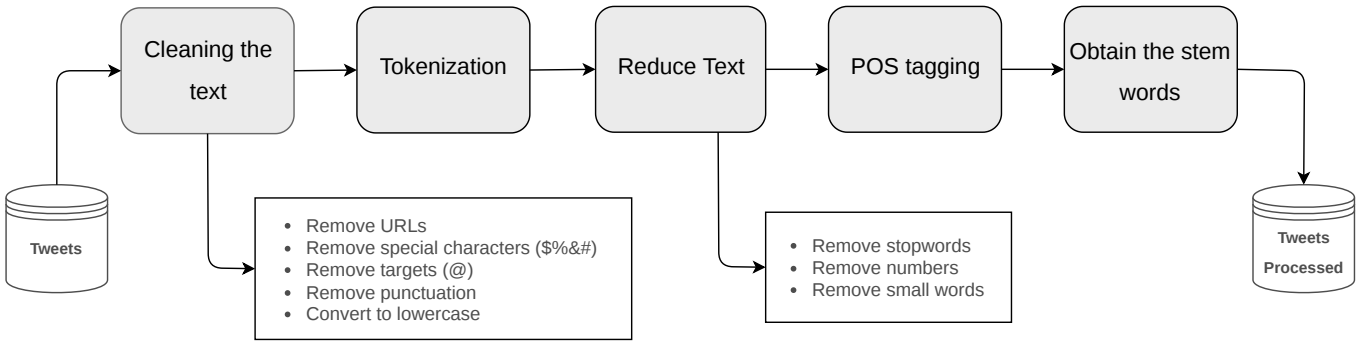


Figure 6.1: General steps use to pre-process texts before analyzing them.

text, *POS tagging and Obtaining the stem words* [43]. The first step removes all the special characters, punctuation, URLs, and mentions present in the text and, in the end, converts the letters to lowercase. Generally, in this first step, it is common to remove all accents. However, for our sentiment analysis, since we are dealing with Portuguese texts and all the Portuguese libraries of words have accents, we decided not to remove them to avoid creating any conflict. Contrarily, for Topic modeling, all accents were removed. After cleaning the text, we do the *Tokenization* step. This step consists of breaking the text into small chunks called Tokens and can be performed at sentences or word level. Tokenization helps interpret the meaning of the text by analyzing the word sequence [44]. For this, it was used the `word_tokenize` function provided by the `NLTK` [45] library. After Tokenization, we start the *Reduce Text* step that removes words that generally do not have much influence in NLP, such as stopwords, numbers, and words with less than two letters. We use `LX-Stopwords` [46] to remove the stopwords, a Portuguese dictionary composed of 2631 words created by the Natural Language and Speech Group of the University of Lisbon.

The last two steps consist of studying the words more grammatically and were done using *SentiLex-PT* [47]. In lack of words cases in this dictionary, we used an NLP package made for different languages named `Stanza` [48]. *Part-of-speech (POS) tagging* will associate a label indicating the grammatical category it belongs to, and the last step will convert each word to its stem form. Obtaining the stem words can be done using stemming or lemmatization. However, in this work, we decide to use lemmatization as this uses vocabulary and morphological analysis to identify the inflected forms of words (lemmas), and studies show that it can provide better results compared to stemming [49–51]. Figure 6.1 resumes the effect of each pre-processing step presented in this section.

Table 6.1: Effect of the pre-process steps in text.

Text cleaned	**Tokenization**	**Reduce Text**	**POS tagging**	**Lemmatization**
manifestantes entram na sede do novo banco	[manifestantes, entram, na, sede, do, novo, banco]	[manifestantes, entram, sede, novo, banco]	[[manifestantes, NOUN], [entram, VERB], [sede, NOUN], [novo, ADJ], [banco, NOUN]]	[manifestante, entrar, sede, novo, banco]

6.3 Lexicon-Based Sentiment analysis

Sentiment analysis can be defined as the process of classifying opinions, emotions, or attitudes from text, speech, or other sources [41]. Sentiment analysis allows us to understand the general opinion about something at a particular time. More concrete for this work, this analysis will provide essential information about the general satisfaction or not about *novobanco* at each time.

There are two main techniques for doing sentiment analysis: Machine Learning Approaches and Lexicon-Based Approaches. Machine Learning approaches create models that can classify text. In contrast, Lexicon-Based Approaches use dictionaries with sentiment scores and sum the sentiment of each word to understand if the words of a particular tweet are mainly positive, negative, or neutral [41]. Despite sentiment analysis already being a very developed process for the English language, studies dealing with the European Portuguese language and other languages are scarce. Because of this, several studies use Lexicon-Based Approaches or approaches that involve translating the original data to English and then using an English sentiment analysis tool. On the other hand, translation errors and language-specific information can have a significant impact on the final result because some context and specific semantics of each language are lost [52]. Another problem that needs to be present when doing sentiment analysis in languages other than English is the lack of suitable datasets to train and validate the models.

Considering all these problems and trying not to mask the specific syntax terms of the Portuguese language, our baseline approach will be lexicon-based, even though this type of approach cannot deal with domain and context-specific orientations.

6.3.1 SentiLex-PT

To do the Lexicon-Based Sentiment analysis, we start by using the *SentiLex-PT* [47] dictionary, one of the complete dictionaries for the Portuguese language because it was conceived taking into account the different syntactic-semantic contexts in which words can occur. *SentiLex-PT* is a dictionary conceived specifically for opinion and sentiment analysis for texts in European Portuguese, and is composed of 7,014 lemmas and 82,347 inflected forms. Each entry of this dictionary corresponds to one word/expression with information about the grammar category (adjective, noun, verb, or idiomatic expression), the sentiment attribute (1, -1, 0), and the sentiment assignment method (some of the words were labeled by software designed specifically for this purpose).

To classify tweets, we start by counting the number of positive ($nPos$), negative ($nNeg$), and neutral ($nNeu$) words present in each tweet regarding the words and expressions present in the *SentiLex-PT* dictionary. After this, to calculate the overall sentiment ($f(t)$) of each tweet (t), we decided in cases where $nNeu > nPos + nNeg$ to classify tweets as neutral, whereas in other cases we multiply the number of words of each sentiment by its correspondent polarity and sum all of those values ($nPos \cdot 1 + nNeu \cdot 0 + nNeg \cdot -1$).

Equation 6.1 resumes this method in a simplified way.

$$f(t) = \begin{cases} nPos - nNeg & \text{for} \quad nPos + nNeg \geq nNeu \\ 0 & \text{for} \quad nPos + nNeg < nNeu \end{cases} \tag{6.1}$$

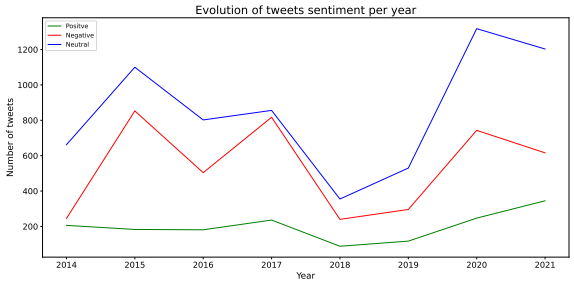
Tweets (t) with $f(t) > 0$ will be classified as positive, $f(t) < 0$ will be classified as negative and $f(t) = 0$ will be classified as neutral.

Knowing that this method fails when giving us the degree of the sentiment of each tweet taking into account the extension of all tweet text, we used another method that follows the equation 6.2. This method computes the ratio of the result of the equation 6.1 and the total number of words present in it ($count(t)$), returning values between -1 and 1. Values close to -1 indicate that the text is highly negative, and values close to 1 indicate that the text is highly positive. Applying this method in our dataset, we obtained values between -0.7 and 0.9.

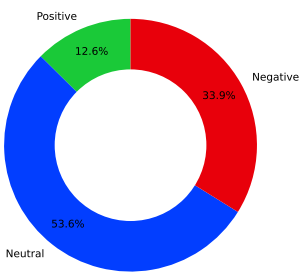
$$f2(t) = \frac{f(t)}{count(t)} \tag{6.2}$$

The advantage of this method is that it can provide some insights into how positive and negative a tweet is. Despite this, the number of positive, negative, and neutral tweets will remain the same as Equation 6.1 because what will define the polarity is the function $f(t)$. However, this method will bring further benefits in section 6.3.3.

Applying these Equations (6.1 and 6.2), from a total of 12743 tweets, we end up with 1603 positive tweets, 4314 negative tweets, and 6926 neutral tweets (Figure 6.2(b)). In Figure 6.2(a), we observe that the number of positive tweets stays low over the years, instead of negative and neutral tweets that change over the years. It is also interesting to observe that in 2017 the difference between positive and negative tweets was the smallest compared to other years, meaning that it probably is the year when more negative news related to *novobanco* appeared. Figure 6.3 shows a word cloud with the main words present in each sentiment classification.



(a) Evolution of tweets sentiment per year.



(b) Percentage of tweets per sentiment.

Figure 6.2: Results of the lexicon-based sentiment analysis with SentiLex-PT dictionary.

(a) Positive. (b) Neutral. (c) Negative.

Figure 6.3: Word cloud for each type of sentiment.

6.3.2 EMOTAIX.PT

Another dictionary that can be used to do the Lexicon-Based Sentiment analysis is the *EMOTAIX.PT* [53]. This dictionary was initially created by Piolat and Bannour [54], but it was later adapted to European Portuguese. *EMOTAIX* is a dictionary created to organize words into different categories according to their emotional load. It is composed of 3992 words where each word can be part of a Positive, Negative, or Neutral category and is classified into three hierarchical levels: global level (with six topics), middle level (with 18 topics), and specific level (with 50 topics). For example, the global level of one positive word can be Benevolence, Well-being, or Composure, and for a negative word, it could be either Malevolence, Ill-being, or Anxiety. From a total of 12743 tweets, using the

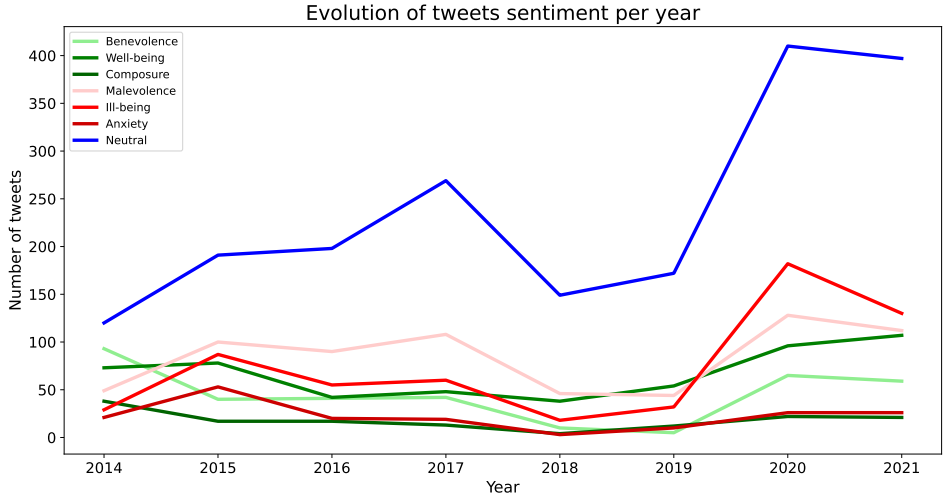


Figure 6.4: Evolution of sentiments present in tweets using EMOTAIX.PT lexicon.

Equations presented in the previous section (6.1 and 6.2), we end up with 917 positive tweets, 1312 negative tweets, and 10514 neutral tweets. Thus, it is easily concluded that this dictionary cannot classify so many words as *SentiLex-PT* despite the detailed classification for each word that it provides. From all the levels that this lexicon has available, it was just made use of the global one in this work. Figure 6.4 shows the

evolution of each category present at the global level over the years (green lines stand for positive categories and red lines for negative categories). It is possible to see that this lexicon found much more neutral words when compared to the other topics and does not have a clear trend. We can observe that in 2020 exists a small peak in all categories, similar to what happened in the search with the *SentiLex-PT* dictionary.

6.3.3 Validation of results

One of the problems of lexicon-based sentiment analysis is the negation of words like "is not perfect". For this sentence, Lexicon-based approaches probably will classify this sentiment as positive because of the word "perfect"despite not being so. In order to understand if our Lexicon-based classification is classifying well, we decided to manually label some tweets to compute the accuracy of our approaches.

