UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Network-based high level classification: novel models and applications

Tiago Santos Colliri

Tese de Doutorado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP SÃO CARLOS

Tiago Santos Colliri

Network-based high level classification: novel models and applications

Thesis submitted to the Instituto de Ciências Matemáticas e de Computação – ICMC-USP – in accordance with the requirements of the Computer and Mathematical Sciences Graduate Program, for the degree of Doctor in Science. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Zhao Liang

USP – São Carlos
March 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

C713n

Colliri, Tiago
Network-based high level classification: novel models and applications / Tiago Colliri; orientador Liang Zhao. -- São Carlos, 2021.
167 p.

Tese (Doutorado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2021.

1. complex networks. 2. machine learning. 3. high level data classification. 4. corruption prediction. 5. COVID-19 prognosis. I. Zhao, Liang, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Tiago Santos Colliri

Classificação de alto nível baseada em redes: novos modelos e aplicações

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Doutor em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. Zhao Liang

USP – São Carlos
Março de 2021

ACKNOWLEDGEMENTS

I would like to acknowledge the Instituto de Ciências Matemáticas e de Computação (ICMC), from University of São Paulo (USP), for having accepted me as a Ph.D. student in the Graduate Program in Computer Science and Computational Mathematics (PPG-CCMC).

I thank Marcia Minakawa, from the Faculty of Public Health (FSP) of the University of São Paulo, for her valuable contributions to Chapter 8 of this thesis.

I also would like to kindly thank my thesis advisor, Prof. Zhao Liang, for his patience and confidence in my abilities, as well as for sharing his scientific experience with me. Without his guidance and support, this work would not have been possible.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

“Another man-ape came to life, and went through the same routine.
This was a younger, more adaptable specimen; it succeeded where the older one had failed.
On the planet Earth, the first crude knot had been tied.”
(Arthur C. Clarke, 2001: A Space Odyssey)

RESUMO

COLLIRI, T.S. **Classificação de alto nível baseada em redes: novos modelos e aplicações**. 2021. 167 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Aprendizado de máquina é uma aplicação da inteligência artificial com foco no desenvolvimento de programas de computador que podem acessar dados e usá-los para aprender por conta própria. Classificação de dados de alto nivel é uma técnica baseada na formação de padrão nos dados, ao invés de somente nas suas características físicas. Redes complexas têm se mostrado bastante úteis para caracterizar relacionamentos entre amostras de dados e, conseqüentemente, são um poderoso mecanismo de captura de padrões de dados. Neste trabalho, são investigadas novas maneiras de se usar a abordagem baseada em rede no desenvolvimento de técnicas de classificação de alto nível. Inicialmente, duas técnicas de classificação são introduzidas, e seus desempenhos são avaliados aplicando-as a conjuntos de dados de referência na área, tanto artificiais quanto reais, bem como comparando seus resultados com aqueles obtidos por modelos de classificação tradicionais, nos mesmos dados. Posteriormente, são exploradas as vantagens inerentes a este tipo de abordagem, tais como a sua versatilidade e interpretabilidade, para se desenvolver novas técnicas baseadas em rede especificamente projetadas para serem aplicadas em dados de problemas reais e relevantes em campos muito diversos, desde o mercado financeiro à corrupção de políticos e cuidados de saúde. Embora estes tipos de aplicação certamente requerem um esforço maior por parte dos pesquisadores, em termos do desafio e pré-processamento dos dados, acredita-se que elas são importantes para aproximar a pesquisa acadêmica da realidade. Entre os resultados obtidos neste trabalho, está a detecção de uma relação não esperada entre dados de votação de projetos de lei e condenações por corrupção e outros crimes financeiros entre deputados brasileiros. Também é demonstrado como é possível adaptar um modelo, que originalmente foi aplicado na detecção de periodicidade em dados meteorológicos, para identificar tendências de alta e de baixa no mercado de ações, acionando automaticamente uma ordem de compra ou de venda para o ativo, de acordo com a situação. Em outra investigação, é apresentada uma técnica para auxiliar os profissionais de saúde na tarefa de monitorar pacientes com COVID-19, por meio da detecção de sinais prévios de insuficiência hepática, renal ou respiratória, apenas com base nos resultados do exame de hemograma completo. Em resumo, acredita-se que este trabalho faz uma importante contribuição para o avanço do estudo de dados públicos em larga escala usando redes complexas.

Palavras-chave: redes complexas, classificação de dados de alto nível, aprendizado de máquina, partidos políticos, votações legislativas, predição de corrupção, mercado de ações, automação de investimentos, COVID-19, detecção de insuficiência, hemograma.

ABSTRACT

COLLIRI, T.S. **Network-based high level classification: novel models and applications**. 2021. 167 p. Tese (Doutorado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Machine learning is an application of artificial intelligence with focus on the development of computer programs that can access data and use them to learn for themselves. High level data classification is a technique based on data pattern formation, instead of only their physical features. Complex networks have been proven to be quite useful for characterizing relationships among data samples and, consequently, they are a powerful mechanism to capture data patterns. In this work, we investigate novel ways of using the network-based approach in the development of high level classification techniques. Initially, two classification techniques are introduced, and their performances are assessed by applying them to benchmark datasets, both artificial and real, as well as comparing their results to those achieved by traditional classification models, on the same data. Afterwards, we explore the inherent advantages offered by this type of approach, such as its versatility and interpretability, by developing novel network-based techniques specifically designed to be applied on data concerning real and relevant problems from very diverse fields, from the financial market to corruption among politicians and healthcare. Although these type of applications certainly require a greater amount of effort from the part of researchers, in terms of the challenge and data preprocessing, we believe they are important to bring academic research closer to the reality. Among our findings, there is the uncovering of an unexpected relationship between legislative voting data and convictions for corruption or other financial crimes among Brazilian representatives. We also demonstrate how one can adapt a model, which originally has been applied to detect periodicity in meteorological data, for identifying up and down trends in the stock market, automatically triggering a buying or a selling order for the asset, accordingly. In another investigation, a technique to help healthcare workers in the task of monitoring COVID-19 patients is presented, by detecting early signs of hepatic, renal or respiratory insufficiency solely based on Complete Blood Count (CBC) test results. In summary, we believe this work makes an important contribution to the advance of large scale public data study using complex networks.

Keywords: complex networks, high level data classification, machine learning, political parties, legislative voting, corruption prediction, stock market, stock trading automation, COVID-19, insufficiency detection, CBC test.

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

ε — Radius used in the ε-radius technique.

k — Number of neighbors (degree) of a vertice / number of neighbors considered in the kNN technique, depending on the context.

V — Number of vertices or nodes in the network.

E — Number of edges or links in the network.

$\mathscr{L}$ — Set of possible classes (labels).

$\mathscr{G}$ — Graph or network.

$\mathscr{V}$ — Set of vertices or nodes in the network.

$\mathscr{E}$ — Set of edges or links in the network.

X_{train} — Training set of data items.

Y_{train} — Training set of data labels.

X_{test} — Testing set of data items.

Y_{test} — Testing set of data labels.

u — Index for a network measure plugged into the network-based high level classifier (NBHL).

$|v|$ — Length of vector v.

$\bar{v}$ — Average of the values in vector v.

$\lfloor x \rceil$ — x rounded to the nearest integer.

W — Weight matrix for network edges.

$Q(A,i)$ — The i-th quantile of array A.

$H(p)$ — Entropy of probability distribution p.

$H(p,q)$ — Cross-entropy loss between a true probability distribution p and the estimated probability distribution q.

$D_{KL}(p \,||\, q)$ — Kullback-Leibler divergence between a true probability distribution p and the estimated probability distribution q.

CONTENTS

CHAPTER

1

INTRODUCTION

The human brain is constantly receiving new data and information from the real world. Most of these input data are discarded, during our daily routine (ATKINSON; SHIFFRIN, 1968). Some of this information are processed by the brain, oftentimes by considering other similar events happened in the past, already stored in the brain, and evaluating how they can possibly be related to the new apprehended information, in the present (ANDERSON, 2000). This phenomenon, although being continuous, is hardly consciously noticed, or even questioned, by us, given the unwitting nature of its occurrence (SEITZ; KIM; WATANABE, 2009). In machine learning-related research, we aim to emulate this same phenomenon, which naturally occurs in the human brain, such that it can also be performed by computers (SPICER; SANBORN, 2019).

A fundamental difference between the animal (human) brain and computers can be observed: traditional computer-based classification considers only the physical features, such as similarity, distance or distribution of the input data (JÄKEL; SCHÖLKOPF; WICHMANN, 2008; KRUSCHKE, 1992). On the other hand, brain-based classification takes into account not only physical features, but also the organizational structure of the data (LAKE; LAWRENCE; TENENBAUM, 2018), such as data patterns (ANDERSON *et al.*, 2004; LAKE *et al.*, 2017).

Complex networks have proven to be quite useful for characterizing relationships among data samples and, consequently, they are a powerful mechanism to capture data patterns (SILVA; ZHAO, 2015; CARNEIRO; ZHAO, 2017; BACKES; BRUNO, 2010). The objective or our research is to take advantage of this representation, by developing fast and efficient novel network-based high level machine learning techniques to perform predictive tasks, such as *regression* and *classification*. Besides of this, we have made a special effort to develop novel machine learning techniques specifically designed to tackle real-world phenomena, in very diverse areas of study.

Although various high level data classification techniques have already been proposed, in different works, the literature still lacks applications of these techniques to real-world problems.

Such type of application usually requires a minimum amount of knowledge regarding the specificities of the problem to be studied and its respective field, so that this information can be properly reflected in the built network. Moreover, it also involves additional data pre-processing tasks, such as data cleaning and editing, which, depending on the size of the input data, may become increasingly time-consuming. Within this context, we believe this work makes an important contribution, by not only introducing novel network-based high level machine learning techniques, but also by demonstrating their practical applications, on the modeling and analysis of relevant real-world phenomena.

In classic *supervised learning*, machine learning algorithms are designed to learn from examples. In this case, initially we have an input dataset X_{train} comprising n instances in the vector form, where each data instance itself is a m-dimensional vector, representing m features of the instance. Correspondingly, the labels of the instances are represented by another vector Y, with the same size of X_{train}. The objective of the *training phase* is to construct a classifier by generating a mapping $f : X_{train} \xrightarrow{\Delta} Y$, through some type of reasoning. In the *testing phase*, we use the classifier constructed so far to classify new data instances without label. The test dataset, denoted as X_{test}, is then used for assessing the performance of the model in the classification task. Therefore, the method used in the mapping process $f : X_{train} \xrightarrow{\Delta} Y$ is an important determinant for the performance of a supervised learning model.

In the first technique introduced in this work, each training data instance is mapped as a node in the network, and the edges are generated through a combination of two distance-based methods: kNN and ε-radius. A similar network construction method has already been employed in other related works (BACKES; BRUNO, 2010; BACKES; CASANOVA; BRUNO, 2013; SILVA; ZHAO, 2012a; SILVA; ZHAO, 2015; CARNEIRO; ZHAO, 2017). The novelty of this technique consists in detecting the impact patterns resulting from the insertion of each training data instance on the network structure, in terms of topological measures, for each class of the input dataset. In the testing phase, the model assigns a label for each new data instance by assessing the level of similarity between the impact provoked by its insertion in the network, in terms of topological structure, and the previously detected impacts patterns, for each class. Hence, this technique differs from traditional classification techniques in the sense that the classifier bases its decisions on the emerged impact patterns during the training phase, for each class, in terms of the network topological properties, in contrast to traditional ones, often based on distance and similarity measures.

Many networks encountered in the real world, such as social, computer and metabolic networks, are found to divide naturally into communities or modules (NEWMAN, 2006). One of the most effective ways of detecting these communities is through the modularity measure. In the second introduced technique, we investigate the possibility of conceiving a classification process which is solely based on the network's modularity score. For this end, two novelties are presented in this model: (1) instead of mapping each data instance as a node in the network, as

usual, we map each data instance's attribute as a node in the network; this is to conserve more information from the input dataset in the model's training phase, and (2) in the testing phase, the level of association of a new data instance to each class is measured in terms of its impact on each of the network components' modularity score.

The two techniques described above are evaluated by applying them to benchmark classification datasets, both artificial and real, and by comparing their performances with those achieved by traditional models, on the same data. The other four techniques introduced in this work are designed to be applied on real-world problems, from diverse areas such as politics, finance and public health. Following, we describe these techniques and their respective applications.

The House of Representatives is an important institution of the government's legislative branch, whose role is to embody the will of the population on the federal level. In democratic countries, it is expected to occur a natural alternation of power between political parties occupying the House seats over time. Within this context, the complex network approach can be highlighted as a suitable tool for analyzing congressional roll call voting records (ANDRIS *et al.*, 2015; MASO *et al.*, 2014; MOODY; MUCHA, 2013; WAUGH *et al.*, 2009). In this work, we make use of the network-based approach to analyze almost 30 years of legislative work from Brazilian representatives, in terms of roll-call votes data. Among our findings, we show that the changes verified in the topological structure of the congressmen network, in the last 28 years, follow very closely the main alternations of power between political parties in the presidency of Brazil. Additionally, our analyses are able to capture very clearly the lessening of influence of the Brazilian workers party (PT) in the House even months before the impeachment of Dilma Rousseff from the presidency. Another important finding in this work is with regard to the possibility of using this same roll-call votes-based network to predict possible convictions of corruption or other financial crimes among congressmen. We have performed tests in this sense using semi-supervised learning methods, through link prediction algorithms, and the obtained results indicate that this task is not only possible, but it can also achieve high accuracy rates.

The stock market has always been subject to a great number of studies aiming to predict future price movements, with the objective of optimizing the financial returns of trading operations. With the arrival of artificial intelligence, we have seen an increase in this type of studies using machine learning techniques (HUANG; NAKAMORI; WANG, 2005; ADEBIYI; ADEWUMI; AYO, 2014; LEE; JO, 1999). In this work, inspired by the concept of functional cartography (GUIMERA; AMARAL, 2005) and the formation of community structures, observed in natural and man-made systems (AKIKI; ABDALLAH, 2019; GLEISER; SPOORMAKER, 2010; HAGMANN *et al.*, 2008), we present a network-based model which makes use of connector hubs to detect price trend reversals in the market, thus allowing the detection of *up* or *down* trends for a stock, also triggering a buying or a selling operation accordingly. We have evaluated the performance of the model by applying it to a database comprising 10 of

the most traded stocks from both New York Stock Exchange (NYSE) and Brazilian Stock Exchange (Bovespa), and the obtained results are encouraging, with the resulting financial returns surpassing the ones corresponding to a "buy and hold" strategy for most of the stocks considered, and even by a high margin for some of them.

The last two models presented in this work were conceived to help in the fight against the COVID-19 pandemic, which arrived during the last year of our research. The first model makes predictions of new confirmed cases and deaths provoked by the virus on a specific region. Although there are already classical and well-known models available in network science specifically designed for such tasks (BARTHÉLEMY *et al.*, 2005; PASTOR-SATORRAS; VESPIGNANI, 2001; PASTOR-SATORRAS *et al.*, 2015), in this work, we have opted for developing a novel one, which yields predictions based on the COVID-19 curves from other regions whose past behavior is similar to the curve to be predicted. We start by building a temporal network, where each node corresponds to a different COVID-19 affected region, and nodes whose curves present similar variations, at the time step t, are connected to each other. Afterwards, a community detection algorithm is ran, and curves from nodes within a same community are used to compute future curve variations for a specific node. The model is evaluated by applying it to predict weekly new confirmed COVID-19 cases and deaths for the 27 federal units of Brazil, and the obtained results are satisfactory, when compared to other similar studies, in terms of mean absolute percentage error.