Although this process seems easy, some precautions must be taken because of human's natural lack of attention and errors, which can lead to classification mistakes. So, asking multiple people to annotate the same content is usually a good idea to ensure the best possible result. Table 6.2 resumes some of the work done to create manual data annotations for texts. Shamma *et al.* [55] and Saif *et al.* [56] classifies three times each tweet and tweets with unanimous classifications are discarded. On the other hand, in Rosenthal *et al.* [57] approach, five different persons classify each tweet and, in cases where the results are ambiguous, to avoid discard results, the average is calculated and mapped to the nearest integer value. However, this is a problematic way to describe an ambiguous tweet. For instance, if a controversial tweet gets two negative labels, two positive labels, and one neutral label, it would be labeled neutral. However, this tweet would certainly not be the same as a tweet with a unanimous agreement on a neutral label [58]. Regarding the type of labels that the methods used, we can see that while Shamma *et al.* and Saif *et al.* use a simple label schema, Rosenthal *et al.* uses a more detailed schema.

Table 6.2: Comparison of techniques used to create manual text annotations. (P=Positive, N=Negative, W=Weak, S=Strong, n=Neutral)

Name	Discarted (%)	Coverage	Labels
Shamma *et al.* [55]	33.6%	3x	P, N, Mixed and Other
Saif *et al.* [56]	26.5%	3x	P, N, 0, Mixed and Other
Rosenthal *et al.* [57]	0%	5x	SP, SN, WP, WN and 0
Our approach	5.2%	5x	P, N, n with detailed descriptions

For this work, we follow mainly the guidelines presented in Rosenthal *et al.* [57]. We start by randomly selecting 250 tweets related to *novobanco* from the mass media search and give them to 30 different participants with ages comprised between 21 and 61. The tweets were distributed to each participant so that each participant classified 50

tweets, and five different persons classified each tweet. There was no contact between participants, and some hidden tests for quality control were created. For our hidden test, we selected some tweets with evident sentiment to compare with the participant's answers. If the answers did not match, it probably was a questionnaire with random answers so it will be discarded. The hidden tests resulted in 5 discarded questionnaires. The big difference between our approach and Rosenthal *et al.* is the labeling schema used for classification. In our approach, we create three labels, but to each label is created a description, presented in Table 6.3, because simple schemes create doubt in respondents. The descriptions for each label were based on a questionnaire presented in [59].

Moreover, knowing that the way Rosenthal *et al.* deals with unanimous classifications is not the best, in our approach, we decided to discard tweets without a minimum of 50% of votes in one category. For example, tweets with two positive votes, two negative votes, and one neutral vote were discarded. Furthermore, to obtain the participant's answers, we created a `React JS` application and published it on Git to be easily accessed by all the respondents (`https://gfmateus99.github.io/novobanco-survey/`). A more detailed view of this application, its characteristics, and its internal structure is provided in Annex I.

Table 6.3: Answers provided to the respondents to classify tweets.

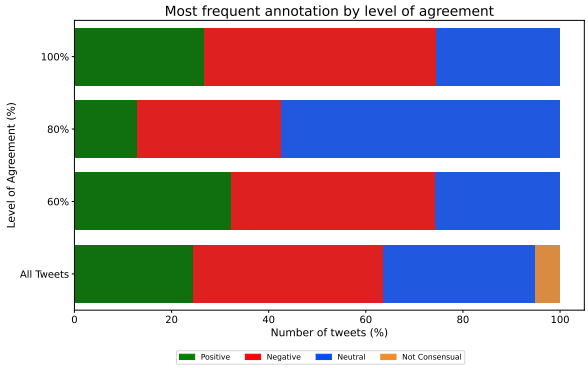
Label	Description
POSITIVE	The tweet contains positive language directly related to *novobanco*, such as expressions of success, compliance with payment deadlines, or a positive attitude.
NEGATIVE	The tweet contains negative language directly related to *novobanco*, such as expressions of criticism, judgment, questioning competence, suspicions of corruption, or expressions of sarcasm.
NEUTRAL	The tweet contains a mix of positive and negative language related to *novobanco* where none stand out. The tweet is not objective, is a subjective opinion, or is not directly related to *novobanco*.

Having all the participant's answers together and to better understand the consistency and agreement of the respondent's answers, we did inter-code reliability using the Fleiss kappa Coefficient [60], which allows us to understand the reliability of agreement (k) between the respondents. Applying this to all tweets manually classified, a value of $k = 0.501$ was obtained, meaning there is moderate strength of agreement between our respondents. This value cannot be considered a bad result considering the subjectivity inherent to some tweets. To better understand the respondent's answers, we can look to Table 6.5(a) where the levels of agreement in tweets are stated. The levels of agreement vary according to the number of equal answers. Thus, a level of agreement of 100% means that all the five answers were the same, a level of agreement of 80% means that there are four equal answers in 5 possible for that tweet, and the same logic applies to the other

levels. As explained before, tweets that do not have a majority of votes (at least a level of agreement of 50%) will be discarded. This resulted in the loss of 13 tweets from the 250 classified manually (5.2%) while the rest 237 (94.8%) were used. Next, in Figure 6.5(b) we can observe that the distribution of each sentiment is similar for all levels of agreement, meaning that in none of the levels exist a high unconformity and variance when compared to other levels of agreement.

Level of Agreement	Count	% of Total
100%	90	36%
80%	54	21.6%
60%	93	37.2%
< 60%	13	5.2%
Total:	250	100%

(a) Annotator agreement rates.



(b) Most frequent annotation by level of agreement.

Figure 6.5: Results of the manual classification agreement.

With this test set obtained, we can advance to the next phase, where it will be analyzed if our sentiment analysis is behaving well. As this can be seen as multi-class classification, to compare and analyze methods, we used mainly the `F1-score` that combines recall and precision into a single metric. To start, we analyze the score of *SentiLex-PT* with Equations 6.1 (standard) and 6.2 (measures the degree of sentiment). For Equation 6.2, we test several approaches to find the best range of degrees where tweets are negative, positive, and neutral. After some tests, we found that this method could improve the general score by classifying tweets with a sentiment between the interval $]-0.1, 0.1[$ as neutral. These are tweets with low sentiment, so they probably are more neutral than positive or negative. Regarding the *EMOTAIX.PT* dictionary using the degree of sentiments to evaluate sentiments did not improve the results in any test. Consequently, for *EMOTAIX.PT*, we used the traditional method that follows Equation 6.1.

Using the best methods found for each of the dictionaries, it was observed that in general *SentiLex-PT* could provide a better `F1-score` for all sentiments (Table 6.4). Knowing that it would be tough to improve the results of each dictionary alone, we tried to create a method that could combine both *SentiLex-PT* and *EMOTAIX.PT* to improve the general score. After some tests, we found that if we classify as negative, tweets with a degree of sentiment provided by *SentiLex-PT* in the interval $]-0.3, 0[$, and for the others, use the *EMOTAIX.PT* classification, we could improve the general score. This new Ensemble method worsened the neutral `F1-score`, but our main objective was to improve the positive and negative scores as those can probably influence the models more. This new ensemble method classifies tweets following Equation 6.3, where t represents a tweet and d is the degree of sentiment in tweet t.

$$ensemble(t,d) = \begin{cases} emotaix & \text{for} \quad d \notin]-0.3, 0[\\ -1 & \text{for} \quad d \in]-0.3, 0[\end{cases} \tag{6.3}$$

Table 6.4 resumes the results of this analysis, having in the count all sentiments individually and globally. The advantage of our ensemble method is that it can provide good accuracy for both positive and negative tweets, which are the ones that can influence most of our models. Regarding the neutral tweets, we could not improve the overall classification too much, probably because our dictionaries are limited and do not have all words. In short, when classifying tweets as positive and negative, our ensemble method has good accuracy, while the classification of neutral tweets is not so trustworthy.

	SentiLex-PT (Degree)			***EMOTAIX.PT***			***Ensemble***		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Positive	0.36	0.15	**0.21**	0.57	0.13	**0.21**	0.86	0.18	**0.30**
Negative	0.67	0.34	**0.45**	0.81	0.17	**0.29**	0.56	0.50	**0.53**
Neutral	0.46	0.9	**0.60**	0.34	0.85	**0.48**	0.38	0.64	**0.48**
Total	0.50	0.46	**0.48**	0.57	0.38	**0.46**	0.60	0.44	**0.51**

Table 6.4: Comparison of approaches used for Sentiment analysis.

6.4 Topic modelling

Besides the sentiment analysis providing a good indication of the content of the tweets, we can see that results show some subjectivity and are not perfect. As a result, we decided to do a topic modeling analysis where we will try to group tweets according to their topics instead of having the sentiment of tweets. The typical way to perform this is using Latent Dirichlet Allocation (LDA) model.

The LDA model is a probabilistic model where the different texts are represented as random mixtures over latent topics [61]. In other words, the LDA algorithm will look at the different texts to generate several groups of similar words, where each group generated corresponds to a topic. Despite LDA performing well for regular and big texts, when it comes to short texts, like tweets, the results decrease because of the minimal word co-occurrence information [62]. Several different strategies have been proposed to deal with the problem of short texts [63–65] but the one that gained more relevance due to its effectiveness was the Dirichlet Multinomial Mixture (DMM) [66]. DMM alleviates the LDA assumption that each text is a random mixture of latent topics. Knowing that the short text content is limited, DMM assumes that every text corresponds to a single topic. For this work, we will use an DMM derivation that integrates Gibbs sampling, presented in [67]. Authors found out that DMM combined with Gibbs sampling could find the number of topics automatically, with an appropriate balance of homogeneity and completeness, and still fast to compute.

6.4.1 DMM with Gibbs Sampling

To better explain the way this method works, we decided to create a simple analogy. For this analogy, we can think of a game with many players, and to each player is assigned a list of words. The game's objective is for players to form groups based on each list of words. Players with a similar list of words should be in the same group, and players with a different list of words should not be in the same group. Initially, players will be randomly assigned to random groups, and at each round of the game, each player will need to re-choose a group based on the two following rules:

- Choose a group with more people.
- Choose a group whose participants share similar words with him.

As the number of rounds goes on and knowing that we have many more groups available than is necessary, some groups will get bigger, others will get smaller, and some groups will be empty. In the end, we should expect that only a part of the groups have players, and the players in the same group share a similar list of words. If this occurs, it means that the game's objective has been reached.

DMM with Gibbs Sampling works exactly like this game but with the change that it does not use human minds to find new groups. More formally, in DMM with Gibbs Sampling, we will have an input D that is the number of tweets that will be analyzed, a list V that contains all of the different words that appear in the tweets, and a number K that is the number of groups/clusters available. In the beginning, we will assign tweets randomly to the K available clusters, recording information about the clusters where each tweet is ($\overrightarrow{z}$), the number of words in cluster z (m_z), and the number of occurrences of word w in clusters z (n_z^w). After this, in each interaction of this method, the tweets will be re-assign to each group based on the conditional distribution $p(z_d = z|\overrightarrow{z}_{\neg d}, \overrightarrow{d})$, where $\neg d$ means the cluster label of document d is removed from $\overrightarrow{z}$ to avoid to create biased results. Each time a a document get's a new cluster all the variables $\overrightarrow{z}$, m_z, n_z and n_z^w are updated accordingly.

It is expected that if we define K with a big number, as the number of interactions goes on, the tweets will start to be re-organized and find the correct number of clusters K for the specific data. Allowing that a word to appear multi times in a tweet, the conditional probability $p(z_d = z|\overrightarrow{z}_{\neg d}, \overrightarrow{d})$ is derive in Equation 6.4.

$$p(z_d = z|\overrightarrow{z}_{\neg d}, \overrightarrow{d}) = \frac{m_{z,\neg d} + \alpha}{D - 1 + K\alpha} \frac{\prod_{w \epsilon d} \prod_{j=1}^{N_d^w} (n_{z,\neg d}^w + \beta + j - 1)}{\prod_{i=1}^{N_d} (n_{z,\neg d} + V\beta + i - 1)} \tag{6.4}$$

The first part of the Equation 6.4 relates to the first rule of the game presented above ("Choose a group with more people"). This part tries to join tweets in big groups instead of being isolated. The $m_{z,\neg d}$ is the number of tweets in each group excluding tweet d, and α is a parameter that needs to be defined manually. A value of α near 0 makes the

probability of tweets choosing an empty group low, while a big value for α makes the probability of tweets choosing an empty table high.

The second part of the Equation 6.4 relates to the second rule of the game presented above ("Choose a group whose participants share similar words"). This part tries that tweets choose groups that share similar words. The $n^{w}_{z,\neg d}$ is the number of occurrences of word w in group z and $n_{z,\neg d}$ is the number number of words w in group z, without considering the tweet d in both cases. Larger $n^{w}_{z,\neg d}$ indicates that the tweet d shares similar interests with group z, so the probability of tweet d choosing group z will be larger. This part also has a parameter β that must be defined manually. If a value of 0 is given to β, tweets will never join tables that do not have a word of tweet d. A low value of β is not reasonable because tweet d can share multiple words with group z, but because others do not appear in the group, tweet d will not join that group. Moreover, as it is allowed that a word appears multiple times in each tweet, we need to add $\prod_{j=1}^{N^{w}_{d}}$ to the expression where N^{w}_{d} is the number of occurrences of word w in tweet d.

6.4.2 Coherence Metric

In topic modeling, one of the significant issues is how to evaluate the multiple models and choose the best. The best and trustful way is to look at the topics and manually evaluate the coherence. However, this can be tedious as the range of parameter values is high, and sometimes we end with many topics found. Consequently, some metrics were created that allow understanding of how much a topic is coherent. Some metrics use statistics and probabilities, while others look for the semantic similarity of words in each topic. However, this second way needs large and good texts to train, which becomes more difficult for the European Portuguese language. Considering this, we decided to use a metric and human classification mix to evaluate the different topics. In our approach, we started by using a coherence metric to find a good range of values to test and understand the influence of the different parameters on the results. After that, we evaluate the topics manually with a smaller range of values.

The coherence metric chosen for this work follows the Equation 6.5 [68]. For each topic, this metric counts the number of times two words (v_i, v_j) of the same topic co-occur and divides this by the number of documents in which the word v_j appears. For obtaining the total coherence for a topic T, the logarithm of this division is summed for every combination of words (v_i, v_j) of a topic. This metric will usually give negative values, as normally, the number of documents in which a word v_j appears will be higher than the number of documents in two words of the same topic (v_i, v_j) co-occur. Thus, as the denominator of the Equation 6.5 usually is higher, the result of the division will be a number under one, which will provide a negative value when the logarithm is applied. Finally, an value ϵ is added to prevent having a logarithm of zero.

$$coherence(T) = \sum_{(v_i, v_j) \in T} \log \frac{D(v_i, v_j) + \epsilon}{D(v_j)} \tag{6.5}$$

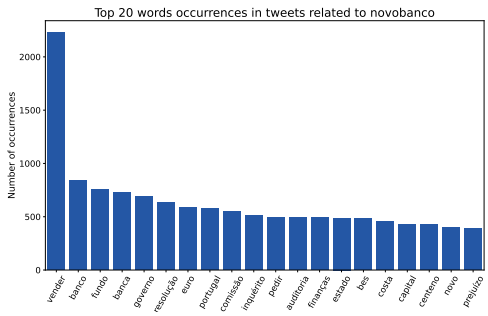
6.4.3 Experiments

Knowing that all the tweets that are being analyzed were obtained through the *"Novo Banco"* and *"novobanco"* keywords, we decided to take out these words to avoid these words creating garbage and damaging our results. This step is similar to adding the words *"Novo Banco"* and *"novobanco"* to our stopwords list to be removed. After this additional step, eight tweets ended with no words, so they were excluded from our initial dataset of 12743 records. Furthermore, to better understand this dataset, we run some statistics presented in Table 6.5. Note that during this section, all statistics and analysis were done after all preprocessing steps presented in 6.2 plus the additional step of taking out the words *"Novo Banco"* and *"novobanco"*. An interesting thing to note through the statistics of Table 6.5 is that the preprocessing steps reduced 3543 words from the vocabulary.

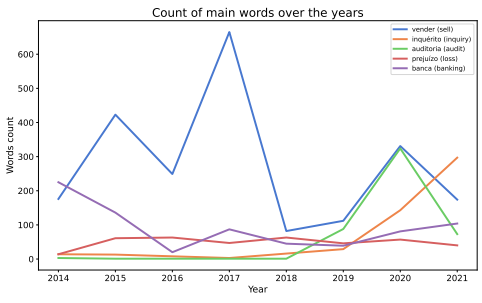
Table 6.5: Statistics about tweets related to novobanco.

Number of tweets	Size of vocabulary (before preprocessing)	Average (Standard deviation) of tweets length	Minimum (Maximum) tweet length
12735	4587 (8130)	5.8 (2.3)	1 (24)

Therefore, in Figure 6.6(a) we list the top 20 word occurrences in tweets related to novobanco, while in Figure 6.6(b) it is presented the occurrences of some words over the years. It is interesting to observe that the word sell (*"vender"*) has significantly more occurrences than the other ones over all the years, and words like inquiry (*"inquérito"*) and audit (*"auditoria"*) have an increase in last years.



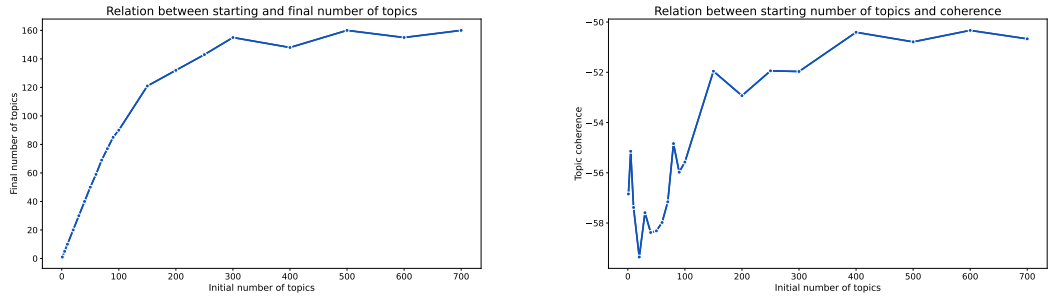
(a) Top 20 words occurrences.



(b) Occurrences of words over the years.

Figure 6.6: Occurrences of main words in tweets related to novobanco.

To start finding the best parameters for DMM with the Gibbs Sampling model, we investigate how the number of initial topics (K) influences the number of final topics. For this, we fixed the number of iterations to 30 and both α and β parameters to 0,1. The results of this search are presented in Figure 6.7. The first graph (6.7(a)) shows the relation between starting and the final number of topics while the second graph (6.7(b)) presents the coherence obtained through the Equation 6.5 for different K values. Focusing on the first graph, we can observe that from the starting number of topics K of 300, the final number of topics found converges for a value near 150, indicating that a value for $K = 300$ is sufficient for the algorithm to find the best topics. Although looking at the second graph, we can see the coherence start converging when $K = 400$. Analyzing both graphs together, we can see that from values of $K = 400$ the number of topics converges, and the coherence does not change a lot which can be a good sign that a value of $K = 400$ is good enough for our model.



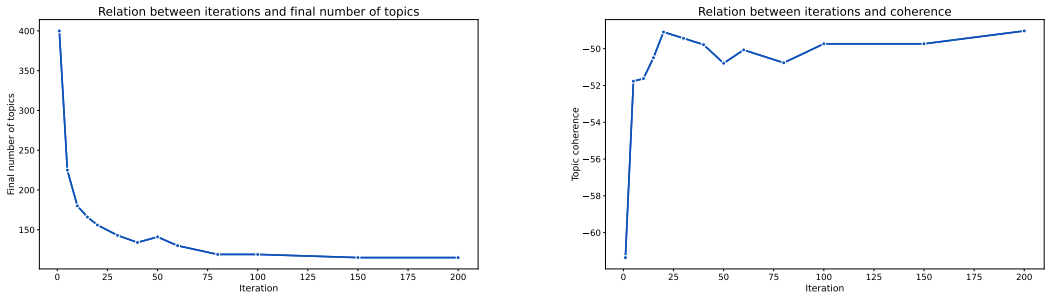
(a) Influence of starting number of topics.

(b) Coherence for different starting number of topics.

Figure 6.7: Relation between the starting number of topic with the final number of topics and coherence.

With a good value for K found, the next test will try to find how many iterations we should run to find the best topics. For this one, we fixed the number of initial topics K to400, both α and β parameters to 0,1, and ran the model with different values for iterations. Figure 6.8 shows the results of this experiment. Looking at the graphs presented in this Figure, we observe that as the number of iterations increases, the number of final topics decreases, and the coherence increases. From iteration 20, the number of topics found started converging, getting a more stable value since iteration 80. The topic's coherence reaches the lowest value at iteration 20, and from this iteration on, coherence has slight variations but without deviating too much from the other values. Observing both graphs together, a value higher than 80 for the number of iterations seems good enough to find the lowest number of topics with good coherence on topics. We decide for the following experiments to choose 100 to the number of iterations.

After finding a good value for the starting number of initial topics K (400) and for the number of iterations (100) defined, the parameters that still need to be studied are α and β. Recalling what was explained before, α will mainly influence the number of

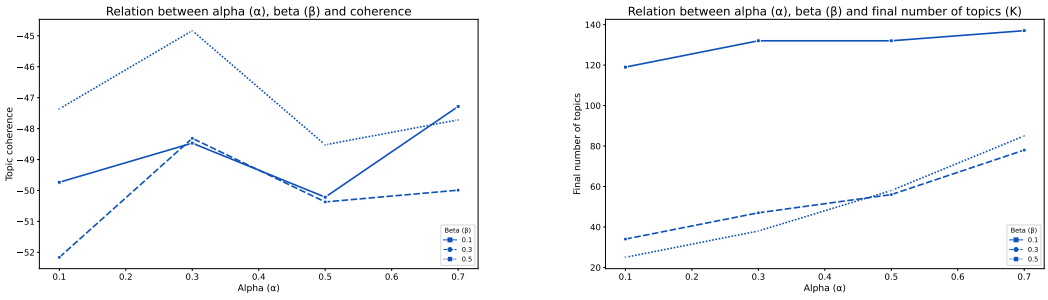


(a) Influence of the number of iterations.

(b) Coherence for the different number of iterations.

Figure 6.8: Relation between the number of iterations with the final number of topics and coherence.

groups, while β will mainly influence how similar tweets from the same topic are. For the next test we defined a range of values for α of $\{0.1, 0.3, 0.5, 0.7\}$ and for β of $\{0.1, 0.3, 0.5\}$. Figure 6.9 provide the results of this experiment. Looking at the first graph of this Figure (6.9(a)), where it is analyzed the relation of α and β with coherence, the first conclusion that we can take is that, in general, the model gets the best coherence for all values of α when $\beta = 0.5$. However, this graph also shows that when $\alpha = 0.3$, all the different models have a peak for all β values. Considering the second graph (6.9(b)), where the relation of α and β with the final number of topics is analyzed, we can observe that as the value of α increases, the value of final topics increases for all different β. This was expected because as the value of α increases, the probability of the number of groups growing will increase. Analyzing both graphs together, it appears that a value of $\beta = 0.5$ combined with $\alpha = 0.3$ provides the best model regarding consistency and the final number of topics.



(a) Influence of alpha and beta.

(b) Coherence for the different alpha and beta values.

Figure 6.9: Relation between alpha and beta values with the final number of topics and coherence.

The experiments explained above allow to obtain what seems to be reasonable values for the different parameters that need to be defined in DMM with Gibbs Sampling. We did some extra tests varying these parameters to evaluate manually if the topics found

would be better. However, the differences did not appear to be significant, so we decided to stick to those values. Our final model was trained with $K = 400$, $\beta = 0.5$, $\alpha = 0.3$, 100 iterations, and resulted in 45 different topics. However, having a model with the best parameters is not enough for topic modeling as the interpretability of topics found is also an important step. To start, we analyze the frequency of different topics to understand if some topics could be discarded (Figure 6.10). This Figure shows that most topics are not frequent while some stand out from others. To get an idea, from the total of 45 topics found in our final model, 27 had one document, and 5 had two documents. This indicates that these topics are not very relevant but before we remove we take a look inside each of them to understand if some could be a potentially good topic. However, in general, none of them appeared to have relevant or significantly different information compared to the topics with more documents. In the end, we removed all of them to have a cohesive set of topics, leading to 13 final topics.