For the last application, we have developed a classification technique similar to the first one presented in this work, with two major improvements. The first improvement consists in adding a network reduction method in the model's training phase, based on the betweenness centrality measure (BRANDES, 2001), which has the role of both reducing its processing time and also discarding possible noise from the input dataset, by leaving only the most representative nodes in the generated network. The second improvement is in the design of a parameter optimization criteria, based on a variant of Kullback-Leibler divergence (KULLBACK; LEIBLER, 1951), to make the technique more adaptable to each dataset. We have applied the model to a dataset specially built for this study, obtained from the analysis of publicly available data of COVID-19 patients from one of the main hospitals in Brazil, with the objective of detecting early signs of renal, hepatic and respiratory insufficiency in infected patients using only Complete Blood Count (CBC) test scores. The obtained results in this task indicate that the model's performance is competitive, when compared to classic and the state-of-the-art classification techniques. Moreover, we consider the results achieved in this application as particularly relevant inasmuch as the proposed technique has the potential to help healthcare professionals who work in regions with scarce material and human resources to assess the severity of COVID-19 patients at high risk of complications, before the consequences become irreversible.

Therefore, as one may observe, we introduce different techniques in each study presented in this work, and also use a distinctive graph-formation approach in each of them, according to

the specificities of the problem to be analyzed. The way the network is built in the stock market investment problem, for instance, in Chapter 6, is completely different from the way we build the congressmen network, in Chapter 5. Additionally, the themes of the problems approached in this work may be very distinct from each other, and from very diverse fields, from economics to politics and public health. Nevertheless, they all represent relevant topics for the society, and have the common factor that all applications in this work are made by using publicly available data, i.e., that can be accessed by everyone. The importance of the first application comes from the possibility of predicting corruption cases among congresspeople solely by analyzing data regarding their voting records. The second application involves an already widely studied problem, from Economics, of whether stock price movements can be predicted or not. While the last two applications address the fight against the COVID-19 pandemic, under two different forms: (1) by predicting the number of new cases and deaths in a specific region, thus helping policy makers in the decision-making process regarding resources planning, and (2) by detecting early signs of insufficiency in infected patients, solely based on CBC test results, to help primary care workers in the severity assessment task.

1.1 Objectives

The use of networks for modeling real-world problems has been continuously increasing since the end of the 1990s, with the publication of the works from Watts and Strogatz (1998) and Barabási and Albert (1999). These two works are still considered the most important ones in the field of complex networks, as they were able to raise the network-based approach to a level much closer to the reality. The models introduced by them are inspired on feature patterns commonly encountered in real-world phenomena, such as the scale-free distribution, for example. Although the number of scientific works making use of network-based approaches has been growing for the last two decades, we believe that, given the versatility of this approach, there are still many open questions and large space for innovations and applications.

The main goal of this study is to develop novel machine learning techniques through network-based data pattern characterization, with a special focus on real-world applications. We aim to take advantage and to explore one of the main characteristics of complex networks, which is their adaptability to the study of problems from very diverse fields, from the financial market to politics and medical care. We will show that, depending on the specificities of each problem, the network-based framework offers many tools for approaching it in the most adequate way. For problems concerning static datasets, for instance, one can make use of the regular static networks for mapping the input data, while for problems regarding data stream or time series, with dynamic features, then it is usually more advisable to make use of a temporal network for modeling the phenomenon.

Most of the problems to be tackled in this study are related to classification or prediction

learning tasks, although we also explore applications related to clustering and regression, as well as semi-supervised learning, in the form of label propagation through link prediction. Following, we list and discuss the specific goals of our research:

1. Inspired by the hybrid network-based high level data classification technique, introduced by Silva and Zhao (2012a), we plain to develop a network-based classification technique which no longer requires a low level method for inferring the labels. The technique we propose still takes into account the impacts provoked by the insertion of new data instances on the network topology for classification purposes, with the difference that, instead of assigning the label whose class is the least impacted by the insertion, in terms of network measures, it assigns the label based on the impacts pattern, detected for each class. Our hypotheses hence is that there are different impact patterns for each class in the input dataset, and these emerging patterns, once detected, can be used for classification purposes.

2. The usual mapping process for converting a dataset into a network represents each data instance as a node in the network. This procedure, although is the most adopted one, has the drawback of oftentimes discarding information that might be useful for improving the classifier's performance. Therefore, we plan to develop a model in which the mapping process represents each data instance's attribute as a node in the network, and is able to infer the labels considering only the impacts on the modularity measure. Additionally, we plan to use an *attention mechanism* in this technique, such as the one used in the Transformer deep learning model (VASWANI *et al.*, 2017), to discriminate which edges generated during the testing phase should be considered more important in the classification process. Our hypotheses, in this case, is that keeping the attributes in the network mapping results in less information loss in this process and, consequently, contributes to increase the classification performance.

3. Considering that graph-based approaches have also been successfully applied to data regarding political analysis (ANDRIS *et al.*, 2015; MASO *et al.*, 2014; MOODY; MUCHA, 2013; WAUGH *et al.*, 2009), we will also delve into this field, and develop a model to analyze legislative voting data, using the historical votes of Brazilian representatives as data source. The goal here is to investigate whether the changes verified in the temporal network's topology, formed by the congressmen, may somehow reflect the main political events happened during the same period. We will also conduct simulations to assess the possibility of using such networks for predicting corruption predictions among politicians. Our hypotheses is that it is possible to predict future convictions solely based on the voting history of each congressman.

4. When it comes to prediction models, the stock market has always been subject to a great number of studies from the part of researchers. In this work, we do not deviate from this rule and will examine this problem as well, by extending and adapting an algorithm

originally intended to detect periodicity in time series (FERREIRA; ZHAO, 2014) in order to develop a model for detecting up and down price trends for an asset based on the topology of the network resulting from its price variation ranges. The proposed model also triggers a buying or a selling order, whenever a new up or down price trend is detected for the asset, respectively. In this study, our hypotheses is that connector hubs, in the built price variations network, can be used as indicators for potential price trend reversals in the market.

5. During the time we conduct our research, the world had witnessed the surge of an unexpected pandemic, caused by COVID-19, which brought severe consequences to the population, and whose death toll has already surpassed two million deaths around the world (WORLDOMETER, 2021). Naturally, this phenomenon attracted the attention of researchers, in many fields, with the objective to contribute for helping to better understand and to fight the disease. In this research, we plan to contribute to this cause through the development of two different models:

   a) A regression model, to predict new COVID-19 cases and deaths in a specific region, through a network-based multiple regression analysis. The hypotheses here is that one can predict the COVID-19 curve variation in a region by considering the curves from other affected regions which presented similar variations in the past.

   b) A classification model, to help medical workers in the task of monitoring COVID-19 patients, by detecting early signs of renal, hepatic or respiratory insufficiency based on Complete Blood Count (CBC) test results. The hypotheses in this study is that the results from these tests can be used as biomarkers to detect each type of insufficiency on COVID-19 patients independently, thus helping healthcare workers in the disease prognosis.

1.2 Motivations

Thanks to the constant advances in the processing power of computers, the overall expectation is that the area of artificial intelligence will become increasingly important in the future, with consequences which may affect our lives significantly. Most human activities involving data analysis are expected to be performed by computers, with more efficiency and accuracy. Within this context, network-based machine learning techniques offer the advantage to not only analyze physical features of the input data (e.g., distance or distribution), as in traditional machine learning techniques, but to also consider the pattern formation in their topological properties (SILVA; ZHAO, 2012a; SILVA; ZHAO, 2016). Besides, as this is intrinsically a graphical approach, it has the convenience of being *interpretable*, which is not the case with other machine learning approaches, such as neural networks (GURESEN; KAYAKUTLU; DAIM, 2011).

The main motivation for this work comes from our belief that, given its versatility and the fact that this is yet a relatively new field of study, there is still a large space for exploration in complex networks. This framework has been proven to be quite useful for characterizing relationships among data samples and, consequently, it is a powerful mechanism to capture data patterns. The hybrid network-based classification technique introduced by Silva and Zhao (2012a), Silva and Zhao (2015) combines a low level of learning method, which can be implemented by any traditional classification technique – such as naive Bayes or SVM – and a high level of learning method, which makes use of network data representation in order to identify the pattern formation, in terms of the network topology structure, for each different class in the dataset. Their results have shown that the high level term of the hybrid classifier is able to improve the performance of the low level classifier, especially in situations where the class configuration's complexity increases, causing more mixtures among different classes. In this work, we plan to extend their research, by investigating whether it is possible to get rid of the low level method in the classification process, as well as the possibility of automatizing the values of parameters ε and k, used for building the network.

Another strong motivation factor for us is in the prospect of using the network-based approach to identify relationships, patterns, trends and other useful information in data concerning real-world problems, in an effort to bring academic research closer to the reality. When we considered the problem of analyzing legislative voting data, for instance, we did not expect, at first, to uncover a hidden relationship between the Brazilian representatives votes and convictions for corruption and other financial crimes. This finding was only possible due to the network-based approach, which allowed us to notice the formation of some sort of "corruption neighborhoods" in the congressmen built network.

Another interesting remark, still concerning "hidden relationships or patterns" between real-world data, is with regard to how the model for stock trading automation, also presented in this work, was conceived. We started by applying an algorithm, which was originally applied to detect periodicity in time series concerning meteorological data (FERREIRA; ZHAO, 2014), for possibly detecting cycles in stock market related data, with a few adaptations in the code structure. Later, the idea of automatically triggering buying or selling operations for the stock was conceived by using the concept of network cartography (GUIMERA; AMARAL, 2005), which is based on observations made in metabolic networks of organisms. Therefore, it is possible to say that the financial trading model presented in this work is an adapted combination of two different studies, from very diverse fields.

1.3 Organization of the Remainder of the Document

The remainder of this thesis is organized as follows. In Chapter 2, we review some relevant concepts and techniques from the topics related to our research, such as complex

networks and machine learning, and also discuss some of the works which have inspired us to develop our research. In Chapter 3 and Chapter 4, two different network-based classification techniques are presented, and their performances are assessed by applying them to benchmark datasets and comparing to the ones achieved by traditional classification models. The first technique, introduced in Chapter 3, is based on the detection of the impacts pattern provoked by the insertion of new data instances in the network's topology, for each class. The second technique, introduced in Chapter 4, has the novelty of, instead representing each data instance as a node in the network, as usual, it represents each data instance's attribute as a node in the network, and the label inference is based solely on the modularity measure. In Chapter 5, we present a model for analyzing legislative voting data, obtained from the Brazilian House of Representatives, and also demonstrate how this type of modeling can be used for the sake of predicting the incidence of convictions for corruption and other financial crimes among congressmen. In Chapter 6, we approach a problem regarding the financial market, by introducing a model for helping investors to identify up and down trends for the price of a stock, also triggering a buying or selling operation, accordingly. In Chapter 7, we focus on the problem of predicting the incidence of new cases and deaths provoked by COVID-19, in a specific region. For this end, a temporal network is employed, representing the evolution of the time series correlations between different regions, pairwise, and the performance of the model is evaluated by applying it to data regarding COVID-19 numbers from the 27 federal units of Brazil. The last model introduced in this work, in Chapter 8, also concerns the COVID-19, but this time we focus on the treatment of the disease. More specifically, we present a model to help healthcare workers in the task of monitoring COVID-19 infected individuals in order to detect possible early signs of renal, respiratory or hepatic insufficiency. The main advantage of the proposed technique lies in the fact that it makes use of only Complete Blood Count (CBC) test results to perform the analysis, which are considered easy to be collected and also cheaper, when compared to other additional tests. At the end of this document, in Chapter 9, we close our study by adding some final remarks.

CHAPTER

2

RELEVANT CONCEPTS AND TECHNIQUES

The topics of our research are contextualized within the great area of *artificial intelligence*. This area can be defined as the development of computational techniques that perceive their environment and take actions that maximize the chance of successfully achieving their goals (RUSSELL; NORVIG, 2016; POOLE; MACKWORTH; GOEBEL, 1998). Informally, the term "artificial intelligence" is also applied for machines able to mimic cognitive functions usually associated to humans, such as learning and problem solving (RUSSELL; NORVIG, 2016).

In this chapter, we present the most relevant concepts and techniques related to the topics of our research. In section 2.1, we introduce the reader to the topic of *complex networks*, describing how this field of study first arrived, as well as reviewing some of its most relevant works until today. We also present, in this section, some of the measures most commonly used in the analysis of networks, and provide some examples of real-world applications. In section 2.2, we familiarize the reader with some relevant concepts and tasks regarding the topic of *machine learning*. In section 2.3, we introduce the concept of *high level* data pattern characterization and discuss the importance of this concept when developing network-based classification techniques, while also reviewing some related works. Finally, in section 2.4, we discuss some examples of *graph-formation techniques* on supervised learning, describing and comparing different forms of performing this task, according to each type of application.

2.1 Complex Networks

The origin of the *graph theory* traces back to 1736, when the Swiss mathematician Leonhard Euler published the solution to the Königsberg bridge problem (BOCCALETTI *et al.*, 2006). This initial work led to the development of other important works, such as the ones made by Vandermonde (1771), Cauchy (1813) and Huillier (1861), which ended up resulting in the creation of a new branch of discrete mathematics, known as *topology*. Later, one of Konig's students, Paul Erdös, would later become responsible for the introduction of probabilistic

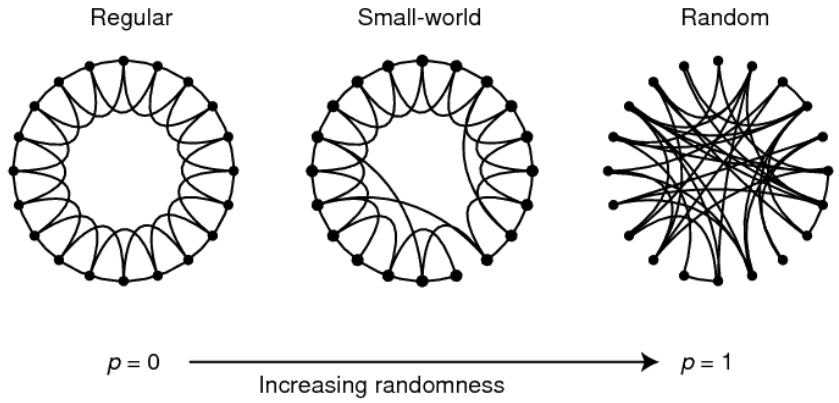
methods in graph theory (ERDÖS; RÉNYI, 1960), while Cartwright and Harary (1956) helped to standardize the terminology of graphs, as well as to broaden its reach by including and demonstrating possibilities of applications in several branches of science. In 1995, the field of complex networks was included as one of the main topics of the area of *complex systems*, i.e., dynamical systems that present properties such as *adaptation*, *emergence* and *transition*. This inclusion was certainly induced by the advances of computer processing powers, opening the possibility to study the properties of large databases of real networks. In 1998 and 1999, there were two papers responsible for great advances in this field, published by Watts and Strogatz (1998), on small-world networks, and by Barabási and Albert (1999), on scale-free networks. Nowadays, new studies in this field have the potential to offer novel tools and perspectives for a wide range of scientific problems, from social networking to drug design (BARABÁSI *et al.*, 2016).