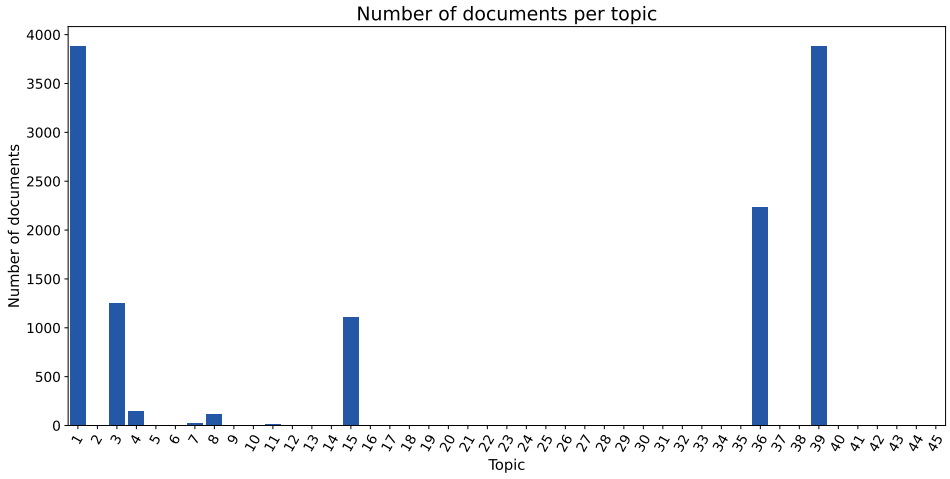


Figure 6.10: Number of documents per topic found in our final model.

With a number of topics more manageable to analyze, we can do a more concise manual investigation of the content of each topic individually. For each set of words, we tried to find a name that could describe the topic well and if each topic has good or poor coherence, and the results are shown in Table 6.6. For a simple understanding in each line of the Table, the first number corresponds to the topic number and matches the topic number in Figure 6.10. Analyzing the results, we could understand that most words inside each topic obtained were related, and it was possible to find a subject for each topic. Despite this, there were some topics where it was more challenging to find a simple word with the capacity to describe the topic. A case of this is topics 16 and 20, where we could not find a subject with the ability to describe the content. However, observing Table 6.6, it is interesting to see that the topics with more documents (1, 3, 15, 36, and 39) are not too difficult to explain.

Table 6.6: Final topics found with the best model after removing topics with low frequency (less than 3).

	Number of documents	**Topic**	**Words**
1	3888	Politics and government	vender, governo, inquerito, comissao, costa, auditoria, centeno, parlamento, psd, pedir
3	1256	Financial problems	vender, prejuizo, euro, credito, imovel, lucro, malparar, ativo, ano, gnb
4	146	Credits and growth perspective	rating, moody, dbrs, lixo, manter, alta, perspetiva, melhorar, positivo, cortar
7	22	Names	rtp, controlar, silva, eduardo, antonio, martim, jose, ana, pedro
8	122	Art	premio, museu, foto, ceder, revelacao, colecao, obra, arte, fotografia, vencer
11	18	Bank problems	costa, carlos, fruta, apodrecer, cabaz, vender, parcialmente, comparar, audicao, podre
15	1110	Bank problems	lesado, bloco de esquerda, bes, emigrante, cliente, comercial, papel, obrigacao, divida, solucao
16	8	–	custar, facilidade, lar, troco, ver, bagao felix, chamar, interessado, surgir
20	3	–	deixar, ongoing, sucesso, uniao, norte, maia, moniz, polemica, correr, hipocrisia
36	2234	Banks	vender, banco, lone star, compra, portugal, banca, financas, bpi, santander
37	4	Media	jornal, disponivel, revista, consulta, manha, permanente, correio, app, sapo, capa
39	3883	Money	fundo, resolucao, vender, banca, banco, trabalhador, euro, portugal, capital, estado
40	3	Bank problems	divida, dinheiro, opiniao, necessidade, dizer, senior, analista, agua, medo, escaldar

7

Influence of human context

This chapter describes .

8 Conclusion

This chapter describes .

8.1 Main Contributions

8.2 Limitations and issues

8.3 Future work

Bibliography

[1] *17 reasons cloud computing is growing like crazy*. Last visited on February 2022. URL: https://www.hso.co.uk/blog/cloud/17-reasons-cloud-computing-is-growing-like-crazy (cit. on p. 1).

[2] M. Armbrust et al. "A view of cloud computing". In: *Communications of the ACM* 53.4 (2010), pp. 50–58 (cit. on p. 1).

[3] G. Aggarwal. *Council post: How the pandemic has accelerated cloud adoption*. Last visited on February 2022. 2021-12. URL: https://www.forbes.com/sites/forbestechcouncil/2021/01/15/how-the-pandemic-has-accelerated-cloud-adoption/ (cit. on p. 1).

[4] V. K. Jayakumar et al. "A Self-Optimized Generic Workload Prediction Framework for Cloud Computing". In: *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2020, pp. 779–788 (cit. on pp. 1, 9, 13).

[5] W. Iqbal, J. L. Berral, D. Carrera, et al. "Adaptive sliding windows for improved estimation of data center resource utilization". In: *Future Generation Computer Systems* 104 (2020), pp. 212–224 (cit. on pp. 1, 11–13).

[6] *Overview of forecasting models¶*. URL: https://unit8co.github.io/darts/userguide/forecasting_overview.html (cit. on p. 3).

[7] *What is a data center?* 2022-05. URL: https://www.cisco.com/c/en/us/solutions/data-center-virtualization/what-is-a-data-center.html (cit. on p. 5).

[8] *Data Center Networking*. Last visited on January 2022. URL: https://www.manageengine.com/network-monitoring/data-center-networking.html (cit. on p. 5).

[9] H. Qi et al. "Data center network architecture in cloud computing: review, taxonomy, and open research issues". In: *Journal of Zhejiang University SCIENCE C* 15.9 (2014), pp. 776–793 (cit. on p. 5).

[10] T. Wang et al. "Rethinking the data center networking: Architecture, network protocols, and resource sharing". In: *IEEE access* 2 (2014), pp. 1481–1496 (cit. on p. 5).

[11] H. Dong et al. "Next-Generation Data Center Network Enabled by Machine Learning: Review, Challenges, and Opportunities". In: *IEEE Access* (2021) (cit. on p. 8).

[12] M. Xue et al. "Machine learning security: Threats, countermeasures, and evaluations". In: *IEEE Access* 8 (2020), pp. 74720–74742 (cit. on p. 8).

[13] A. A. Bankole and S. A. Ajila. "Cloud client prediction models for cloud resource provisioning in a multitier web application environment". In: *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*. IEEE. 2013, pp. 156–161 (cit. on p. 9).

[14] S. Islam et al. "Empirical prediction models for adaptive resource provisioning in the cloud". In: *Future Generation Computer Systems* 28.1 (2012), pp. 155–162 (cit. on p. 9).

[15] *Time series analysis: Definition, types, techniques, and when it's used*. Last visited on February 2022. URL: https://www.tableau.com/learn/articles/time-series-analysis (cit. on p. 9).

[16] W. Fang et al. "Rpps: A novel resource prediction and provisioning scheme in cloud data center". In: *2012 IEEE Ninth International Conference on Services Computing*. IEEE. 2012, pp. 609–616 (cit. on pp. 9, 13).

[17] R. N. Calheiros et al. "Workload prediction using ARIMA model and its impact on cloud applications' QoS". In: *IEEE transactions on cloud computing* 3.4 (2014), pp. 449–458 (cit. on pp. 9, 13).

[18] X. Wang et al. "Online cloud resource prediction via scalable window waveform sampling on classified workloads". In: *Future Generation Computer Systems* 117 (2021), pp. 338–358 (cit. on pp. 9, 11).

[19] G. Lai et al. "Modeling long-and short-term temporal patterns with deep neural networks". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018, pp. 95–104 (cit. on p. 9).

[20] T. Khan et al. "Workload forecasting and energy state estimation in cloud data centres: ML-centric approach". In: *Future Generation Computer Systems* 128 (2022), pp. 320–332 (cit. on pp. 9, 13, 16).

[21] K. Zhu et al. "Differentiated transmission based on traffic classification with deep learning in datacenter". In: *2020 IFIP Networking Conference (Networking)*. IEEE. 2020, pp. 599–603 (cit. on pp. 9, 13).

[22] F. Estrada-Solano, O. M. Caicedo, and N. L. Da Fonseca. "Nelly: Flow detection using incremental learning at the server side of sdn-based data centers". In: *IEEE Transactions on Industrial Informatics* 16.2 (2019), pp. 1362–1372 (cit. on pp. 9, 13).

[23] W. Iqbal et al. "Adaptive prediction models for data center resources utilization estimation". In: *IEEE Transactions on Network and Service Management* 16.4 (2019), pp. 1681–1693 (cit. on pp. 9–11, 13).

[24] I. K. Kim et al. "CloudInsight: Utilizing a Council of Experts to Predict Future Cloud Application Workloads". In: *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. 2018, pp. 41–48. DOI: 10.1109/CLOUD.2018.00013 (cit. on pp. 9, 10, 13).

[25] C. Liu et al. "An adaptive prediction approach based on workload pattern discrimination in the cloud". In: *Journal of Network and Computer Applications* 80 (2017), pp. 35–44 (cit. on pp. 9, 10, 13).

[26] R. A. Jacobs et al. "Adaptive mixtures of local experts". In: *Neural computation* 3.1 (1991), pp. 79–87 (cit. on p. 10).

[27] Y. H. Hu, S. Palreddy, and W. J. Tompkins. "A patient-adaptable ECG beat classifier using a mixture of experts approach". In: *IEEE transactions on biomedical engineering* 44.9 (1997), pp. 891–900 (cit. on p. 10).

[28] B. L. Dalmazo, J. P. Vilela, and M. Curado. "Online traffic prediction in the cloud: a dynamic window approach". In: *2014 International Conference on Future Internet of Things and Cloud*. IEEE. 2014, pp. 9–14 (cit. on pp. 11, 13).

[29] L. Chen et al. "Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization". In: *Proceedings of the 2018 conference of the ACM special interest group on data communication*. 2018, pp. 191–205 (cit. on p. 12).

[30] S. Salman et al. "DeepConf: Automating data center network topologies management with machine learning". In: *Proceedings of the 2018 Workshop on Network Meets AI & ML*. 2018, pp. 8–14 (cit. on p. 12).

[31] B. Arzani et al. "Taking the blame game out of data centers operations with netpoirot". In: *Proceedings of the 2016 ACM SIGCOMM Conference*. 2016, pp. 440–453 (cit. on p. 12).

[32] A. Tongaonkar, R. Keralapura, and A. Nucci. "Santaclass: A self adaptive network traffic classification system". In: *2013 IFIP Networking Conference*. IEEE. 2013, pp. 1–9 (cit. on p. 12).

[33] A. Kansal et al. "Virtual machine power metering and provisioning". In: *Proceedings of the 1st ACM symposium on Cloud computing*. 2010, pp. 39–50 (cit. on p. 13).

[34] R. Guidotti et al. "A survey of methods for explaining black box models". In: *ACM computing surveys (CSUR)* 51.5 (2018), pp. 1–42 (cit. on p. 14).

[35] C. Molnar. *Interpretable machine learning*. Lulu. com, 2020 (cit. on p. 15).

[36] P. Gupta. *Decision trees in machine learning*. Last visited on February 2022. 2017-11. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052 (cit. on pp. 16, 17).

[37] J. H. Friedman and B. E. Popescu. "Predictive learning via rule ensembles". In: *The annals of applied statistics* 2.3 (2008), pp. 916–954 (cit. on p. 17).

[38] *Reliance on social media in today's society*. Last visited on February 2022. 2019-10. URL: https://www.insegment.com/blog/reliance-on-social-media-in-todays-society/ (cit. on p. 34).

[39] A. Hutchinson. *Here's why Twitter is so important, to everyone*. Last visited on February 2022. 2016-03. URL: https://www.socialmediatoday.com/social-networks/heres-why-twitter-so-important-everyone (cit. on p. 34).

[40] L. Borges. *PS Contra Aumento de Remunerações E Prémios no Novo banco*. Last visited on February 2022. 2020-05. URL: https://www.publico.pt/2020/05/13/politica/noticia/ps-aumento-remuneracoes-premios-novo-banco-1916384 (cit. on p. 40).

[41] V. Kharde, P. Sonawane, et al. "Sentiment analysis of twitter data: a survey of techniques". In: *arXiv preprint arXiv:1601.06971* (2016) (cit. on pp. 45, 47).

[42] J. Hirschberg and C. D. Manning. "Advances in natural language processing". In: *Science* 349.6245 (2015), pp. 261–266 (cit. on p. 45).

[43] H. Bonthu. *Rule-based sentiment analysis in Python for Data Scientists*. 2021-06. URL: https://www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/ (cit. on p. 46).

[44] S. Chakravarthy. *Tokenization for natural language processing*. 2020-07. URL: https://towardsdatascience.com/tokenization-for-natural-language-processing-a179a891bad4 (cit. on p. 46).

[45] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.", 2009 (cit. on p. 46).

[46] A. Branco. *LX-Stopwords*. URL: https://hdl.handle.net/21.11129/0000-000B-D31A-A (cit. on p. 46).

[47] P. Carvalho and M. J. Silva. "SentiLex-PT: Principais caracteristicas e potencialidades". In: *Oslo Studies in Language* 7.1 (2015) (cit. on pp. 46, 47).

[48] P. Qi et al. "Stanza: A python natural language processing toolkit for many human languages". In: *arXiv preprint arXiv:2003.07082* (2020) (cit. on p. 46).

[49] C. D. Manning, P. Raghavan, and H. Schütze. "The term vocabulary and postings lists". In: *Introduction to information retrieval*. Cambridge University Press, 2018, pp. 19–45 (cit. on p. 46).

[50] V. Balakrishnan and E. Lloyd-Yemoh. "Stemming and lemmatization: a comparison of retrieval performances". In: (2014) (cit. on p. 46).

[51] A. Beri. *Stemming vs lemmatization*. 2021-01. URL: https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221 (cit. on p. 46).

[52] D. A. Pereira. "A survey of sentiment analysis in the Portuguese language". In: *Artificial Intelligence Review* 54.2 (2021), pp. 1087–1115 (cit. on p. 47).

[53] S. F. O. Costa. "Adaptação e teste de uma base lexical de palavras emocionais para o português europeu:(EMOTAIX. PT)". In: (2012) (cit. on p. 49).

[54] A. Piolat and R. Bannour. "EMOTAIX: un scénario de Tropes pour l'identification automatisée du lexique émotionnel et affectif". In: *LAnnee psychologique* 109.4 (2009), pp. 655–698 (cit. on p. 49).

[55] D. A. Shamma, L. Kennedy, and E. F. Churchill. "Tweet the debates: understanding community annotation of uncollected sources". In: *Proceedings of the first SIGMM workshop on Social media*. 2009, pp. 3–10 (cit. on p. 50).

[56] H. Saif et al. "Evaluation datasets for Twitter sentiment analysis: a survey and a new dataset, the STS-Gold". In: (2013) (cit. on p. 50).

[57] S. Rosenthal, N. Farra, and P. Nakov. "SemEval-2017 task 4: Sentiment analysis in Twitter". In: *arXiv preprint arXiv:1912.00741* (2019) (cit. on p. 50).

[58] K. Kenyon-Dean et al. "Sentiment analysis: It's complicated!" In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018, pp. 1886–1895 (cit. on p. 50).

[59] S. Mohammad. "A practical guide to sentiment annotation: Challenges and solutions". In: *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*. 2016, pp. 174–179 (cit. on p. 51).

[60] J. L. Fleiss. "Measuring nominal scale agreement among many raters." In: *Psychological bulletin* 76.5 (1971), p. 378 (cit. on p. 51).

[61] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022 (cit. on p. 53).

[62] C. Li et al. "Enhancing topic modeling for short texts with auxiliary word embeddings". In: *ACM Transactions on Information Systems (TOIS)* 36.2 (2017), pp. 1–30 (cit. on p. 53).

[63] L. Hong and B. D. Davison. "Empirical study of topic modeling in twitter". In: *Proceedings of the first workshop on social media analytics*. 2010, pp. 80–88 (cit. on p. 53).

[64] X. Quan et al. "Short and sparse text topic modeling via self-aggregation". In: *Twenty-fourth international joint conference on artificial intelligence*. 2015 (cit. on p. 53).

[65] X. Yan et al. "A biterm topic model for short texts". In: *Proceedings of the 22nd international conference on World Wide Web*. 2013, pp. 1445–1456 (cit. on p. 53).

[66] K. Nigam et al. "Text classification from labeled and unlabeled documents using EM". In: *Machine learning* 39.2 (2000), pp. 103–134 (cit. on p. 53).

[67] J. Yin and J. Wang. "A dirichlet multinomial mixture model-based approach for short text clustering". In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 233–242 (cit. on p. 53).

[68] D. Mimno et al. "Optimizing semantic coherence in topic models". In: *Proceedings of the 2011 conference on empirical methods in natural language processing*. 2011, pp. 262–272 (cit. on p. 55).

[69] *GET /2/tweets/search/all | docs | twitter developer platform*. Last visited on February 2022. URL: https://developer.twitter.com/en/docs/twitter-api/tweets/search/api-reference/get-tweets-search-all (cit. on p. 70).

A

Datasets specification

Table A.1: Media accounts that were used to retrieve data. (Information in the table was last accessed in February 2022)

Name	Identifier	Number of tweets	Followers
CM Jornal	@cmjornal	380.3K	438.9K
CNN Portugal	@cnnportugal	378.5K	310.1K
Diário de Notícias	@dntwit	308.7K	226.6K
Expresso	@expresso	485.2K	509.6K
Jornal de Negócios	@JNegocios	208.4K	176.8K
Jornal de Notícias	@JornalNoticias	262.9K	534.9K
Lusa Notícias	@Lusa_noticias	328.3K	244.6K
Observador	@observadorpt	315.1K	224.7K
Jornal Económico	@ojeconomico	161.6K	222K
Publico	@Publico	518K	813.5K
Sapo	@sapo	102.1K	280.7K
SIC Noticias	@SICNoticias	373.8K	826.6K
Sol	@SolOnline	215.5K	263.7K
TSF	@TSFRadio	263.7K	291.6K
RTP Notícias	@RTPNoticias	97.5K	402K
RTP 1	@RTP1	24.1K	210.1K

Table A.2: Features description of Twitter Dataset. Reasons descriptions for some features to have been discarded are presented in table A.4.

Feature	**Description** (Obtained from [69])	**Type**	**Removed**	**Reason**
conversation_id	The Tweet ID of the original Tweet of the conversation(which includes direct replies, replies of replies)	string	NO	-
text	The content of the Tweet.	string	NO	-
created_at	Creation time of the Tweet.	date (ISO 8601)	NO	-
lang	Language of the Tweet, if detected by Twitter. Returned as a BCP47 language tag.	string	NO	-
author_id	Unique identifier of this tweet user.	string	NO	-
attachments	Specifies the type of attachments (if any) present in this Tweet.	object	YES	1
source	The name of the app the user Tweeted from.	string	NO	-
reply_settings	Shows who can reply to this Tweet. Fields returned are *everyone*, *mentionedUsers*, and *following*.	string	YES	2
entities	Contains details about text that has a special meaning in a Tweet.	object	YES	3
context_annotations	Contains context annotations for the Tweet.	array	YES	1
public_metrics	Engagement metrics for the Tweet at the time of the request.	object	NO	-
possibly_sensitive	Indicates if this Tweet contains URLs marked as sensitive, for example content suitable for mature audiences.	boolean	YES	3
in_reply_to_user_id	If this Tweet is a Reply, indicates the user ID of the parent Tweet's author.	string	NO	-
referenced_tweets	A list of Tweets this Tweet refers to.	array	NO	-
withheld	Contains withholding details for withheld content.	object	YES	1
geo	Contains details about the location tagged by the user in this Tweet, if they specified one.	object	YES	3
id	The Tweet unique ID	string	NO	-

Table A.3: Description of metrics motorized in the novobanco data center. All the metrics are given in numerical values (with the respective scale) in a time series format. (*Reasons are presented in Table A.4)

Metric	Removed	Reason*	Description
Timestamp (Index)	NO	-	Time of occurrence of the record.
Check_MK			
Time usage by phase	YES	3	time usage of a monitoring tool that registers server metrics values.
Average	YES	3	
CPU utilization			
Per Core utilization	YES	3	CPU utilization measures how much work a CPU does to manage an operating system's tasks or process resources of a server.
Average	NO	-	
Disk IO summary			
Read wait time	YES	5	Monitors summary metrics related to active disk I/O time and the pace at which data is transferred from the hard drive to the RAM of a server.
Write wait time	NO	-	
Disk Read operations	YES	4	
Disk Write operations	YES	5	
Write Throughput	YES	4	
Read Throughput	YES	4	
File system			
Used filesystem space	YES	4	Monitors metrics related to the server file system. Each server utilizes a file system to manage available space in the server.
Filesystem Shrinking	NO	-	
Filesystem growth	YES	4	
Filesystem size	YES	2	
Free space	YES	4	
Interface			
Input Bandwidth	NO	-	Monitors different interfaces of each server as the bandwidth, packets, and errors that pass through the server network, either an input or output.
Output Bandwidth	NO	-	
Input non-unicast packets	NO	-	
Output non-unicast packets	YES	4	
Input unicast packets	YES	4	
Output unicast packets	YES	4	
Errors	YES	1	
Length of output queue	YES	2	
Memory and pagefile			
RAM used	NO	-	Monitors the memory used to maintain recent data. Specifically, it monitors the main memory of a computer (RAM), where all hardware, programs, and data that need to be quickly reached are maintained.
RAM installed	YES	2	
Errors	YES	2	
PING			
Round trip average	NO	-	Monitors how long it takes for packets to transit between a source and a target in a network.
Packet loss	YES	1	
Uptime	YES	4	Monitors the time a server runs without a restart or a shutdown, i.e., monitors the time a server is fully functional (the performance of the server).

Table A.4: Reasons for some features been discarded during data cleaning process

Reasons	**Description**
1	Discarded because of the high percentage of missing values
2	Discarded because all the values are the same in this feature.
3	Discarded because this feature has low importance for the problem that is being analyzed
4	Discarded because the feature is correlated with another one
5	Discarded because the values stagnate from a specific date and fail to provide relevant information

I

Questionnaire view

To collect the several manual annotations tweets, we need a questionnaire with the following characteristics:

- Good usability;
- Easy to share and access by all respondents;
- Present just 50 tweets to each respondent;
- Capable of presenting different tweets to different participants;
- Possibility to have hidden tests;
- Provide answers in a format easy to further analyze.

There are paid programs for the English language where we can set up all these rules, and the software finds respondents automatically and send the respective answers as APPEN[1]. Differently, there is free software such as Label Studio[2] that allows us to define everything manually and share the questionnaire with specific respondents of our choice. However, looking at our purpose, we thought that the interface provided by Label Studio was not very clear and could create some difficulties for respondents to use. Besides that, Label Studio does not allow to restrict the number of tweets for each respondent to analyze and cannot insert hidden tests.

Having this in mind, we decided to create a React app from scratch since we wanted simple software that followed the above characteristics. In this way, we could implement everything as we wanted and were necessary, providing a simple, easy, and specific app that fit our purpose. This app was published online via GitHub to be easily accessed by all participants (https://gfmateus99.github.io/novobanco-survey/), and the database for this app is 250 tweets selected randomly from the dataset with tweets related to *Novo Banco*. The internal structure of the developed application can be seen in I.1.