The term *complex network* refers to a graph consisting of a large number of *nodes* (or vertices) joined by *links* (or edges), with a non-trivial topology (ALBERT; BARABÁSI, 2002). Some examples of complex networks include the internet (FALOUTSOS; FALOUTSOS; FALOUTSOS, 1999), biological neural networks (SPORNS, 2002), social networks among individuals (CARRINGTON; SCOTT; WASSERMAN, 2006), food chains (MONTOYA; SOLé, 2002), blood distribution networks (WEST; BROWN; ENQUIST, 2009) and power grid distribution networks (ALBERT; ALBERT; NAKARADO, 2004). The study of the topology and the dynamics of these networks allows us to a better understanding and characterization of the phenomena involved in these systems.

The *Erdös-Rényi model* for generating random networks is considered the first model in this field, with the use of relatively simple algorithms. The first algorithm generates a random network $\mathscr{G}$ from a predefined certain number of vertices V and number of edges E (ERDÖS; RÉNYI, 1960). The second algorithm of this model, introduced by Gilbert (1959), generates a random network $\mathscr{G}$ from a predefined certain number of vertices V and probability of connection p between each pair of nodes. The degree distribution of random networks, when $V \gg \langle k \rangle$, i.e., when the number of vertices is much bigger than the network average degree, follows a Poisson distribution, with the form $P(k) = e^{-\langle k \rangle} \frac{\langle k \rangle}{k!}$. Although not originally intended for modeling real systems, random networks are still in use nowadays, specially for comparison purposes.

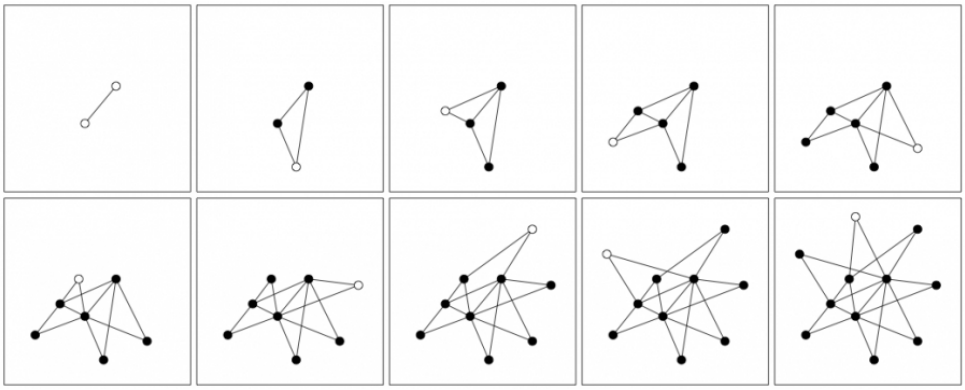
At the end of the 1990's period, researchers approach the challenge of comparing the properties of real networks with graph theoretical models, which led to the introduction of two important models in this field. The *Watts-Strogatz model*, also known as the *small-world model*, is the first model introduced in this period, with the main message that "real networks are not random" (WATTS; STROGATZ, 1998). Its algorithm starts with a regular network, such that each node has the same degree k, being connected to $E/2$ neighbors in each direction. Then, each edge in the clockwise direction is rewired to any other node with a probability p. As we raise the value of p from 0 to closer to 1, more random the network becomes, with the small-world

Figure 1 – The *small-world* model



Source: Watts and Strogatz (1998).

Figure 2 – The *scale-free* model



Source: Barabási *et al.* (2016).

property being more present for values in between (Figure 1). The networks generated by this model are also random, with the difference that they present small-world properties.

Other important model introduced in the 1990's period is the *Barabási-Albert model*, also known as the *scale-free model* (BARABÁSI; ALBERT, 1999). The main message of this model is that the "network structure and evolution are inseparable". It also introduces the concept of "preferential attachment", where at each timestep we add a new node i with n links that connect him to m nodes already in the network (Figure 2). The probability of the node j to receive an edge from node i is $P(k_i) = \frac{k_i}{\sum_j k_j}$. This combination of growth with preferential attachment results in the so-called *rich-gets-richer* phenomenon. This happens because more connected vertices have higher probability to receive edges from new vertices. The degree distribution of scale-free networks follows a power law, with the form $P(k) \sim k^{-\gamma}$. Many real networks are consonant with this model, such that in most cases $2 < \gamma \leq 3$.

Table 1 – Examples of measures to analyze the network according to each level of abstraction.

Type of Analysis	Objective	Examples of Measures
Element-level	Methods to identify the most important nodes of the network.	All centrality measures, such as: *degree centrality* (k_i), *closeness centrality* (C_i), *betweenness centrality* (Bt_i) and *eigenvector centrality* (π_i).
Group-level	Involves methods for defining and finding cohesive groups of nodes in the network.	The most typical is the *local clustering coefficient* (cc_i).
Network-level	Focuses on the topological properties of the network as a whole.	*Average degree* ($\langle k \rangle$), *transitivity* (C), *average local clustering coefficient* ($\langle cc_i \rangle$), *diameter* (d), *average shortest path length* (l), *assortativity* (r) and *modularity* (Q).

Source: Research data.

In order to characterize a network, we can look at its topological structure, by extracting some measures to help us learning more about the network, as well as comparing it with other networks. This type of analysis can be made at three different levels of abstraction (Table 1). A very important characteristic of all these measures is their *universality*, i.e., they can be used in the analysis of all types of networks, disregarding the area from the data used for generating the network.

2.1.1 Commonly Used Network Measures

The decision regarding which measure one should use when analyzing the network topological structure depends on the specificities of the problem being studied. Below, we list some of the most commonly used network measures, along with their respective definitions. For a more complete review of measures used in the characterization of complex networks, one can refer to the survey made by Costa *et al.* (2007).

- *Degree centrality* (k_i): considered the simplest measure, is the number of links incident upon a node. If a communications network is under attack, for instance, in an attempt to deliberately damage the communication system, it is expected that the main targets are the most connected nodes (ALBERT; JEONG; BARABÁSI, 2000).

- *Closeness centrality* (C_i): the average length of the shortest path between the node and all other nodes in the graph. Examples of application include road networks modeling (ABRAHAM *et al.*, 2011) and vehicle routing problems (CHABRIER, 2006).

- *Betweenness centrality* (Bt_i): broadly speaking, estimates the extent to which a vertex lies on paths between other vertices. Applications of this centrality measure include

identifying the most influential individuals in social networks (GOH *et al.*, 2003) and predicting preferential attachment in scientific coauthorship networks (ABBASI; HOSSAIN; LEYDESDORFF, 2012).

- *Eigenvector centrality* (π_i): a measure of the influence of a node in a network. The Google's PageRank algorithm, for instance, is a variant of the eigenvector centrality (PAGE *et al.*, 1999). It had also been successfully applied for analyzing connectivity patterns in fMRI data of the human brain (LOHMANN *et al.*, 2010).

- *Local clustering coefficient* (cc_i): quantifies how close the node's neighbors are to being a clique (complete graph). It is commonly used in preferential attachment models (KROT; PROKHORENKOVA, 2015).

- *Average degree* ($\langle k \rangle$): a relatively simple measure, which statistically quantifies the average degree of the vertices of a component.

- *Transitivity* (C): also called *global clustering coefficient*, measures the total number of closed triangles in a network. It is commonly applied on link prediction techniques (LIU; LÜ, 2010).

- *Average local clustering coefficient* ($\langle cc_i \rangle$): quantifies the degree to which local nodes in a network tend to cluster together.

- *Average shortest path length* (l): the average minimum number of edges separating all nodes in a fully connected component.

- *Assortativity* (r): numerically translates the preference for vertices of a network to attach to others that are similar or different regarding the vertices's degree in a structural sense (NEWMAN, 2003). Examples of applications include the analysis of collaboration networks (CHANG *et al.*, 2007) and in the identification of sentiment expression patterns in Twitter users (BLISS *et al.*, 2012).

- *Modularity* (Q): broadly speaking, compares the number of connections between vertices which share a same characteristic with the expected number of connections when occurred randomly. One important application of this measure is on the detection of the optimal community structure in a network (NEWMAN, 2006).

2.1.2 Network-Based Modeling Applied to Real-World Phenomena

In this work, we give a special emphasis on the use of complex networks for modeling and analyzing real-world phenomena. Therefore, in this section, we are going to review some examples of works which also have focused on this type of application, in different areas of study. For a more complete review in this theme, one can refer to the work made by Costa *et al.* (2011).

Computational analyses of datasets regarding large-scale connection patterns in the cerebral cortex of various mammalian species – such as most cortical regions of the macaque monkey, cat, and rat – have revealed a broad range of network characteristics, including the existence of clusters of brain regions, hierarchical organization, small-world attributes, distinct functional streams, motifs, and real contributions to global network measures (SPORNS; TONONI; KÖTTER, 2005). These studies have also provided a comprehensive structural description of the network of elements and connections forming the human brain, introducing what is known today as the human connectome, thus contributing to understanding the network organization of the brain, both in anatomical, structural and functional terms (BULLMORE; SPORNS, 2009).

Guimera and Amaral (2005) analyzed the metabolic networks of twelve organisms from three different superkingdoms, and their findings have led to the introduction of the concept of "network cartography". The study shows that biological networks may present functional modules, with nodes having different universal roles according to their pattern of intra- and inter-module connections. In this form of representation, typically, the great majority of nodes are non-hubs, being only connected to other nodes within their respective modules, and just few of them are hubs. Additionally, non-hubs and hubs can be sub-categorized into seven different roles, according to their respective modules and their connections to nodes from other modules. We make use of concepts from the network cartography analysis in the Chapter 5 and Chapter 6 of this thesis.

Ribeiro *et al.* (2018) have built a network formed by Brazilian politicians involved in corruption scandals, based on an extensive research of a 27-years time range documentation. The analysis of this network topology indicates that corrupted politicians tend to operate in small groups, with only a few hubs and a modular structure that often encompasses more than one corruption scandal. They also show that the vertex degree distribution is similar to an exponential one, and presents abrupt changes when a new political party takes power. Moreover, they also find out that the dynamical structure of the network can be used for successfully predicting partners in future scandals. The study presented in the Chapter 5 of this thesis is inspired on this work.

Fang, Sivakumar and Woldemeskel (2017) made use of a network-based method to classify catchments in the large-scale basin formed by the Mississippi River. Six community structure detection algorithms were applied and had their performances assessed, in terms of consistency. The obtained results were promising, also indicating that, in addition to geographic proximity, network topology, represented by organization of the river (e.g. main stem, river branching), also plays an important role in the formation of different communities of catchments.

There are also interesting studies with focus on the simulation of real phenomena, such as the spreading of epidemics (BARTHÉLEMY *et al.*, 2005), computer viruses (ALBERT; JEONG; BARABÁSI, 2000) and rumors (MORENO; NEKOVEE; PACHECO, 2004). The *SIR* model, for instance, is commonly applied for simulating the spreading of infectious diseases, and represents

Table 2 – Some tasks associated to machine learning.

Type of Learning	Tasks	Examples of Methods
Supervised	Classification, Prediction, Regression	Support Vector Machines, Decision Trees, kNN, naive Bayes, Regression Analysis, Multilayer Perceptron.
Unsupervised	Clustering, Association, Anomaly Detection	k-means, DBSCAN.

Source: Research data.

each individual of a population as a node in the network. There are three possible classes for each node during the simulation: susceptible (S), infected (I) or recovered (R). At each time step in the modeling, if a susceptible node has contact with an infected one, then he also becomes infected, with a probability β. Additionally, at each turn, infected vertices may become recovered (or immunized) independently, with a probability μ. A similar methodology is used in the rumor propagation model, also known as the *xyz* model.

2.2 Machine Learning

Machine learning (ML) is an application of *artificial intelligence*, focusing on the study of computer algorithms that can learn from data and improve automatically through experience (BISHOP, 2006; MITCHELL, 1997). It often makes use of mathematical models and statistical techniques to give computers the ability to "learn" (i.e., progressively improve the performance on a specific task) with data, without being explicitly programmed (SAMUEL, 1959). Another accepted definition for machine learning is as techniques to improve algorithms such that they enhance their performance as they "acquire more experience" (ALPAYDIN, 2010). The main categories of ML tasks are: *supervised learning*, in the forms of *regression*, *prediction* and *classification*, and *unsupervised learning*, in the forms of *clustering*, *association* and *anomaly detection* (Table 2).

The objective in *supervised learning* is to induce concepts from a dataset with classes already known and labeled. When the class labels are composed by discrete values, the task is denominated *classification*, and when they are composed by continuous values then the task is denominated *regression*. In *unsupervised learning*, the main objective is to group the data according to some specific similarity criteria, since there is no previous knowledge about the classes in the sample (MITCHELL, 1997). There are still cases when only part of the classes in the dataset are known and labeled, for these cases there is an alternative category called *semi-supervised learning*. This type of learning tries to propagate the known labels to unlabeled classes in the dataset, with the objective of reducing the work of the specialist on labeling the examples (CHAPELLE; SCHÖLKOPF; ZIEN, 2006). It also can be employed with the objective

of predicting certain phenomena, which, in this case, are represented by the labels.

Most machine learning algorithms are associated with *induction*, i.e., reasoning from observed training cases to general rules, which are then applied to the test cases. However, there are also approaches, specially in semi-supervised learning, which make use of *transduction*, i.e., reasoning from observed, specific training cases to specific test cases. Examples of transductive learning approaches include transductive SVM (GAMMERMAN; VOVK; VAPNIK, 1998) and graph-based label propagation algorithms (ROSSI; LOPES; REZENDE, 2016; VEGA-OLIVEROS *et al.*, 2014; TALUKDAR; CRAMMER, 2009).

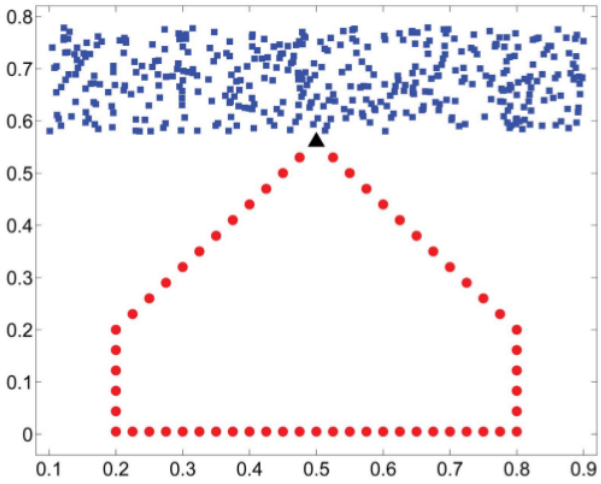
Another important aspect to be considered in machine learning are the concepts of *bias* and *variance*. Models with high bias trained on data with low variance are unable to capture the underlying pattern of the data, which leads to *underfitting*. On the other hand, models with low bias trained on data with high variance may lead to *overfitting*. Therefore, if one needs to approach a problem using machine learning techniques it is always important to have in mind the *bias-variance tradeoff* when searching for a solution. This can be done by both trying to extract the most adequate features from the dataset, according to each problem, and by building a model which is not too simple or too specific, regarding the training data, for avoiding underfitting and overfitting, respectively.