[1] https://appen.com/
[2] https://labelstud.io/

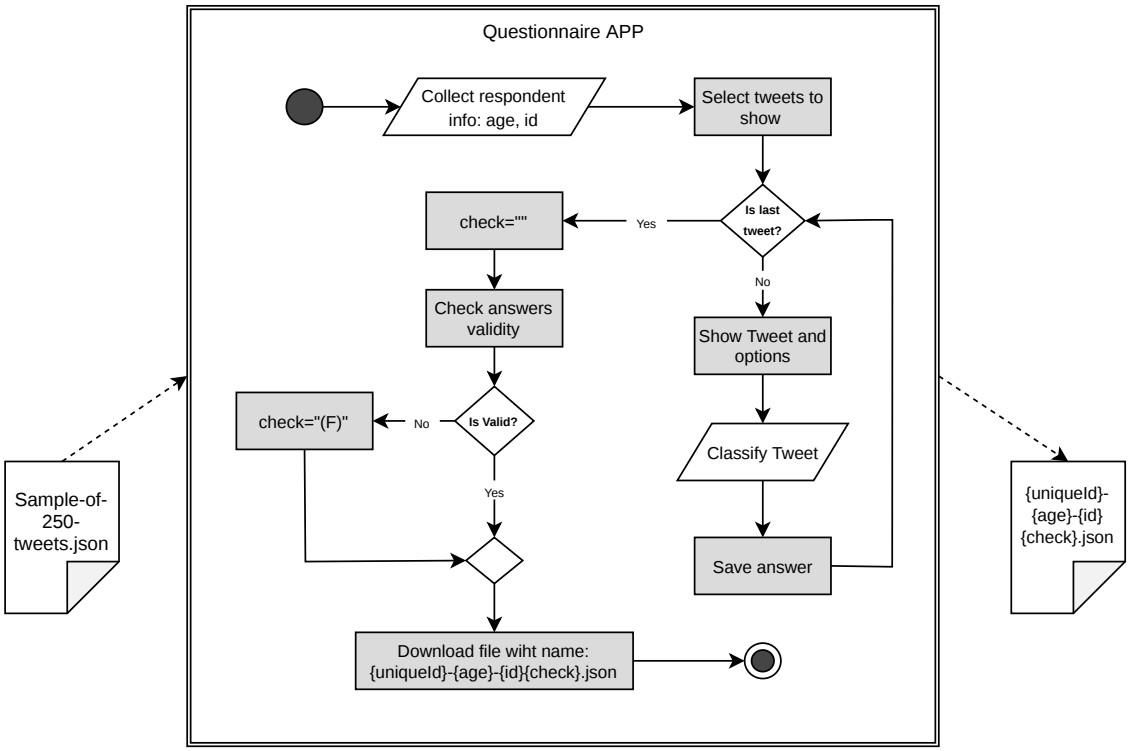


Figure I.1: Flow diagram of the internal structure of the developed application.

When the questionnaire is opened, the respondent has to insert some personal information, as shown in Figure I.2. The id camp is there to select the part of tweets that each respondent will classify (for example, id 1 will classify the first 50 tweets of the database, id 2 the tweets between positions 50 and 100 of the database, and so on). We previously assigned the id that each respondent should insert. After this, the user can start the questionnaire and will have 50 tweets to classify, as shown in I.3. When the respondent arrives at the end of the questionnaire, a file in `.json` format with the respondent's answers is automatically downloaded and in case of a download fail, the respondent will have a button available to repeat the download as Figure I.4 shows. The name of this file will be according to the information inserted by the respondent at the beginning. If the user inserted a *age="32"* and *id="4"* the file is downloaded with the name `"uniqueId-32-4.json"`. *uniqueId* is a random identifier to distinguish files of same *id*. Before the download, a hidden test is done to check if the respondent's answers were not random. For this test, we selected some tweets that the sentiment is obvious to compare with the respondent answers. If the answers did not match those tweets, it probably was a questionnaire answered with random answers, so the respective questionnaire needs to be discarded. To identify these cases, an additional camp with the `"(F)"` is added to the file's name. `"uniqueId-32-4.json"` would be a file that passed with success in our hidden tests, while `"uniqueId-32-4(F).json"` means that it did not pass in tests. To conclude the participation, respondents will have to send that file to a stated email address.

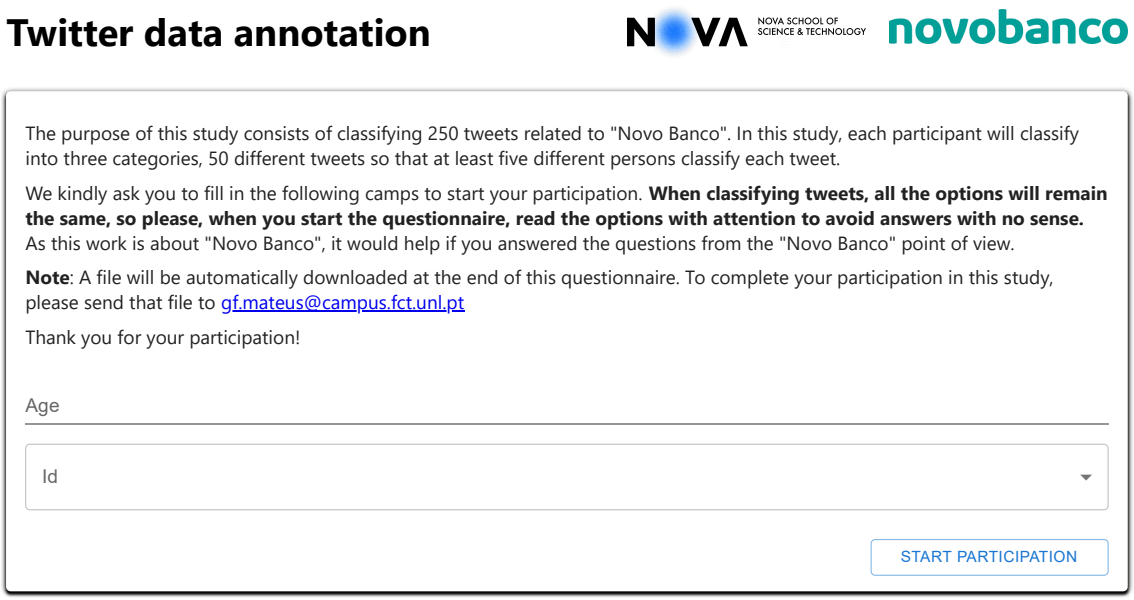


Figure I.2: First page of our questionnaire.

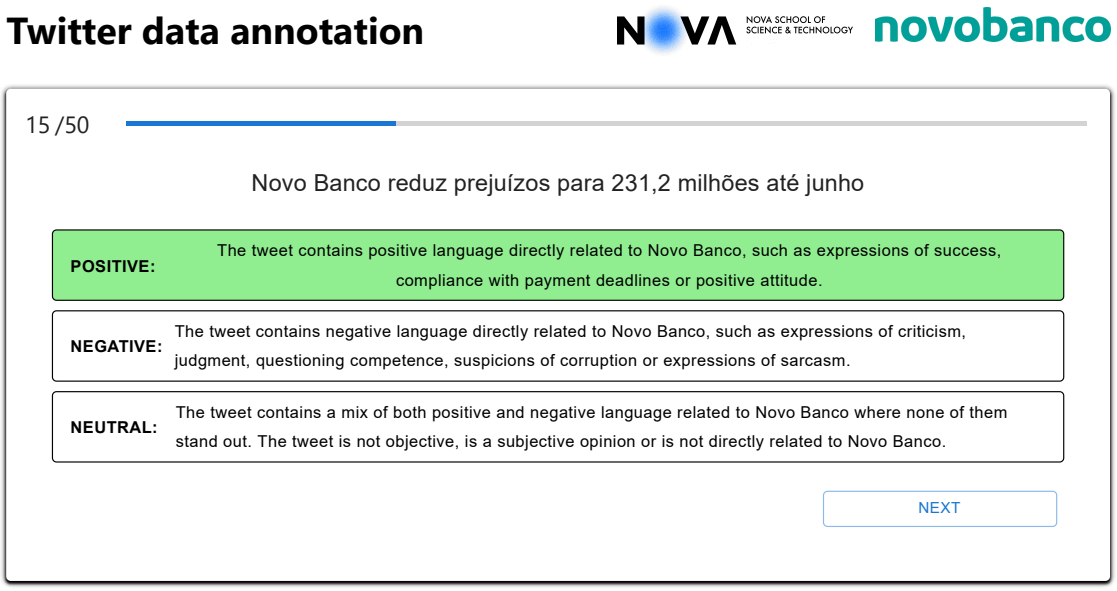


Figure I.3: Page with one question of our questionnaire.

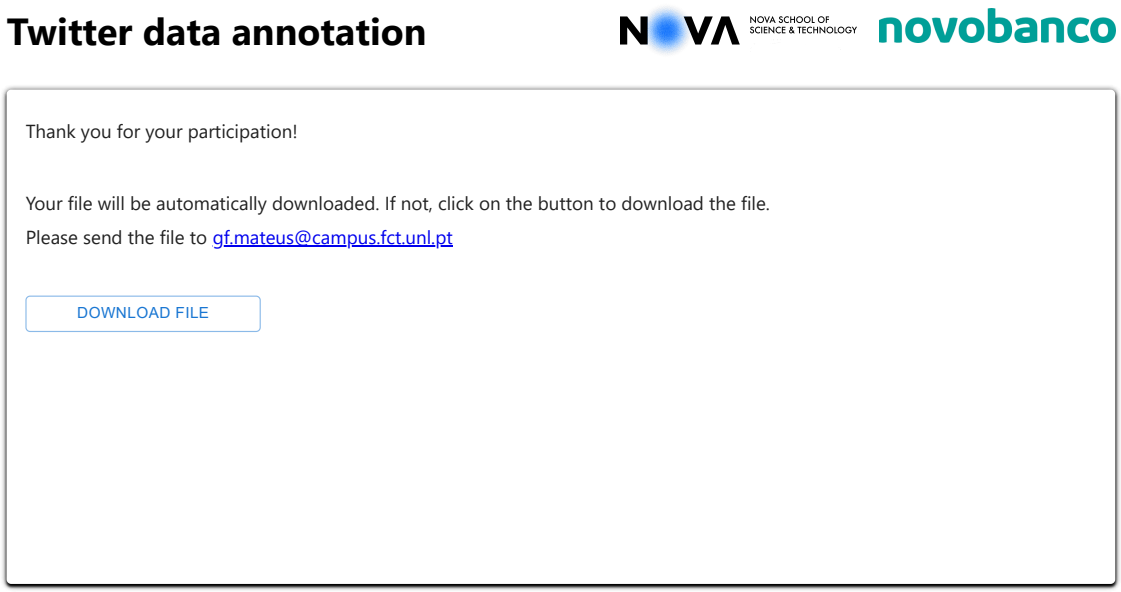


Figure I.4: Last page of our questionnaire.

II

Other Metrics Plots

II.1 RAM Utilization

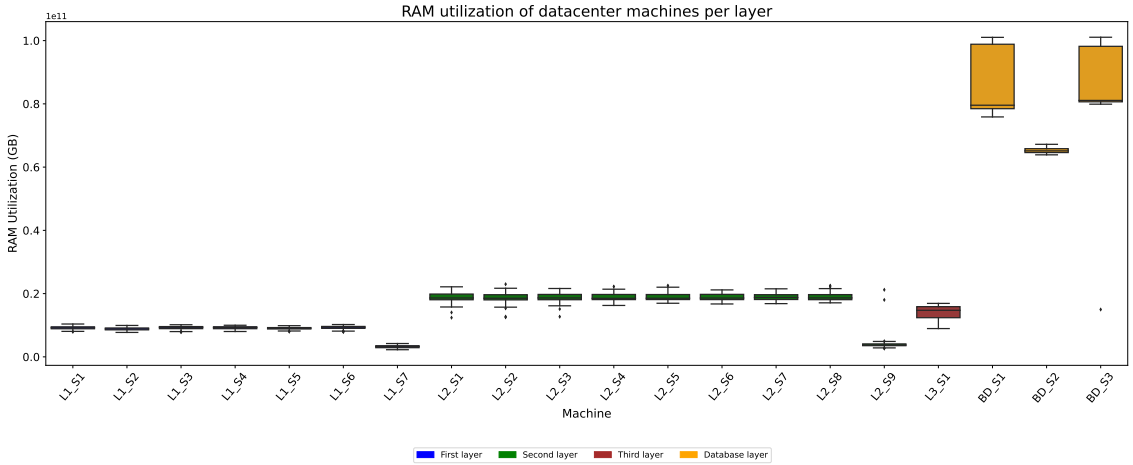
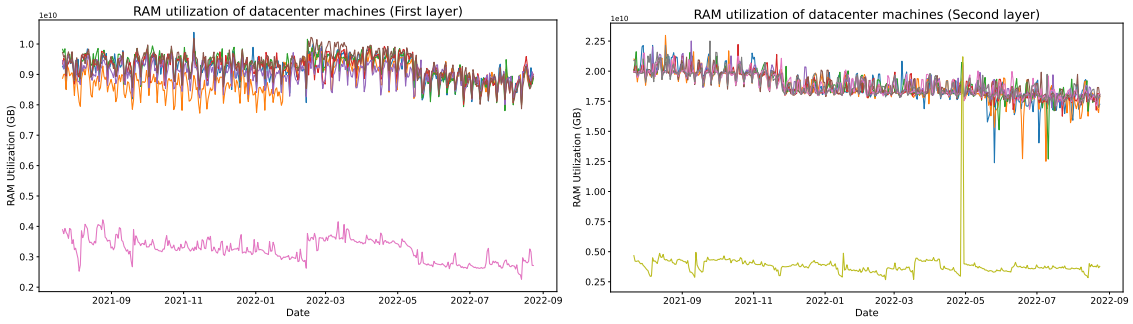


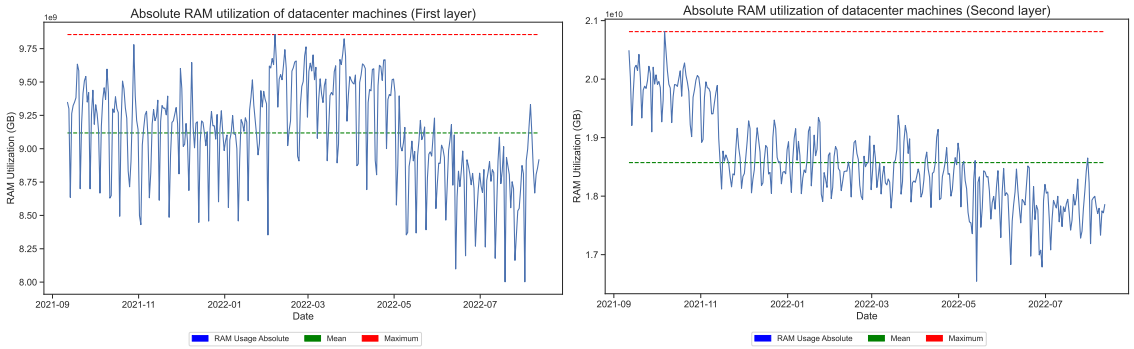
Figure II.1: RAM Average over servers and layers.



(a) RAM utilization of the servers of first layer.

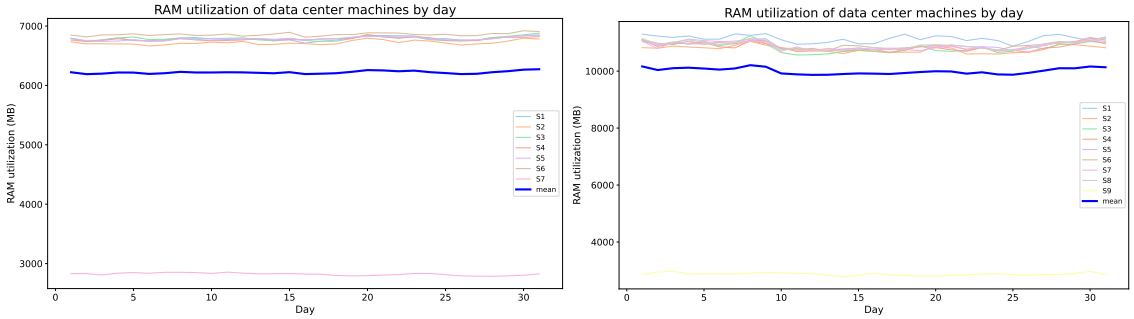
(b) RAM utilization of the servers of second layer.

Figure II.2: RAM usage layers



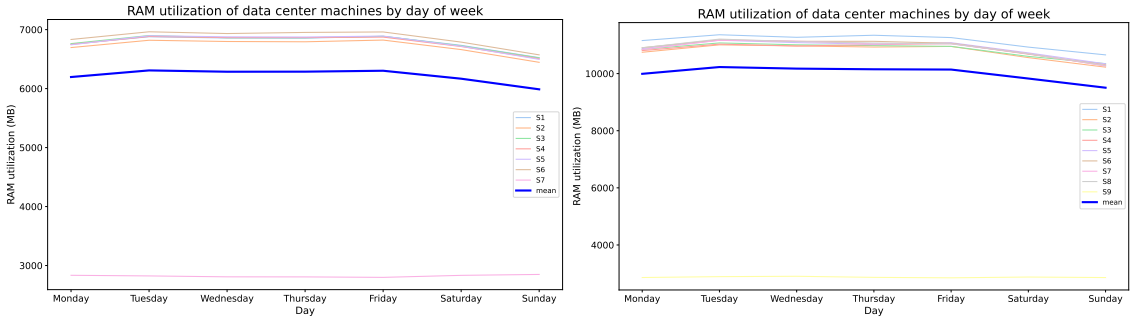
(a) Absolute RAM utilization of first layer. (b) Absolute RAM utilization of second layer.

Figure II.3: Absolute RAM utilization of servers (excluding the ones that serve as backup).



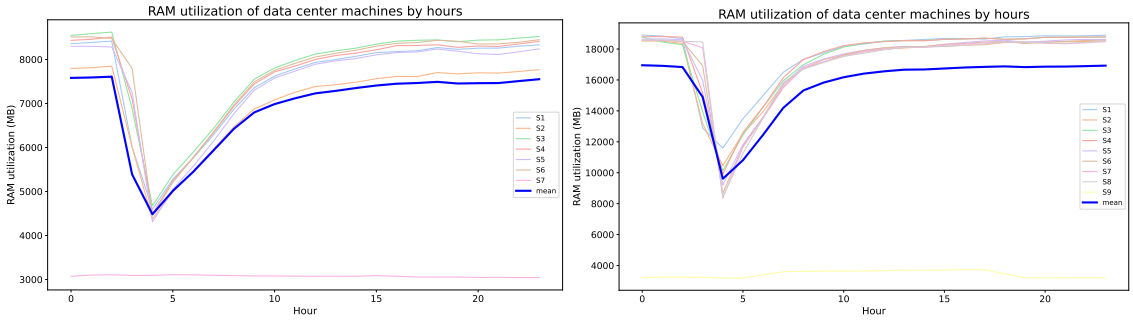
(a) RAM utilization of first layer by day of month. (b) RAM utilization of second layer by day of month.

Figure II.4: RAM utilization of data center machines by day of month



(a) RAM utilization of first layer by day of week. (b) RAM utilization of second layer by day of week.

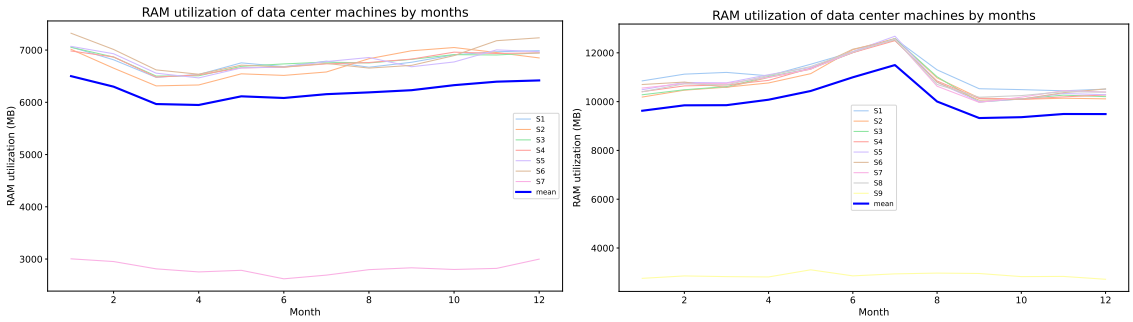
Figure II.5: RAM utilization of data center machines by day of week



(a) RAM utilization of first layer by hours.

(b) RAM utilization of second layer by hours.

Figure II.6: RAM utilization of data center machines by hours

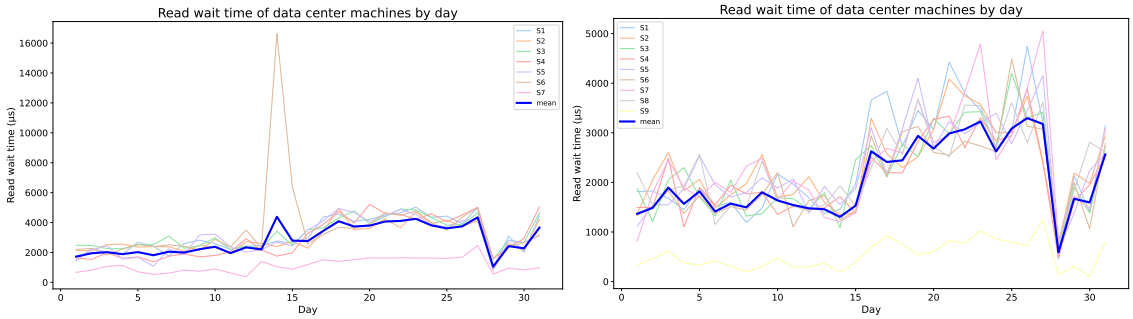


(a) RAM utilization of first layer by months.

(b) RAM utilization of second layer by months.

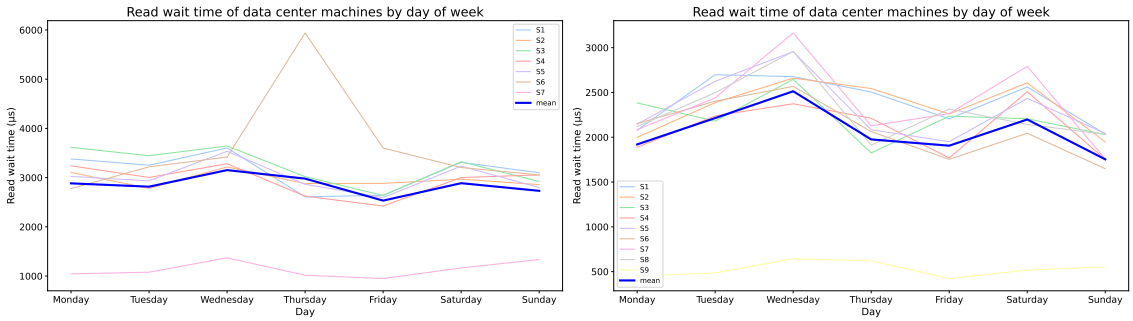
Figure II.7: RAM utilization of data center machines by months

II.2 Read wait time



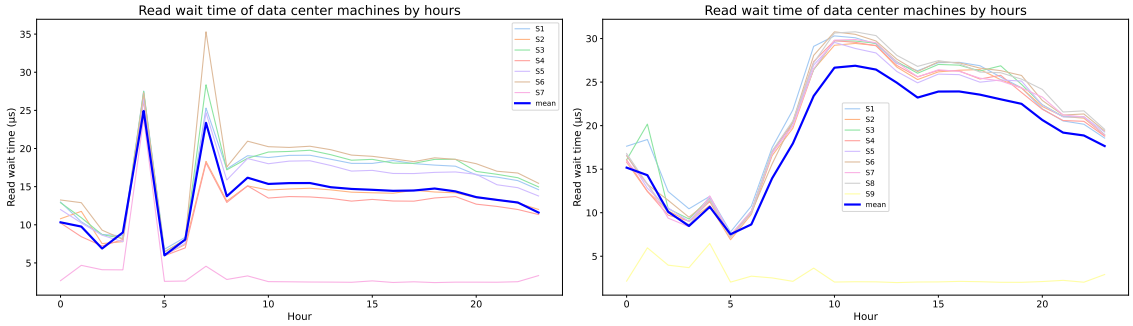
(a) Read wait time of first layer by day of month. (b) Read wait time of second layer by day of month.