2.3 Network-Based High Level Classification

Traditional machine learning techniques consider only physical features (e.g., distance or similarity) of the input data. Therefore, this type of learning can be categorized as *low level techniques*. Machine learning techniques that consider not only physical attributes but also the pattern formation are referred to as *high level techniques* (Figure 3). Within this context, the pattern formed by the topological structure of complex networks allows the high level learning process. To be more specific, in a classification task, for instance, two data samples will be classified into the same class if they are constituent elements of a well defined pattern, no matter how far they may stay from each other from the perspective of their physical features. Hence, the former is referred to as *low level* machine learning techniques while the latter *high level* ones.

Some of the works related to high level network-based machine learning techniques include: semi-supervised learning (ZHU, 2005; CHAPELLE; SCHÖLKOPF; ZIEN, 2006; SILVA; ZHAO, 2012b), clustering (LIU *et al.*, 2018; PALLA *et al.*, 2005; FORTUNATO, 2010; KARYPIS; HAN; KUMAR, 1999; SCHAEFFER, 2007; SILVA; ZHAO, 2012c; SILVA; ZHAO; CUPERTINO, 2013), regression (LOGLISCI; MALERBA, 2017; CASIRAGHI, 2017; NI; YAN; KASSIM, 2012; GAO *et al.*, 2014) and classification (ANGHINONI *et al.*, 2019; VERRI; URIO; ZHAO, 2016; CARNEIRO; ZHAO, 2017; SILVA; ZHAO, 2012a; NETO; ZHAO, 2013; SILVA; ZHAO, 2016). Given its feature of being able to adapt to several types of problems, from different areas of application, we believe there are still many possibilities to be explored when it comes to

Figure 3 – Representation of a dataset containing two classes, with one of them presenting a clear pattern formation. Traditional classification techniques would have problems on identifying the pattern and consequently labeling the black bold triangle data item as belonging to the red class, while high level techniques are expected to identify it correctly.



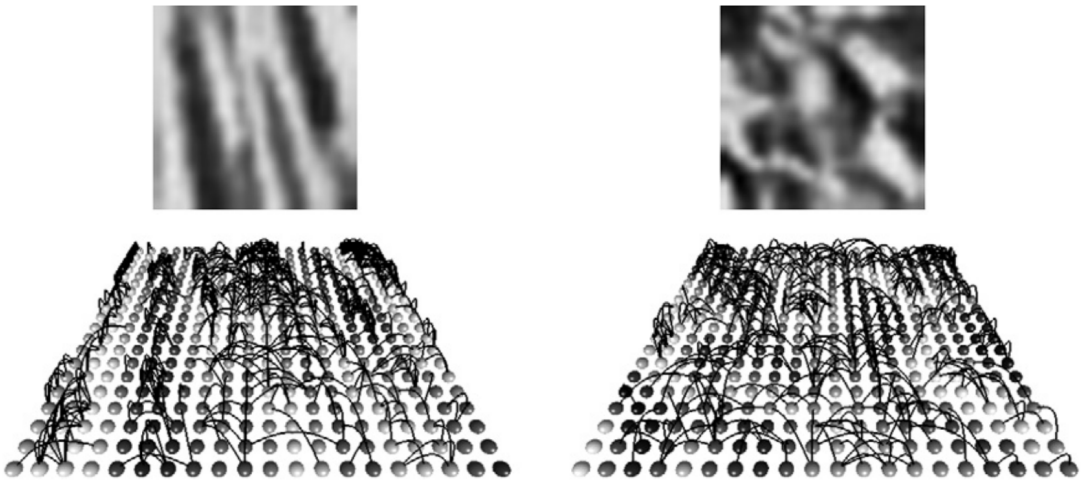
Source: Silva and Zhao (2012a).

the use of complex networks on machine learning.

Backes, Casanova and Bruno (2013) have used the complex network-based approach to develop a model for analyzing and classifying image textures. This is because, for this task, one needs to take into account not only the isolated micro-textures of the image, but also the relation among them and their neighbors. Thus, they took advantage of this approach to represent and characterize the relation among structural elements of texture in the form of a complex network. In their work, the classification model starts by converting the texture to a pixel network, where each node represents a pixel and the edges between a pair of nodes are generated by using the ε-radius technique, based on the Euclidean distance between their respective x and y coordinates in the image (Figure 4). Local textures detection and analysis are made by associating a weight to each edge, considering both the distance and gray level similarity between each pair of connected nodes. The topology characterization of each network is made by computing four features (or measures) from their degree histogram: mean, contrast, energy and entropy. A set of weights thresholds T is also applied to each network, in order to obtain different samples which, at the end, will represent the network's final signature. The optimal values for parameters ε and T were achieved by evaluating the model's performance according to different configurations for both of them. The obtained results show that the proposed method is robust, outperforming traditional texture classification methods. A similar approach had also already been used in the classification of shape images, with equally encouraging results (BACKES; BRUNO, 2010).

Silva and Zhao (2012a), Silva and Zhao (2016) introduced a new technique based on the extraction of features of the underlying network constructed from the input data, which not only can realize classification according to the pattern formation, but also is able to improve the

Figure 4 – Two examples of complex networks generated from different image textures.



Source: Backes, Casanova and Bruno (2013).

performance of traditional classification techniques. In their work, each class from the training data forms an isolated network component, and each of these components calculates the changes that occur in its pattern formation upon the insertion of a new test instance. If a slight change or no change occurs, then the test instance is said to conform to the formed pattern of that class. Consequently, the high level classifier produces a large relevance value for that test instance in that class. On the other hand, if these changes drastically modify the formed pattern of the class, then the high level classifier produces a small value of membership for that test instance in that class. In the work from Silva and Zhao (2012a), three network measurements were used to analyze these changes: average degree ($\langle k \rangle$), clustering coefficient (C) and assortativity (r). While in the work from Silva and Zhao (2015), the network measurements are extracted from a tourist walk in the network. Therefore, a major problem that this work left open is how one could choose other network measures in an intuitive way, and also how to define the weight of influence that is given to each of them. Moreover, the classifier from this model is not entirely high level yet, since it needs a second opinion from a low level classifier before yielding the final class for a new testing instance.

Carneiro and Zhao (2017) have proposed a classification technique based on the *importance* concept, as a measure derived from PageRank (PAGE *et al.*, 1999) and embedded in the constructed network, which considers both physical and topological features of the input dataset. The technique hence eliminates the necessity of the low level term in the classification process, because it already considers both the physical features and the underlying network properties. Each data instance is represented by a node in the network, and the edges between them can be created according to both their distance, pairwise, and also to a *purity* measure, which characterizes the level of mixture of a network component in relation to other components of different classes. In the testing phase, a new data instance is inserted in the network and is classified into that class where it has the highest importance score, compared to others. The technique had

its performance evaluated firstly by applying it to both artificial and real benchmark datasets, obtaining favorable results when compared to traditional classification techniques. Afterwards, it was applied on two datasets in the detection of heart abnormalities, presenting even better results, as its was able to outperform traditional classification models on this task. A network construction method optimization has also been proposed for this technique (CARNEIRO *et al.*, 2019), based on a particle swarm optimization framework, for building the network while optimizing a quality function driven by the classification accuracy.

2.4 Graph-Formation Techniques on Supervised Learning

In this work, we make use of different network formation techniques for mapping the input dataset. Although there are currently works focused on the comparison of different graph-formation techniques on semi-supervised learning (BERTON; LOPES; VEGA-OLIVEROS, 2018; SOUSA; REZENDE; BATISTA, 2013; JEBARA; WANG; CHANG, 2009; ROHBAN; RABIEE, 2012), the literature still lacks studies concerning this subject on supervised learning tasks. For this reason, we are now going to summarize some of the concepts regarding this topic.

Virtually any dataset can be represented and analyzed in the form of a network. In order to accomplish this, for a dataset X, its data x_i are mapped as vertices and the level of similarity between each pair of vertices may generate an edge connecting them or not. Therefore, we will have $X \mapsto \mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$, where $\mathscr{V} = \{v_1, v_2, v_3 \ ..., \ v_n\}$ is the set of vertices and $\mathscr{E}$ is the set of edges. The connections can be defined by the value of a specific attribute already in the dataset, such as friendships among individuals, or flights between airports, for instance. Or they can also be created using traditional graph formation techniques, such as kNN and ε-radius.

In the first and last studies of this thesis, presented in Chapter 3 and Chapter 8, respectively, we make use of a combination of kNN and ε-radius techniques to generate the edges between each pair of vertices in the network. This scheme has been used in previous works (SILVA; ZHAO, 2012a), and the whole idea is that the ε-radius technique is used for dense regions ($|\varepsilon\text{-}radius(x_i)| > k$), while the kNN is employed for sparse regions. Mathematically, the neighbors connected to a training vertex x_i are yielded by:

$$N(x_i) = \begin{cases} \varepsilon\text{-}radius(x_i, y_{x_i}), & \text{if } |\varepsilon\text{-}radius(x_i, y_{x_i})| > k \\ k\text{NN}(x_i, y_{x_i}), & \text{otherwise} , \end{cases} \tag{2.1}$$

where y_{x_i} denotes the class label of the training instance x_i, $\varepsilon\text{-}radius(x_i, y_{x_i})$ returns the set $\{x_j, j \in \mathscr{V} : dist(x_i, x_j) < \varepsilon \wedge y_{x_i} = y_{x_j}\}$, and $k\text{NN}(x_i, y_{x_i})$ returns the set containing the k nearest vertices of the same class of x_i. One important advantage of using this combination of both techniques is that, besides being able to preserve the dense and sparse regions from the original dataset, it also prevents from leaving vertices isolated, i.e., without connections, in the resulting network.

In the second study of this work, presented in Chapter 4, we introduce a different technique for building the network, based on the idea of preserving more information from the input dataset in the network mapping procedure. Currently, the most adopted way to make this mapping is by converting each data instance to a node in the network, oftentimes by considering the Euclidean distance for generating the edges between each pair of nodes. In this way, all features have the same weight in this mapping process and, consequently, the amount of information in the resulting network is reduced when compared to the amount of information in the input dataset. This simplification becomes more significant as the number of features in the dataset increases. For this reason, in this study we opt for, instead of mapping each data instance as a node in the network, as usual, we map each data instance's attribute as being a node. Although this procedure will overall require more memory consumption from the part of the model, it has the advantage of preserving more information from the input dataset when building the network, hence allowing one to make use of this extra information for improving the learning process, during the training phase.

Another important remark to be made, still regarding graph-formation techniques on supervised learning, is with regard to the number of components in the resulting network. Given that, in the models presented in this work, the classification task is based on the impacts provoked by the insertion of each test data instance on the network's topology, for each class, then it is mandatory that each class in the dataset should be represented by only one component in the resulting network, i.e., that the number of components in the network should be equal to the number of classes in the dataset. This is necessary to allow the model to properly detect the topological pattern formation for each class, individually, during the training phase, so that the classification process can be based on these previously detected patterns, during the testing phase.

In the studies presented in Chapter 5, Chapter 6 and Chapter 7, we use different approaches for building the network, according to each problem. In the congressmen network, from Chapter 5, since the object of analysis are legislative voting data, we opt to represent each congressperson as a node in the temporal network, and to generate the edges based on their voting record similarity, pairwise. While in the COVID-19 curves temporal network, from Chapter 7, each node represents the curve from a region affected by the disease, and the edges are based on their weekly variations similarity, pairwise. By proceeding this way, we therefore have the possibility of tackling our problem by analyzing the changes occurred in these resulting temporal networks with time, in terms of their topological features. In this sense, it is also worth remembering that the network-based approach is essentially a *graphical* approach, thus it is always possible to take advantage of this convenience by plotting the resulting network at different temporal stages, to help in this analysis process. Additionally, there is also the possibility of applying different community detection algorithms in the built networks, which can also contribute in the analysis, as a handy *clustering* resource.

The graph-formation technique used in Chapter 6 stands out from the others in this work, as it is based on a time series periodicity detection network-based model (FERREIRA; ZHAO, 2014). In this technique, we start by splitting the variations from the time series of the input dataset X_{train} into a predefined number of ranges. Afterwards, each of these ranges is mapped as a node in the network, and edges are created between each pair of variation ranges if they ever appeared consecutively in X_{train}. Hence, the graph-formation technique used in this study is based on the price variations' temporal order, and the detected communities, in this case, are used for identifying possible price trend patterns in the built variations network.

CHAPTER

3

A NETWORK-BASED HIGH LEVEL DATA CLASSIFICATION TECHNIQUE

In this chapter, a high level network-based data classification technique is introduced and has its performance evaluated by using both artificial and real benchmark datasets. The technique starts by mapping the input dataset to a network, using a combination of kNN and ε-radius techniques, where each data instance becomes a node in the network. The label for each new data instance is assigned by comparing the impact provoked by the new data instance's insertion in the network and the impact patterns detected for each class during the training phase, in terms of the network's topological structure.

3.1 Introduction

The technique introduced in this chapter is inspired on a hybrid network-based data classification technique, which makes use of a low and high level of learning methods, in a combined form, with the high level method focusing, broadly speaking, on preserving the network's topological structure pattern for each class during the classification phase. In the approach proposed in this study, we focus on preserving the data instances' impact patterns on the network topological structure, for each class. Additionally, the technique we propose does not require a low level term, as in the original technique which inspired this study (SILVA; ZHAO, 2012a), relying only on the high level method for classification purposes.

The proposed model is based on the idea that data instances from the same class tend to provoke similar impacts on the network's topological structure, and that these "impact patterns", so to speak, are sufficient to correctly classify unlabeled data instances. Although this idea may, at first, sound somewhat peculiar, the tests we conducted, using both artificial and real datasets, indicate that the proposed model is able to perform relatively well, when compared with traditional and well-known classification models.

Regarding the organization of this chapter, besides this introduction, in section 3.2 we discuss the motivation for this study. In section 3.3, we explain in details how the proposed classifier is built, as well as how it calculates the impacts in the network caused by each new data item, and how this information is used later, in the testing (or classification) phase. In section 3.4, we present the obtained results from tests performed on 8 classification toy datasets and 9 real datasets, along with a comparison between the results achieved by our technique in these tests and those achieved by other traditional and well-known models, as well as by the hybrid high level technique which inspired this work. At the end, in section 3.5, we present our final remarks.

3.2 Motivation

Data classification is a common task, which can be performed by both computer and human being (animal). However, a fundamental difference between them can be observed: computer-based classification considers only the physical features, such as similarity, distance or distribution of the input data. On the other hand, brain-based classification takes into account not only physical features, but also the organizational structure of the data, such as data patterns. To be more specific, two data samples will be classified into the same class if they are constituent elements of a well defined pattern, no matter how far they may stay from each other from the perspective of their physical features. Therefore, the former is referred to as *low level* machine learning techniques while the latter *high level* ones.