Figure II.8: Read wait time of data center machines by day of month



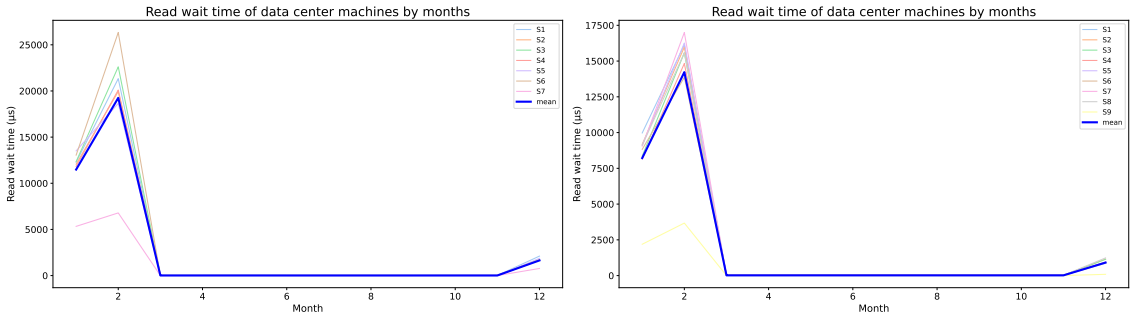
(a) Read wait time of first layer by day of week. (b) Read wait time of second layer by day of week.

Figure II.9: Read wait time of data center machines by day of week



(a) Read wait time of first layer by hours. (b) Read wait time of second layer by hours.

Figure II.10: Read wait time of data center machines by hours

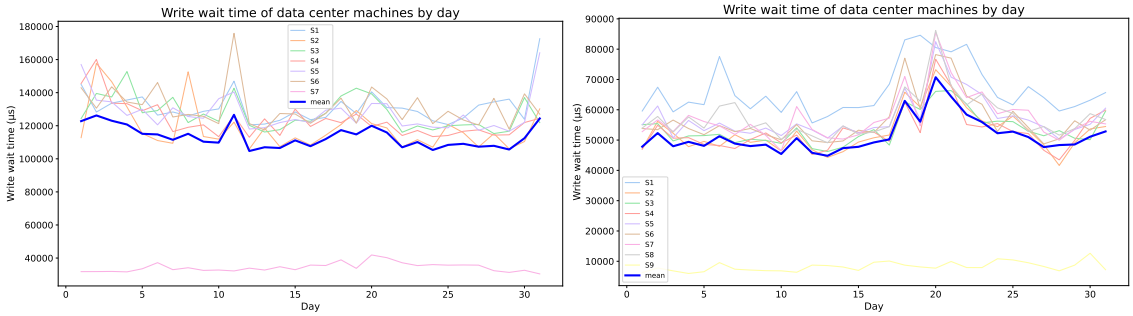


(a) Read wait time of first layer by months.

(b) Read wait time of second layer by months.

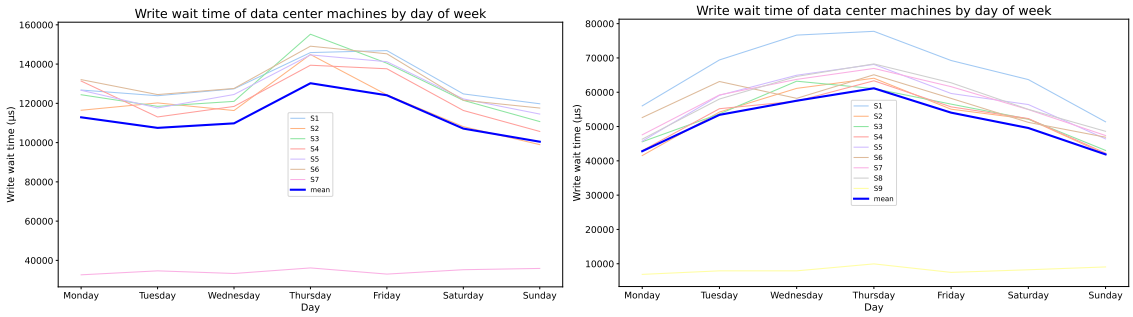
Figure II.11: Read wait time of data center machines by months

II.3 Write wait time



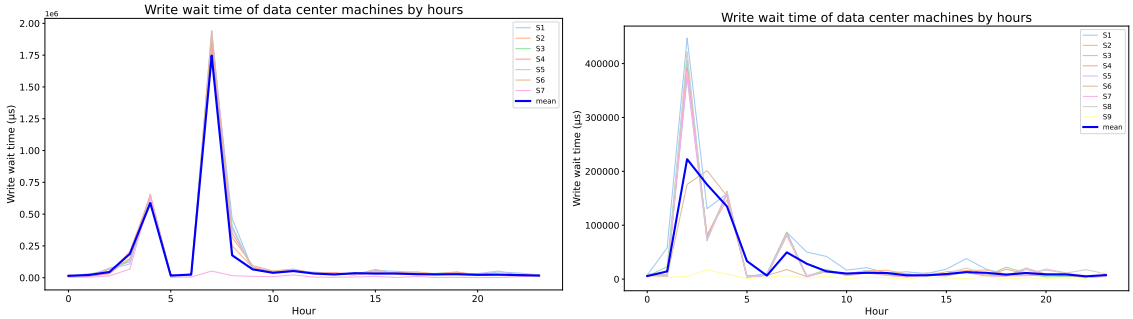
(a) Write wait time of first layer by day of month. (b) Write wait time of second layer by day of month.

Figure II.12: Write wait time of data center machines by day of month



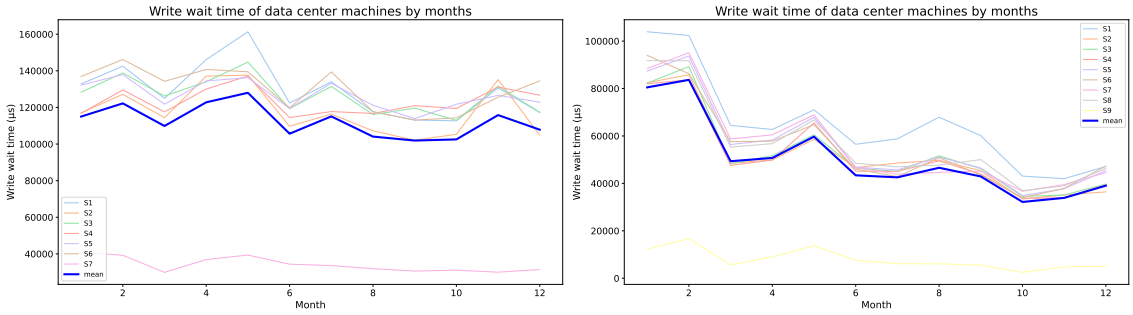
(a) Write wait time of first layer by day of week. (b) Write wait time of second layer by day of week.

Figure II.13: Write wait time of data center machines by day of week



(a) Write wait time of first layer by hours. (b) Write wait time of second layer by hours.

Figure II.14: Write wait time of data center machines by hours

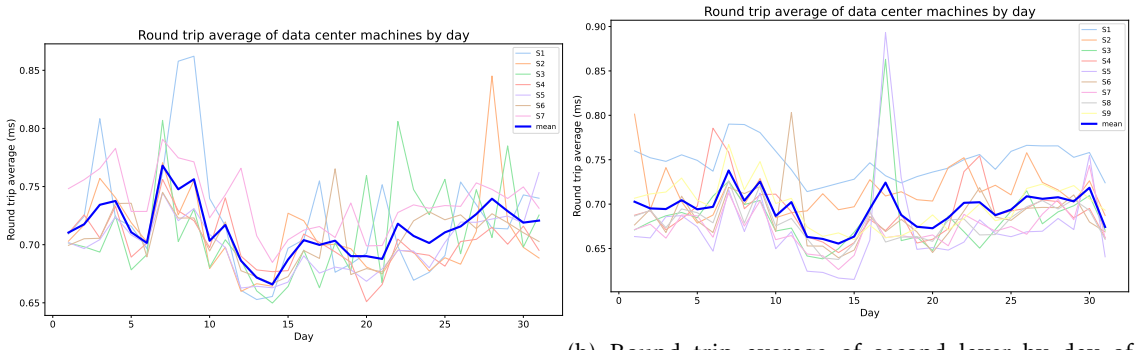


(a) Write wait time of first layer by months.

(b) Write wait time of second layer by months.

Figure II.15: Write wait time of data center machines by months

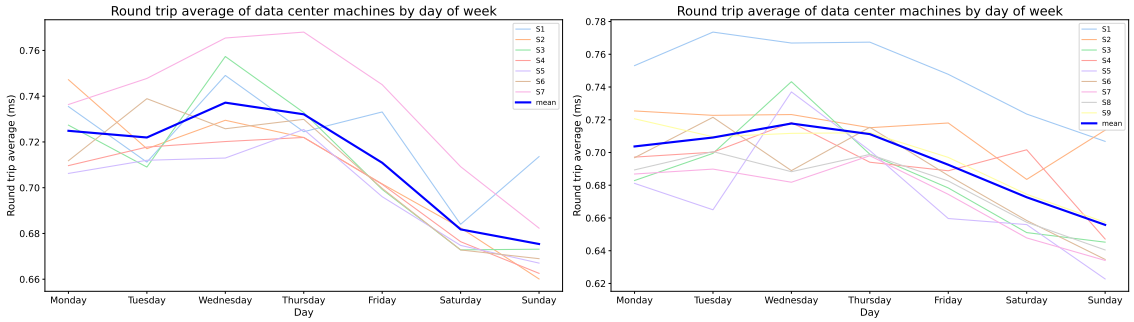
II.4 Round trip average utilization



(a) Round trip average of first layer by day of month.

(b) Round trip average of second layer by day of month.

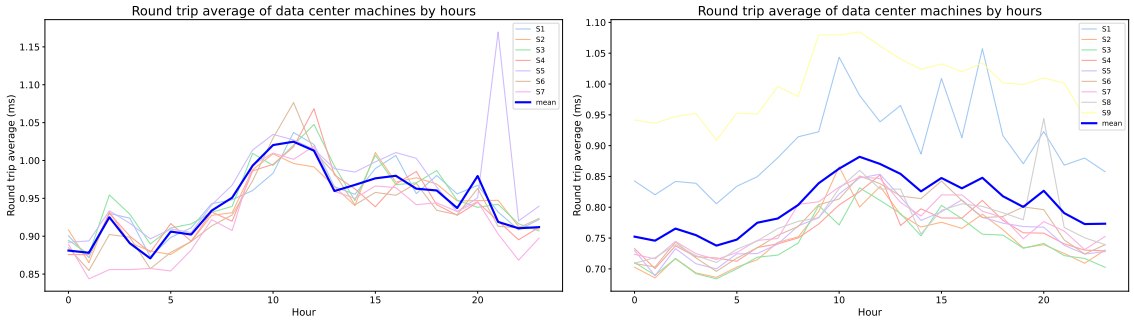
Figure II.16: Round trip average of data center machines by day of month



(a) Round trip average of first layer by day of week.

(b) Round trip average of second layer by day of week.

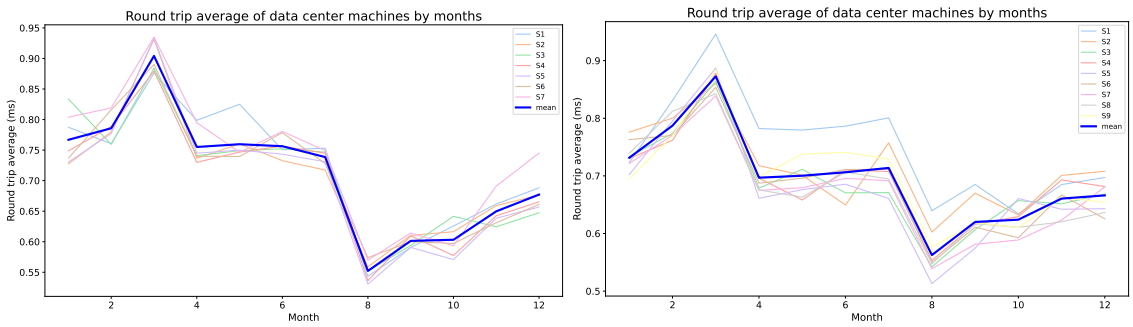
Figure II.17: Round trip average of data center machines by day of week



(a) Round trip average of first layer by hours.

(b) Round trip average of second layer by hours.

Figure II.18: Round trip average of data center machines by hours



(a) Round trip average of first layer by months.

(b) Round trip average of second layer by months.

Figure II.19: Round trip average of data center machines by months