Complex networks have been proven to be quite useful for characterizing relationships among data samples and, consequently, they are a powerful mechanism to capture data patterns. The pure network-based high level classification technique introduced in this work is inspired on the hybrid technique proposed by Silva and Zhao (2012a), Silva and Zhao (2015). This hybrid technique combines a low level of learning method, which can be implemented by any traditional classification technique, such as Naive Bayes, SVM, *k*NN, neural networks, etc., and a high level of learning method, which makes use of network data representation in order to identify the pattern formation, in terms of the network topology structure, for each different class in the dataset. Their results have shown that the high level term of the hybrid classifier is able to improve the performance of the low level classifier, especially in situations where the class configuration's complexity increases, measured in terms of more mixtures among different classes. That technique was built upon the idea that, in a dataset, the data relationships may present internal structures which are, oftentimes, unique and are shared among data items of the same class. It also assumes that these internal structural patterns, formed by items of the same class and identified in the training phase, will be preserved during the testing (or classification) phase.

In this sense, the technique we propose in this study makes use of a very different approach. Here, instead of focusing on the preservation of the data topological structure, we now

focus on the impact that the addition of each new item causes, at the moment it is added in the network. The term *impact*, within this context, can be understood as the changes that occurs in the network's properties, which, as we mentioned above, can be identified by extracting some specific network measures. The idea behind our technique is that items from the same class will cause similar impacts on the network, and therefore it is possible to determine the class of a new item solely by looking at the way it impacts the network. The classifier is based on only two open parameters, whose default values may be changed. We have tested its performance by applying it on 8 artificially generated datasets, as well as on 9 different real classification datasets. A comparison of these results has also been made, by applying 9 traditional and well-known classification models on the same datasets. The results from these tests are stimulating, which encourage us to continue our research to improve the efficiency of the model, as well as to apply it to a greater variety of datasets. In comparison to traditional classification techniques, the proposed technique is able to classify data according to pattern formation (high level features) instead of only considering physical features (low level features). Moreover, the proposed classification technique is purely high level, i.e., the low level term appearing in the technique presented in Silva and Zhao (2012a), Silva and Zhao (2015) is no longer necessary. Additionally, it is simpler than previous high level classification techniques as fewer parameters are required.

3.3 Materials and Methods

In this section, we describe in details how our network-based high level (NBHL) classification technique works. We start by providing an overview of its training and classification phases in subsection 3.3.1. In subsection 3.3.2 and subsection 3.3.3, we provide a more detailed description of the tasks performed by the model in the training and testing phases, respectively. In subsection 3.3.4, we describe the network measures plugged into the model.

3.3.1 Model Overview

In *supervised learning*, initially we have an input dataset, here written as $X = \{(x_1, y_1), ..., (x_n, y_n)\}$, where the first component, $x_i = (f_1, ..., f_d)$, denotes the attributes of the i-th instance of d dimensions, and the second component, $y_i \in \mathscr{L} = \{L_1, L_2, L_3..., L_n\}$, denotes the class label attributed to that same instance. The objective of the training phase is to learn the mapping $x \xrightarrow{\Delta} y$. To measure the level of learning, normally we use a second dataset for testing. In the absence of a second dataset, the initial dataset can then be split into 2 sub-datasets: X_{train}, to be used for training the classifier, and X_{test}, whose values of y, in case they exist, will be suppressed so it can be used for testing the classifier.

In our model, the training phase starts by splitting the sub-dataset X_{train} into 2 parts: X_{net} and X_{items}. The same occurs with Y_{train}, forming: Y_{net} and Y_{items}. In the first step of the training phase, an initial network is built using the data from X_{net} and Y_{net}, forming a network consisting

of sub-sets of connected components, each of which representing one unique class of the dataset. Afterwards, we extract some network measures for each of these network components, before starting the second step. In the second step of the training phase, each item in X_{items} is added to the initial network, one by one. At each new insertion, the network measures of the affected component (a sub-set representing a unique class) are recalculated. Note that, in the training phase, only one component will be affected by each insertion, which is the one that represents the item's class, taken from Y_{items}. We consider the variation on each measure of the affected component as being the item's impact values. The impact values for each new data item insertion during this phase are then stored in a 2-d array, whose dimensions are the training data instances and their respective impacts on each considered network measure, which will be our "impact list".

The impact list is used during the classification phase, when we have the impact values caused by the insertion of a new (unlabeled) item in the network and, in order to determine to which class this new data item belongs, we then check in the impact list the nearest past impact to the current impact. The new item is hence labeled with the same class from the nearest past impact, according to the information stored in the impact list. The main idea behind this model is that data items from the same class will cause similar impacts on their respective components (which represent their classes). Therefore, in this sense, it is possible to recognize one item's class only by checking the similarity between the impact it causes on the network and the past impacts, stored in each component's "memory" (so to speak).

3.3.2 Description of the Training Phase

The proposed NBHL classification technique has two parameters, k and p. The parameter k is supplied to the kNN algorithm, used for finding the neighbors of each data item in the network. The parameter p will define the proportion by which X_{train} will be split into 2 sub-datasets. The first sub-dataset, X_{net}, with length equal to $p|X_{train}|$, is used for generating the initial network. The second sub-dataset, X_{items}, with length equal to $(1-p)|X_{train}|$, will have its items added to the initial network, formed by X_{net}, one by one. Each impact value caused by the insertion of a new data instance on its respective network component is stored in a 2-d array, with a number of rows equal to the length of X_{items} and a number of columns equal to the number of network measures used for detecting the impacts, plus one column for storing the respective affected component.

Mathematically, during the first phase, X_{train} and Y_{train} have their data (x_i, y_i) mapped as vertices of the network, and an edge connecting a pair of vertices may be generated according to the level of similarity between them. Therefore we will have $X_{train} \mapsto \mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$, where $\mathscr{V}$ is the set of vertices and $\mathscr{E}$ is the set of edges. The connections are created using two traditional graph formation techniques in a combined form. The neighbors connected to a training vertex x_i

are given by:

$$N(x_i) = \begin{cases} \varepsilon\text{-}radius(x_i, y_{x_i}), \text{ if } |\varepsilon\text{-}radius(x_i, y_{x_i})| > k \\ k\text{NN}(x_i, y_{x_i}), \text{ otherwise }, \end{cases} \tag{3.1}$$

where y_{x_i} denotes the class label of the training instance x_i, ε-$radius(x_i, y_{x_i})$ returns the set $\{x_j, j \in \mathscr{V} : dist(x_i, x_j) < \varepsilon \wedge y_{x_i} = y_{x_j}\}$, and $k\text{NN}(x_i, y_{x_i})$ returns the set containing the k nearest vertices of the same class of x_i. Note that the ε-radius technique is used for dense regions ($|\varepsilon\text{-}radius(x_i)| > k$), while the kNN is employed for sparse regions. The value of ε is calculated according to the value stipulated for the parameter k, by:

$$\varepsilon = \sum_{l=1}^{n} \frac{\text{median}\{k\text{NN}_{dist}(x_i, y_{x_i})\}}{n} \quad , \tag{3.2}$$

where n is the number of classes in Y_{train}, $x_i \in X_{train}$, y_{x_i} denotes the class label of the training instance x_i, and $k\text{NN}_{dist}(x_i, y_{x_i})$ returns the distance from x_i to its k nearest neighbors with class label y_{x_i}. As a result of this technique, at the end of the training phase we will have one connected network component for each class. Reminding that the training phase is made of two different steps. In the first step the network is generated from the X_{net} and Y_{net} data. In the second step the items in X_{items} are added to the initial network, one by one, according to their respective classes, given by the data in Y_{items}, and that will also define to which component of the network the item belongs. After each insertion, the measures of the affected component are recalculated, in order to identify the impact caused by the new item in the component. Hence, at the end of the training phase we will have a list with all these impact values as well.

The impact I of an item x_i on a network component is represented by a 1-d array, whose number of columns is equal to the number of network measures plugged into the model. Its values are given by:

$$I_i^{(l)}(x)(u) = \Delta G_i^{(l)}(u)\rho^{(l)} \tag{3.3}$$

where $\Delta G_i^{(l)}(u) \in [0,1]$ is the variation of the u-th network measure that occurs on the component representing class l, if x_i belongs to this component and $\rho^{(l)} \in [0,1]$ is the proportion of data items pertaining to class l in the network. The number of measures to be plugged into the high level classifier is user-controllable. Since our classification process is highly dependent on the value of the impacts caused on these measures, then it is expected, intuitively, that the efficiency of the classifier improves as more relevant measures are used for detecting these impacts.

The last step of the training phase is to generate a list of α values, which will have the role of weighting the impact values on each component's measure during the classification phase. Therefore the α list will have the form of a 2-d array, with its columns being the α values for each measure plus one column for the component. The number of rows in the array will be equal to the number of components in the network, which, in this case, is equal to the number

of classes in Y_{train}. The final values of α, for each network component representing the class l, must satisfy the following constraint:

$$\sum_{u=1}^{M} \alpha^{(l)}(u) = 1 \quad , \tag{3.4}$$

where M is the number of network measures plugged into the model and $\alpha^{(l)}(u) \in [0,1], \forall u \in \{1,...,M\}$. After performing several tests, we chose to define the values of α as being equal to the normalized standard deviations of the impact values, for each measure and each component. So that the higher it is the standard deviation of the impact values (taken from the impact list) for a measure, the higher it will be its α value for that component. By giving larger weights for measures with higher standard deviations (and hence also with higher variations), we aim to maximize the area in the decision space formed by the points representing the past impacts for each component, at the moment of labeling a new item in the classification phase.

3.3.3 Description of the Testing Phase

In the second phase, i.e., the classification or testing phase, a new data item is inserted in the network, without any label on it. Its connections (or edges) with the existing items are defined according to the same rules described in Equation 3.1. The only difference now is that, since we do not know the class of this new item, it will then be connected to its neighbors disregarding their class labels. To determine the class for this new item, the decision is made based on the number of components affected by its insertion. In case the new item affects only one component, then the classifier will return 100% of chance of pertinence of the item to the class of this component, and the calculation of its impact is not necessary. Otherwise, when more than one component is affected by the item's insertion, the classifier then calculates the impact provoked by the item on each one of them, separately. The component with the smallest distance between the current impact and any of its past impacts will be the one to "keep" this new item for him, and thus labeling him with his own class.

It is worth noting that, in this decision process, we are not taking into account, at any moment, whether the changes provoked by the insertion of the new item on the structure of each component are big or small. Instead, we are solely estimating the level of similarity between the new item's impact and the impacts provoked by all the past insertions in each component. Hence, in this manner, we could also say that what the classifier is trying to do is to identify the "impacts pattern" on the topological structure, for each component, i.e., for each different class of the dataset.

It is also important to mention that, during the classification phase, the changes occurred in the network – as new test instances are added to its components – should not be discarded. Our tests indicated that a continuous update of the network during the classification phase not only contributes to save processing time, since, as the components' diameters grow, more test

instances should then be connected to only one of the network components along the process, but it also is able to increase the overall performance of the model, in a *self-learning* manner.

Mathematically, during the classification phase, the impact value of the new instance $x_i \in X_{test}$ on each network measure u, with respect to the class $l \in \mathscr{L}$, is given by:

$$f_i^{(l)}(x,u) = \alpha^{(l)}(u) I_i^{(l)}(x,u) \quad , \tag{3.5}$$

which will return, for each instance x_i, the 1-d array $f_i^{(l)}(x)$, with length equal to the number of network measures plugged into the model. Observe that, now, the impact values are weighted by the respective α values defined for each network measure u and for each component $C^{(l)}$, at the end of the training phase. This rule also applies to the past impact values, which are stored in the impact list. The membership of the test instance $x_i \in X_{test}$ with respect to the class l, yielded by the component $C^{(l)}$, is given by:

$$C_i^{(l)}(x) = \underset{\mu_i \in C^{(l)}}{\arg\min}\, dist\{f_i^{(l)}(x), f_i^{(l)}(\mu)\} \quad , \tag{3.6}$$

where $dist\{a,b\}$ returns the distance between the arrays a and b. In this way, each affected component will yield the smallest distance between the current impact and any of its past impacts. Finally, the probability P of x_i to belong to the class l, when more than one component is affected by the instance insertion, is given by:

$$P_i^{(l)}(x) = \frac{C_i^{(l)}(x)}{\sum_{l=1}^{n} 1 - \frac{C_i^{(l)}(x)}{\sum_{l=1}^{n} C_i^{(l)}(x)}} \quad . \tag{3.7}$$

This final equation has the role of returning the probabilities according to the normalized minimum distances yielded by the n affected components, when $n > 1$, such that the constraint

$$\sum_{l=1}^{n} P_i^{(l)}(x) = 1 \tag{3.8}$$

is satisfied, and the probabilities are inversely proportional to distances.

3.3.4 Network Measures Used for Testing

For conducting our tests, we have used 6 network measures to evaluate the changes occurred in each network component affected by the insertion of a new instance, which will also be their impact values. These measures are described below.

- *Average degree* ($\langle k \rangle$): a relatively simple measure, which statistically quantifies the average degree of the vertices of a component.
- *Assortativity* (r): numerically translates the preference for vertices of a network to attach to others that are similar or different regarding the vertices's degree in a structural sense (NEWMAN, 2003).

- *Average local clustering coefficient* ($\langle cc_i \rangle$): quantifies the degree to which local nodes in a network tend to cluster together.
- *Transitivity* (C): measures the total number of closed triangles in a network.
- *Average shortest path length* (l): gives us, as it says, the average minimum number of edges separating all nodes in a fully connected component.
- *Second moment of degree* ($\langle k^2 \rangle$): used here as an indicator of how long are the tails of the degree distribution in a graph.

3.4 Results and Discussion

In this section, we present the obtained results when applying the proposed NBHL technique to both artificial and real benchmark datasets.

3.4.1 Tests Performed on Toy Data

In this subsection, we present the results obtained when the proposed Network-Based High Level (NBHL) classification technique is applied on different classification scenarios generated from artificial data. We also present a comparison of its performance with those obtained by applying traditional classification models on the same data. Furthermore, we also apply on these same data the hybrid high level classification technique, hereafter to be referred to as HHL, from Silva and Zhao (2012a), which inspired us to conceive the NBHL technique. The comparison of the two techniques is made in order to highlight the existing differences between them.

The following traditional and well-known classification models are applied on the same artificial datasets: Decision Tree (SAFAVIN; LANDGREBE, 1991), Logistic Regression (GELMAN; HILL, 2007), Multilayer Perceptron (HINTON, 1989), Support Vector Machines (VAPNIK, 2000) and Naive Bayes (RISH, 2001). We also apply the following ensemble methods: Bagging of Decision Tree and Bagging of MLP (BREIMAN, 1996), Random Forest (BREIMAN, 2001) and AdaBoost (FREUND; SCHAPIRE, 1995). All models are implemented through the tool introduced by Pedregosa *et al.* (2011). The default RBF kernel is used for the SVM model, in all tests performed. For the other classification models, we have used their respective default parameter values.

For the HHL technique, we define its ε value, which is used in the radius nearest neighbors algorithm, according to the same rule applied to our model, described in section 3.3. Its network measures are also the same 6 measures plugged into our model, listed in subsection 3.3.4. As for its α values, we set them as being 1/6 each, and its λ values, which controls the weight of the high level term in its classification process, vary from 0.1 to 1.0. The low level term for

Table 3 – Accuracy rates (%) for each toy dataset, obtained by the following classification models, in that order: AdaBoost, Bagging of Decision Tree, Bagging of MLP, Decision Tree, Logistic Regression, MLP, Naive-Bayes, Random Forest, SVM, HHL (best result followed by its k and λ values, respectively) and NBHL (best result followed by its k and p values, respectively).

	Ada	*BagDT*	*BagMLP*	*DT*	*LR*	*MLP*	*NB*	*RF*	*SVM*	*HHL*	*NBHL*
Rnd 2 class	100	100	100	100	100	100	100	100	100	100 (1, .1)	100 (1, .1)
Rnd 3 class	96	96	96	96	100	100	100	100	100	100 (1, .1)	100 (1, .1)
Rnd 4 class	32	76	76	76	80	80	80	76	80	80 (1, .1)	80 (2, .3)
Blobs	64	68	68	68	64	52	76	76	72	64 (1, .2)	96 (1, .1)
Circles 0.0	100	100	100	100	40	56	56	96	56	100 (1, .1)	100 (1, .1)
Circles 0.25	52	60	56	60	44	52	60	56	60	48 (1, .3)	64 (1, .8)
Moons 0.0	100	96	96	96	76	72	80	96	96	100 (1, .3)	100 (1, .1)
Moons 0.25	100	96	96	96	84	88	84	92	88	96 (1, .4)	96 (1, .1)
Average Rank	4th	2nd	3rd	2nd	5th	6th	4th	2nd	3rd	3rd	1st

Source: Research data.

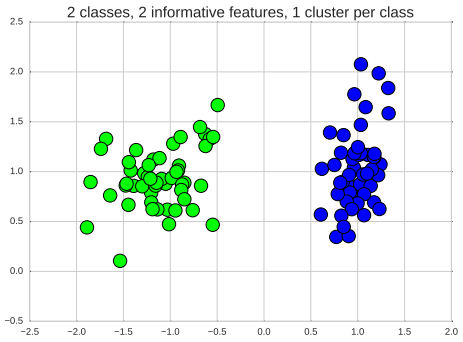
the hybrid classification is generated by the SVM model. The parameter k, for both high level techniques, varies from 1 to 4, and the parameter p, which is used only in the NBHL technique, varies from 0.1 to 0.9.

The toy data generated for the tests can be seen in Figure 5. It comprises 8 different datasets, containing 100 samples each. For the sake of facilitating the visualization, all toy datasets have two dimensions. For splitting each of them into 2 sub-datasets, for training and testing purposes, we make use of the function available in Pedregosa *et al.* (2011), which returns a train-test split with 75% and 25% the size of the inputs, respectively. As one can observe, the degree of difficulty varies among the datasets, with Figure 5a probably being the easiest one for classification purposes.

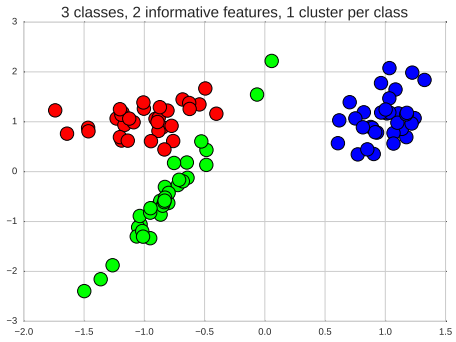
The main results from the tests are presented in Table 3. The last row in the table shows the average rank for each model, which corresponds to the average of each model's relative performance on each dataset. These numbers show that the NBHL technique performed very well in the tests, with its best results coming from Blobs and Circles with a 0.25 noise datasets, in which it was able to achieve the highest score among all models considered. Those artificial datasets have different shapes, which represent different data patterns. Although those datasets are low dimensional and visually simple ones, traditional techniques already present difficulty to cope with the variation of data patterns in those datasets. On the other hand, the high level techniques are able to capture various data patterns in the unique scheme. This is the fundamental difference between low and high level techniques.

In Figure 6 and Figure 7, we have the box plots generated from all results obtained by the HHL and NBHL techniques in the tests, considering all different combinations of values supplied for the parameters k and λ, for HHL, and for parameters k and p, for NBHL. These

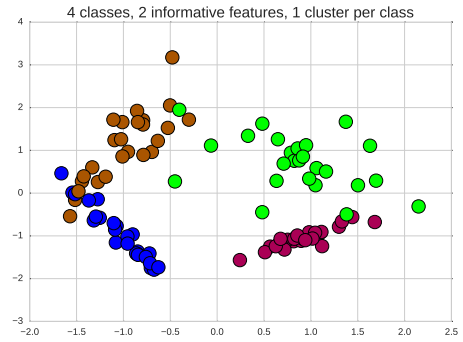
Figure 5 – Toy datasets used for performing the tests



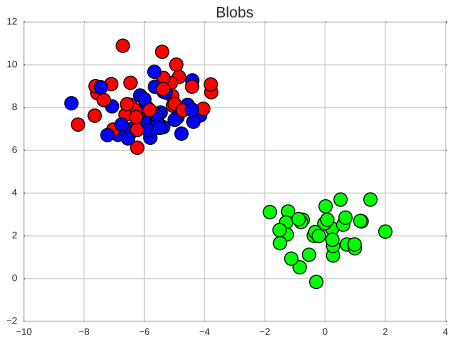
(a) Random 2 classes



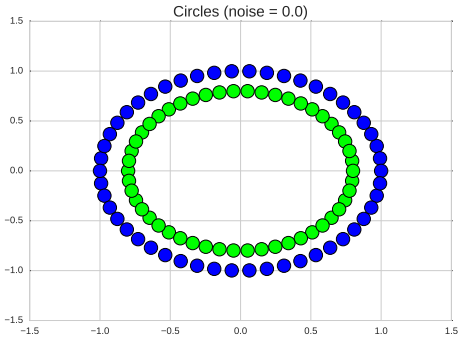
(b) Random 3 classes



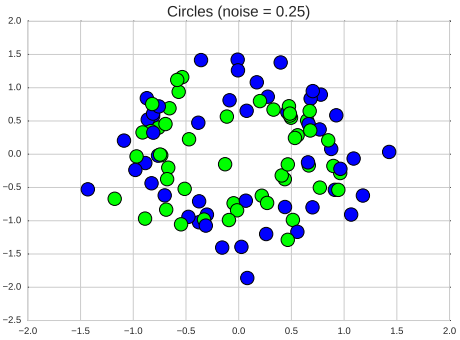
(c) Random 4 classes



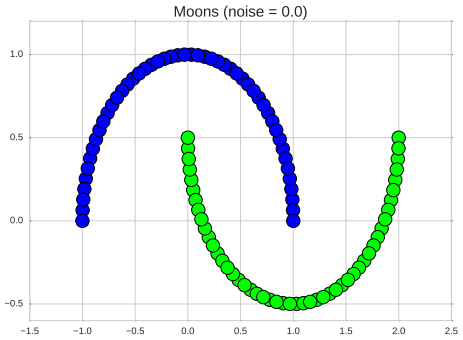
(d) Blobs 3 classes



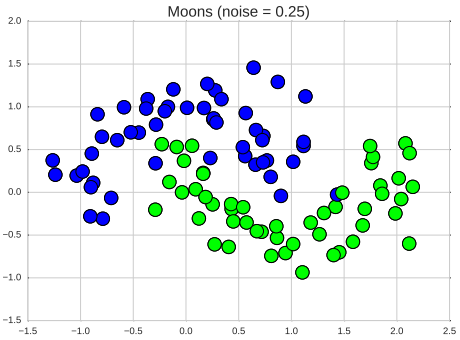
(e) Circles noise 0.0



(f) Circles noise 0.25



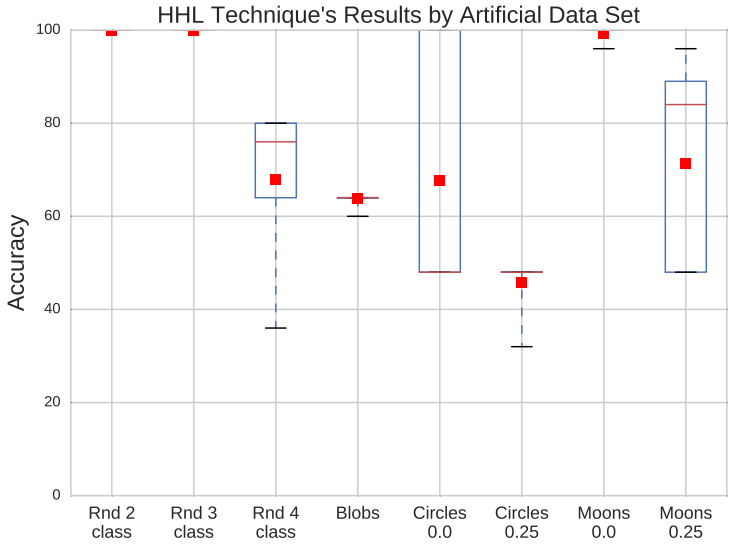
(g) Moons noise 0.0



(h) Moons noise 0.25

Source: Elaborated by the author.

Figure 6 – Overview of the Hybrid High Level - HHL technique performance on each toy dataset, with different combinations of values being supplied for its parameters k and λ. The k values vary between 1 and 4, and the parameter λ values vary between 0.1 and 1.0.



Source: Elaborated by the author.

plots show that, in these tests, the means, which are indicated by a red square in the figures, are the same for the two techniques only on the first two datasets, in which both achieved 100% accuracy in all results, i.e., they were able to classify all items correctly, for all parameters values supplied. For all other testing datasets, the mean accuracy is higher for NBHL technique.

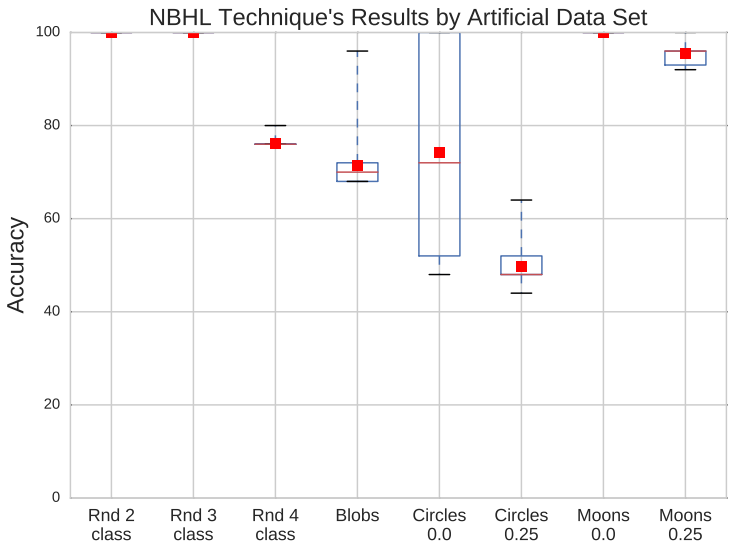
With the purpose of stressing even further the differences between both techniques, we also included, in Figure 8, the box plots generated by the HHL's results filtered by a λ value equal to 1.0, which brings us the scores achieved by this technique when only its high level term is considered in the classification process. Having in mind that the HHL technique – which is, as its own name says, a *hybrid* technique – was not originally conceived for this type of analysis, we are adding this information in the results strictly for differentiation purposes.

3.4.2 Tests Performed on Real Data

In this subsection, we present the results obtained by applying the proposed NBHL technique, along with other traditional and well-known classification models, on UCI real classification datasets. A succinct meta-information of the selected datasets is given in Table 4. For a detailed description, one can refer to Lichman (2013). For splitting each UCI dataset into 2 sub-datasets, for training and testing purposes, we make use of the same function used in subsection 3.4.1, available in Pedregosa *et al.* (2011), which returns a train-test split with 75% and 25% the size of the inputs, respectively.

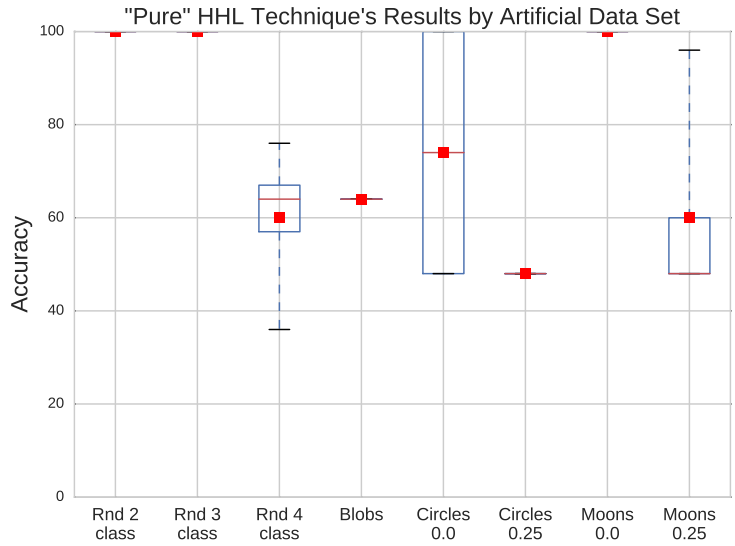
The traditional models applied on the datasets in this section are also the same ones

Figure 7 – Overview of the proposed Network-Based High Level - NBHL technique performance on each toy dataset, with different combinations of values being supplied for its parameters k and p. The k values vary between 1 and 4, and the parameter p values vary between 0.1 and 0.9.



Source: Elaborated by the author.

Figure 8 – Overview of the "pure" Hybrid High Level - HHL technique performance on each toy dataset, with the values for its parameter k varying between 1 and 4, and its parameter λ fixed at 1.0.



Source: Elaborated by the author.

Table 4 – Meta information of the real classification datasets used for testing and for comparing the results obtained by the NBHL technique.

	N^o of Samples	**N^o of Dimensions**	**N^o of Classes**
Breast Cancer	569	30	2
Digits	1,767	64	9
Glass	214	10	6
Iris	150	4	3
Pima	768	8	2
Teaching	151	5	3
Wine	178	13	3
Yeast	1,484	8	10
Zoo	101	17	7

Source: Research data.

Table 5 – Parameters supplied for the SVM (γ) and NBHL (k and p) classifiers for obtaining the results presented in Table 6, for each dataset.

	SVM	NBHL	
	γ	k	p
Breast Cancer	0.001	2	0.1
Digits	0.001	1	0.5
Glass	1.0	2	0.8
Iris	1.0	1	0.5
Pima	1.0	3	0.1
Teaching	1.0	1	0.2
Wine	0.001	6	0.9
Yeast	1.0	3	0.9
Zoo	0.1	1	0.5

Source: Research data.

described and used in subsection 3.4.1. The parameters supplied for the SVM (γ) and NBHL (k and p) classifiers are summarized in Table 5.

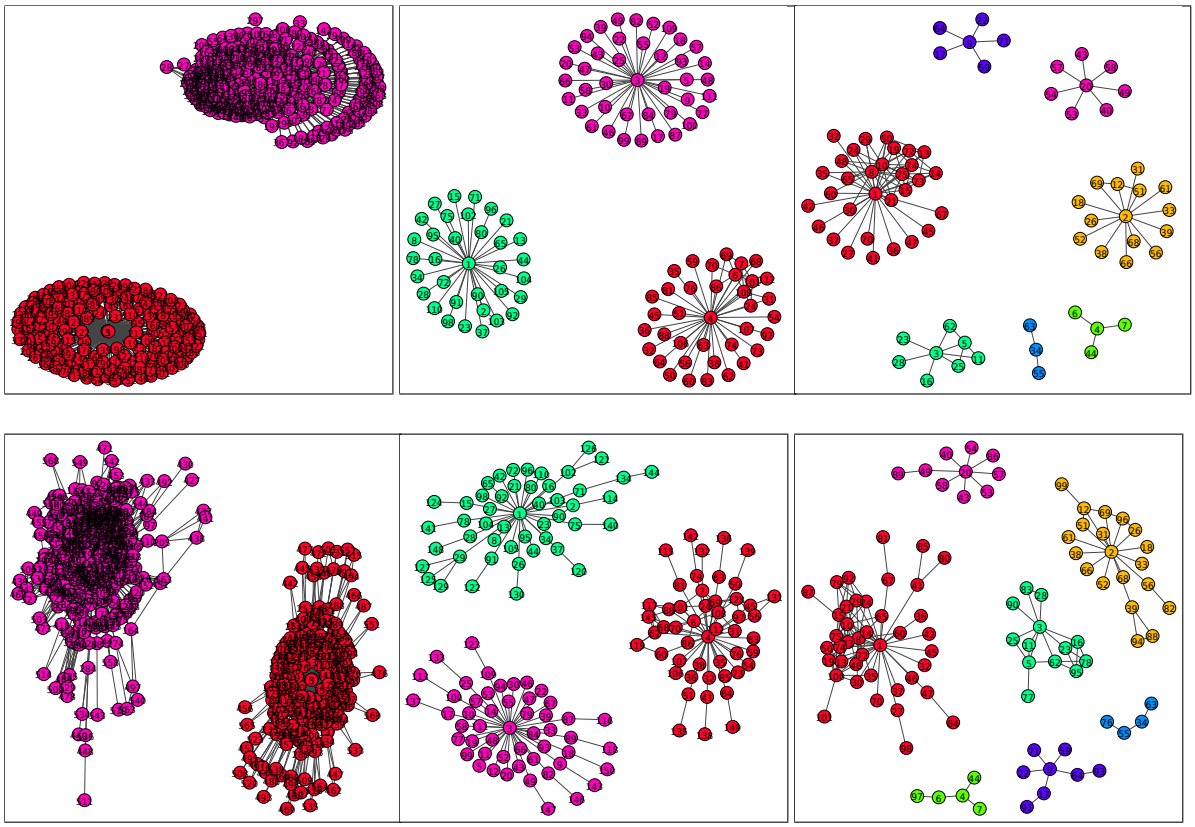
The main results from these tests are presented in Table 6. The average rank here is the same used in Table 3, corresponding to the average relative performance of each model, on each dataset. As it is shown in Table 6, the NBHL is ranked in the third place, which means that it performed well when compared with other traditional and well-known classification techniques applied on the same datasets. It is possible to highlight, for instance, that it had the second best performance on the Digits dataset, with a 99.1% accuracy rate, right behind the SVM model. It was also able to achieve good scores on the Iris and Zoo datasets, with 97.4% and 100% of accuracy, respectively. However, it presented a relatively low performance on the Yeast and Wine datasets. We see this relatively low score on these specific datasets as a sign that, even though its preliminary results can be considered stimulating, there is still a lot of open research possibilities to improve the model's efficiency.

Table 6 – Accuracy rates (%) for each real dataset, obtained by the following techniques, in that order: AdaBoost, Bagging of Decision Tree, Bagging of MLP, Decision Tree, Logistic Regression, MLP, Naive-Bayes, Random Forest, SVM and NBHL. The parameters supplied for the SVM and NBHL techniques are displayed in the Table 5.

	Ada	*BagDT*	*BagMLP*	*DT*	*LR*	*MLP*	*NB*	*RF*	*SVM*	*NBHL*
Breast Cancer	97.8	86.0	91.6	89.5	95.8	89.5	93.7	93.7	93.0	94.4
Digits	26.2	84.9	84.4	85.5	95.3	96.7	83.3	94.0	99.5	99.1
Glass	46.3	63.9	64.8	57.4	57.4	67.3	46.3	70.4	64.8	66.7
Iris	89.5	97.4	97.4	97.4	86.8	92.1	100.0	97.4	97.4	97.4
Pima	79.2	71.3	75.0	72.9	80.7	70.3	76.6	74.0	67.7	73.4
Teaching	47.4	57.9	57.9	55.3	44.7	60.9	44.7	68.4	55.3	55.3
Wine	86.7	84.4	91.1	84.4	93.3	97.8	93.3	95.5	71.1	80.0
Yeast	45.0	50.9	51.2	53.1	53.1	57.7	11.3	57.9	57.9	36.7
Zoo	76.9	100.0	100.0	100.0	100.0	96.1	100.0	100.0	100.0	100.0
Average Rank	7th	8th	4th	6th	2nd	2nd	5th	1st	4th	3rd

Source: Research data.

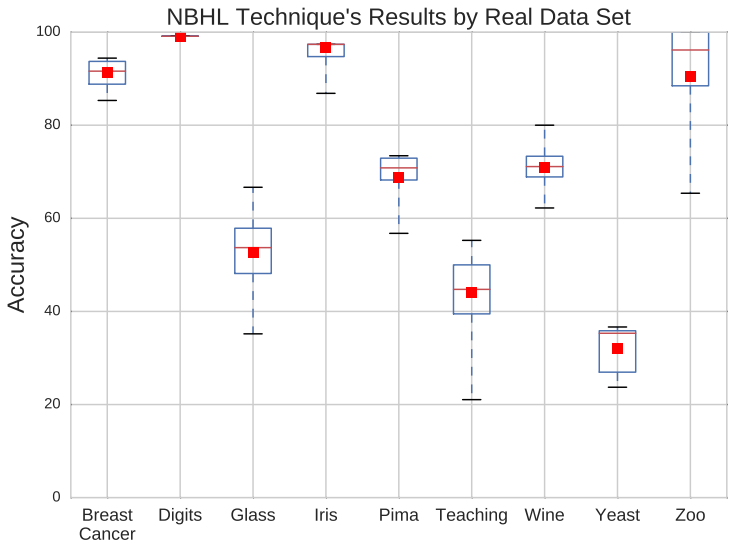
Figure 9 – Networks after the training (above) and after the testing (below) phases, for the (a) Breast Cancer, (b) Iris and (c) Zoo datasets.



(a) Breast Cancer (b) Iris (c) Zoo

Source: Elaborated by the author.

Figure 10 – Overview of the NBHL classifier performance on each dataset, with different combinations of values being supplied for its parameters k and p. The k values vary between 1 and 6, and the parameter p values vary between 0.2 and 0.9.



Source: Elaborated by the author.

In Figure 9, it is possible to see the networks generated by the proposed model after the training phase and after the classification phase, for the Breast Cancer, Iris and Zoo datasets. It is interesting to note, in this figure, the main changes occurred in the topological structure of each network, during the classification phase, for the three datasets. It is also worth noting, especially in Figure 9c, that even when we have a relatively small dataset, the classifier is still able to correctly detect the particularities of the impact patterns, for each class, and thus to properly distribute the testing items among all components identified by him during the training phase.

In Figure 10, there are the box plots of all results obtained during the tests performed with the NBHL, using different combinations of values for the two parameters k and p. The k values varied between 1 and 6, while the parameter p values varied between 0.2 and 0.9. As we can see, when different combinations of values for these two parameters are supplied for the model, the final results may vary. However, these variations do not seem to affect its overall performance considerably, for most of the datasets. Therefore we believe this figure helps to demonstrate the overall robustness of the model, when it comes to the values of its parameters k and p.

3.5 Chapter Remarks

In this chapter, we have introduced a novel pure high level classification technique, which is built upon the idea that the impacts caused by the insertion of new items on the internal structures shared among data items of the same class tend to be alike, leading to the emergence

of some sort of "impacts pattern" for each class, and that this peculiar trend could be sufficient for a classifier to infer from which class a new unlabeled instance belongs to. Several tests were performed, on 8 artificially generated datasets and on other 9 different real classification datasets, using distinct combinations of values for the two parameters k and p. A performance comparison has also been made, by applying 9 traditional and well-known classification models on these same datasets. The results obtained through these tests can be considered as quite encouraging.

We believe, given the intrinsic evolving nature of the rationale behind our model, that the method presented here may be especially indicated for the classification of datasets with temporal attributes. In which their topological structure, rather than being static, may change with time. It is expected that, in this kind of situations, our classification technique has the potential of performing well when compared with other classification methods, which have a more stable approach.

CHAPTER

4

A MODULARITY-BASED HIGH LEVEL DATA CLASSIFICATION TECHNIQUE

In this chapter, another high level network-based data classification technique is introduced and has its performance evaluated by using both artificial and real benchmark datasets. This technique starts by mapping the input dataset to a network, with the difference that now each data instance's attribute becomes a node in the network, and only the ε-radius technique is considered when building the network. Additionally, the label for each new data instance is assigned by assessing the impact caused by its insertion in the network, in terms of the modularity score.

4.1 Introduction

Many networks encountered in the real world, such as social, computer and metabolic networks, are found to divide naturally into communities or modules (NEWMAN, 2006). One of the most effective ways of detecting these communities is through the modularity measure. The well-known fast greedy algorithm, for instance, determines the optimal number of communities in the network by maximizing the modularity score of the graph. In this chapter, we investigate whether the use of this measure alone would be sufficient for building a data classification technique. The obtained results, by testing the proposed model with both artificial and real benchmark datasets, indicate that it is indeed possible to achieve competitive performances, by only considering this measure in the classification process.

The main novelty of the network-based classification model presented in this study is that, instead of mapping each data instance as a node in the network, as usual, it maps each data instance's attribute as a node in the network. This modification in the mapping procedure aims to preserve more information from the dataset, in such a way that this information can be used for improving the model's performance in the classification process. Additionally, the proposed

technique considers only the modularity measure for calculating and comparing the impacts provoked by the insertion of each new data instance in the network. This simplification is possible due to the form that the network is built during the training phase, by using an algorithm which minimizes the radius distance parameter for creating the edges while still outputting a network with one component for each class in the dataset. Moreover, an *attention mechanism* is used in this technique, similar to the one used in the Transformer deep learning model (VASWANI *et al.*, 2017), to discriminate which edges generated during the testing phase should be considered more important in the classification process.

Besides this introduction, this chapter is organized as follows. In section 4.2, we discuss the motivation for conceiving the model. In section 4.3, we start with an overview of the proposed model, then we provide a detailed description of the model's training phase, showing how it maps the input dataset to an attributes network, and also describe the model's testing phase, explaining how the probabilities for a new testing instance of belonging to each class are calculated. In section 4.4, we present the obtained results, along with some discussions, from the application to benchmark datasets and compare the performance with those obtained by other traditional classification models. At the end, in section 4.5, we conclude the chapter by adding some final remarks.

4.2 Motivation

One of the key aspects when employing a graphical approach to a classification problem lies in how the network is built from the information provided by the input dataset (BERTON; LOPES; VEGA-OLIVEROS, 2018). Usually, for this process, each data instance is mapped as a node in the network, and the edges among them are generated according to the distance they are from each other, pairwise, based on a threshold parameter. This threshold can be set either as a fixed value, thus assuming the form of a radius in the dimension space, for instance, or it can also be based on a kNN algorithm, where we hence have that a node will be connected to its k nearest neighbors in the network, disregarding how far they are from each other. However, although the procedure of mapping every data instance as a node in the network is still the most adopted one, it has the drawback of, oftentimes, discarding useful information from the dataset during the mapping process. Specially when the number of features in the dataset is larger, which turns more difficult to transfer all this information to the network without any loss, since some simplifications are required for the mapping process.

In this study, we introduce a different technique for building the network, where, instead of mapping each instance of the dataset as a node, as usual, we map each of the instance's attributes as being a node. Although this procedure will overall require more memory consumption from the part of the model, it has the important advantage of allowing to preserve more information from the dataset when building the network, and hence to also make use of this extra

information for improving its learning process, during the training phase. In addition, we also introduce a technique for building the network which aims to minimize the value of the threshold parameter responsible for determining the edges among the nodes, while still yielding a network with only one component per class. This technique is used in our model for the sake of making the network more sensitive to the addition of new instances, in terms of its modularity measure, during the testing phase.

The proposed model is evaluated by applying it to benchmark artificial and real classification datasets, as well as by comparing its performance with the ones achieved by other traditional classification models on the same data. The results obtained are encouraging, with the model being ranked on second place among the 10 classifiers considered, both for the artificial and real selected datasets.

4.3 Materials and Methods

In this section, we start, in subsection 4.3.1, by providing an overview of the proposed modularity-based high level classification model (MBHL). In subsection 4.3.2, we have a detailed description of the model's training phase, explaining how it builds the attributes network and how it calibrates the threshold parameter. In subsection 4.3.3, there is the description of the testing phase, explaining how the new instances are classified. At the end of the section, in subsection 4.3.4, we show the datasets used for evaluating the model's performance.

4.3.1 Model Overview

In *supervised learning*, initially we have an input dataset comprising n instances and m features, in the form of $X = \{x_1, x_2, ..., x_n\}$, where each instance i consists of m dimensions, such that $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,m})$, as in the following 2d array:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,m} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,m} \end{bmatrix} . \tag{4.1}$$

The labels for each instance i are usually provided in the form of $Y = \{y_1, y_2, y_3 \dots, y_m\}$, such that $y_i \in \mathscr{L} = \{L_1, L_2, L_3 \dots, L_n\}$, which represent the n classes in the dataset. The objective of the *training phase* is to explore the covariations and to identify possible patterns in the dataset, in order to learn the mapping $X \xrightarrow{\Delta} Y$. To measure the level of learning, in the *testing phase*, normally we make use of a second dataset. In the absence of a second dataset, then the initial dataset can be split into 2 subdatasets: X_{train} and Y_{train}, to be used for training the classifier, and X_{test} and Y_{test}, whose values of Y are suppressed so they can be used for evaluating the classifier's performance.

A network can be defined as graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is a set of nodes and $\mathscr{E}$ is a set of tuples representing the edges between each pair of nodes $(i, j) : i, j \in \mathscr{V}$. In the proposed model, each node in the network represents an attribute d of an instance x_i in the dataset, such that the size of $\mathscr{G}$ is always equal to $n \cdot m$. The connections (or edges) between nodes from different instances are created based on how far they are from each other, in the spacial dimension formed by each attribute d, according to a threshold parameter ε.

The main novelty introduced by the proposed model lies in its network formation method, which involves a calibration process to generate a network with one component for each class in the dataset, while giving the minimum possible value for the threshold parameter ε. In this way, the network resulted from the training phase will be more sensitive, in terms of its modularity measure, to the insertion of new instances during the testing phase, which, by its turn, is expected to contribute for increasing the performance of a classification method which is solely based on this measure, as it is the case of the rationale behind the model proposed in this study. In the training phase, the attributes network is built by using the minimum possible values for the parameter ε, adjusted to each dataset. In the testing phase, new instances are then inserted in the network and their labels will be yielded basically by the class which results in the higher positive impact on the network's modularity measure, weighted by two other parameters, γ and ρ. Following, we provide more details about the training and testing phases of the model, along with examples of application of its training phase on four benchmark classification datasets.

4.3.2 Description of the Training Phase

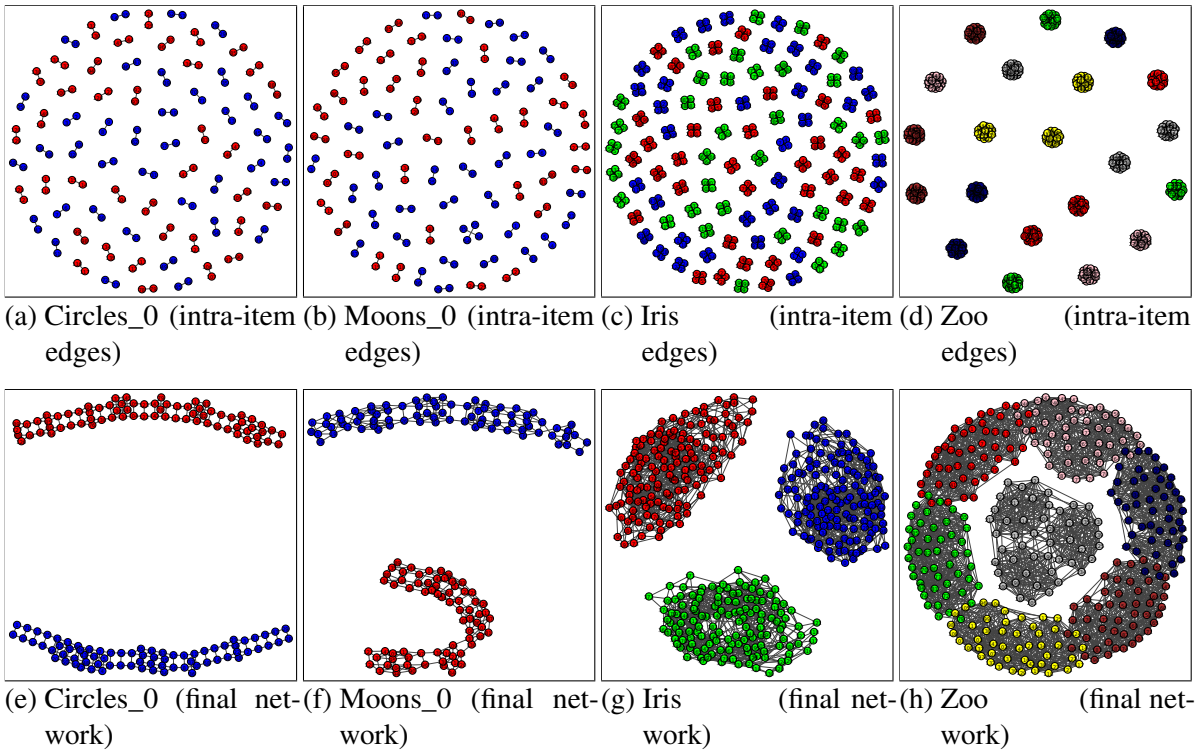
The training phase starts by generating the initial values of the parameter ε, used for creating the edges in the network. Afterwards, it generates the attributes network $\mathscr{G}$, while it calibrates the values of ε in order to keep them at a minimum necessary value to obtain one component for each class in the dataset. For each class L, the parameter ε_d^L yields the maximum difference between the values of attributes $x_{i,d}$ and $x_{j,d}$, with $i \neq j$, in order to connect their respective nodes by a link in the network. Its initial values are given by:

$$\varepsilon_d^L = \theta \sqrt{\frac{\sum_{i=1}^{n} (x_{i,d} - \bar{X}_d)^2}{n-1}} \quad , \tag{4.2}$$

which yields the standard deviation of the sample comprising all n values for the attribute d in X_{train} multiplied by a predefined value θ, such that $\theta \in [0, 1]$. Note that those are the initial values for ε, which oftentimes will change later, during the calibration of the parameter ε when generating the attributes network. Therefore, as lower the value of θ, the lower will be the initial values of ε and more changes are expected to occur for its values during the calibration process.

After generating the initial values for ε, the next step in the training phase is to create the network from the training dataset, in which every attribute will become a node in the network and the edges between them are defined by the parameter ε. We start by generating a new graph

Figure 11 – Examples of networks generated by following the steps in the training phase for four datasets. (Above) network with only edges between attributes of a same instance, and (Below) final attributes network for: Circles_0 dataset (2 classes and 2 features), Moons_0 dataset (2 classes and 2 features), Iris dataset (3 classes and 4 features), and Zoo dataset (7 classes and 16 features). The number of training instances is reduced for the Zoo dataset only for the sake of visibility.



(a) Circles_0 (intra-item edges) (b) Moons_0 (intra-item edges) (c) Iris (intra-item edges) (d) Zoo (intra-item edges)

(e) Circles_0 (final network) (f) Moons_0 (final network) (g) Iris (final network) (h) Zoo (final network)

Source: Elaborated by the author.

$\mathscr{G}$, with $n \cdot m$ vertices, where n is the number of instances and m is the number of dimensions in X_{train}. Then, we create the "intra-item" edges, by taking all nodes representing an attribute $x_{i,d}$, from the same instance i, and connecting them, pairwise, such that any attribute x_{i,d_1} will be connected to the attribute x_{i,d_2}, except when $d_1 = d_2$, to avoid self-loops. At this point, we will already have a network as it is shown in Figure 11a, Figure 11b, Figure 11c and Figure 11d, with each node representing an attribute and still without any edges connecting nodes from different instances. Afterwards, the model starts to calibrate the parameter ε by, at each iteration, setting its values as the minimum distance between the components from a same class in the network and connecting then by new edges, correspondingly, until we have only one component per class, as it is shown in Figure 11e, Figure 11f, Figure 11g and Figure 11h.

The complete process for generating the network is outlined in Algorithm 1. In lines (2:4), we create the network, add the vertices and the intra-item edges. In lines (5:27), there is the ε calibration process, when the "inter-edges", i.e., the edges connecting attributes from different instances, are created, by determining the minimum values for ε such that, at the end, we have one component for each class in the network.

Algorithm 1 – Attributes network G generation

```
1:  procedure GENERATE G(X,Y)
2:      G ← new graph with n × m vertices (number of instances × number of features)
3:      G ← add intra-item edges
4:      es_list ← [ ]
5:      while len(G^components) > len(ℒ) do
6:          for class in set(Y) do
7:              es ← GetInterEdges(X, ε, class)
8:              es ← e in es if e not in es_list
9:              G ← add edges es
10:             es_list ← add es
11:         end for
12:         comps ← components in G
13:         classes ← [L if len(comps^(class=L)) > 1]
14:         C ← largest component in comps^class in classes
15:         others ← [other components in comps^(class=C^class)]
16:         difmin ← [∞] · m
17:         for node1 in C do
18:             i1 ← X values for node1
19:             for node2 in others do
20:                 i2 ← X values for node2
21:                 dif ← [0] · m
22:                 for d = 0 to m do
23:                     dif_d = |i1_d − i2_d| − ε_d^(C^class)
24:                 end for
25:                 if max(dif) < max(difmin) then
26:                     difmin = dif
27:                 end if
28:             end for
29:         end for
30:         for d = 0 to m do
31:             if difmin_d > 0 then
32:                 ε_d^(C^class) += difmin_d
33:             end if
34:         end for
35:     end while
36: return G
37: end procedure
```

The Algorithm 2 shows how the edges for connecting nodes from different instances in the network are created, according to the current values of the parameter ε. In this case, the model will generate edges between each pair of nodes $x_{i,d}$ and $x_{j,d}$ representing a same attribute d and a same class L, where $i \neq j$, whenever the distance between them are within the range ε_d^L. In this way, the total number of edges between two instances will range from 0 until the number of dimensions m in the dataset.

The last step in the training phase consists of generating the final values of the parameter ε, to be used in the testing phase. These values are given by the arithmetic mean of the current values of ε at each iteration during the calibration process. Hence, for testing purposes, the values of ε are yielded by:

$$\varepsilon_d^L = \frac{\sum_{i=1}^{n} \varepsilon_{d\,i}^L}{n} \quad , \tag{4.3}$$

where n is the number of changes occurred for ε_d^L during its calibration process, d is one of the dimensions in the dataset and $L \in \mathscr{L}$. In this way, for testing purposes, the values of ε will be somewhere between its initial values, when generally only a few edges are generated or even none of them, and its final values in the calibration process.

Algorithm 2 – Inter-items edges generation

```
1:  procedure GETINTEREDGES(X, ε, L)
2:      es ← [ ]
3:      combs ← combinations of indexes in X^L, pairwise
4:      for i in indexes of X^L do
5:          pairs ← combs^{pair_0 = i}
6:          for p in pairs do
7:              for d = 0 to m do
8:                  dist = |X^L_{p_0,d} − X^L_{p_1,d}|
9:                  if dist ≤ ε^L_d then
10:                     es ← add pair p
11:                 end if
12:             end for
13:         end for
14:     end for
15: return es
16: end procedure
```

4.3.3 Description of the Testing Phase

In the testing phase, a new instance is inserted in the network $\mathscr{G}$ and the model then needs to assign it a label, according to its attributes values. To accomplish this task, the model starts by simulating its insertion in each of the network's components, generating edges according to the values of the parameter ε, and then calculates the impacts of this insertion on the network's *modularity* measure Q, for each component, i.e., for each class. The probabilities for the new instance to belong to each class L are yielded based on the number of edges generated for each attribute d, weighted by its respective parameter γ_d, and on the impacts on the modularity Q, weighted by its respective parameter ρ^L, such that the higher the positive impacts on Q when the instance is inserted in the component from class L, the higher the probability of the instance to belong to this class.

The modularity measure, broadly speaking, compares the number of connections between vertices which share a same characteristic with the expected number of connections when occurred randomly, and it is often used for detecting communities in a network. The *fast greedy* algorithm (CLAUSET; NEWMAN; MOORE, 2004), for instance, determines the optimal number of communities in the network by maximizing the modularity score of the graph. For the classification task proposed in this work, we take into account two other factors regarding this measure, which are: (1) How meaningful is each of these new connections (or edges) induced by the insertion of the new instance in the network, and (2) The ratio between the respective number of new connections generated and the size of the network component, since, overall, larger components tend to receive more links from the new instance, which would incur in biased estimations from the classifier. Therefore, for the testing phase, we adopt two parameters, γ and ρ, for managing these two mentioned factors in the model.

The parameter γ has the role of yielding the correlations between each attribute in X_{train} and the classes in Y_{train}. These values are used for weighting the number of edges generated between a new instance and the already existing nodes in the network, such that attributes which are more correlated to Y_{train} have higher weights on the final probability scores yielded for

each class. In order to determine these weights, we opt for making use of the Ridge Regression (HOERL; KENNARD, 1970), with the values of γ hence assuming the values of the coefficients w returned by this linear regression model. Ridge Regression regularizes an Ordinary Least Squares model by favoring simpler models, i.e., with smaller coefficients, by minimizing the following loss function:

$$|Y - Xw|^2 + \lambda |w|^2 \quad , \tag{4.4}$$

where λ is a term to control the regularization strength. The main distinction of Ridge Regression among other linear regression models is that it enforces the coefficients w to be lower by introducing a constraint as the second term in Equation 4.4 to penalize large values for w. So the lower the regularization term λ is, the more the model will resemble an Ordinary Least Squares model. In our adaptation, the values of γ are yielded by:

$$\gamma_d = \frac{\sqrt{|w_d|}}{\sum_{d=1}^{m} \sqrt{|w_d|}} \quad , \tag{4.5}$$

where $|w_d|$ is the absolute value of the coefficient returned by the Ridge Regression model for attribute d and m is the number of dimensions in the dataset. The root square in Equation 4.5 is inserted for the sake of balancing the values of γ in cases when the differences between them become too large.

Please note that the strategy resulting from Equation 4.5, of adopting normalized weights for each attribute according to their respective level of importance in the dataset, is a type of *attention mechanism*, often used in *deep learning* models such as the Transformer (VASWANI *et al.*, 2017), in which the final weights are usually obtained through a SoftMax function. In this work, we adopt this procedure to identify which edges should be considered more "meaningful" when a new data instance is inserted in the network to be classified, in the testing phase.

The third parameter of the model ρ is responsible for balancing the impacts on the network's modularity measure, when inserting new instances during the testing phase, according to the ratio of the number of instances per class in the training dataset. This is necessary for dealing with cases of unbalanced datasets. Its values are given by:

$$\rho_L = \frac{|X_{train}|}{|X_{train}^L|} \quad , \tag{4.6}$$

where $L \in \mathscr{L}$ and $|X_{train}^L|$ stands for the length of all instances in X_{train} from class L. The final values of ρ must also be normalized, hence assuming the form $\rho_L / \sum_L \rho_L$.

Likewise in the training phase, the new instance has its attributes mapped as m nodes in the network, where m is the number of dimensions in the dataset, and the edges among its nodes (intra-item edges) are created according to the same rule used in the training phase, with all its pairs of attributes x_{i,d_1} and x_{i,d_2} being connected, pairwise, as long as $d_1 \neq d_2$, to avoid self-loops. The generation of edge $(x_{i,d}, x_{j,d})$ between the node representing attribute d of the

new instance i and any already existing node in the network j, also representing attribute d, from class L, is yielded by:

$$(x_{i,d}, x_{j,d}) = \begin{cases} 1, \text{ if } |x_{i,d} - x_{j,d}| \leq \varepsilon_d^L \\ 0, \text{ otherwise }, \end{cases} \tag{4.7}$$

where $L \in \mathscr{L}$. The total number of edges created are then averaged by each attribute d and weighted by the respective parameter γ, providing us with the indicator E, for each class L, according to:

$$E^L = \gamma_d \frac{\sum_{d=1}^{m} N_d}{n} \quad , \tag{4.8}$$

where N_d represents the total number of edges generated for attribute d between the new instance's nodes and the other nodes in the network and m is the number of dimensions in the dataset. These values are later normalized, by:

$$E^L = \frac{E^L}{\sum_L E^L} \quad , \wedge L \in \mathscr{L} \quad . \tag{4.9}$$

Next, the overall impact I^L of the new instance's insertion on the network's modularity measure Q, for each class L, is calculated and has its value weighted both by E and by the parameter ρ, according to:

$$I^L = \rho^L E^L \frac{(Q^L - Q_0)}{Q_0} \quad , \text{ and} \tag{4.10}$$

with Q_0 being the value of the network's modularity measure Q at the end of the training phase. These values are also later normalized, providing us with what will be the probabilities $P(i^{class=L})$ of the new instance i to belong to each class L, in the form of:

$$P(i^{class=L}) = \frac{I^L}{\sum_L I^L} \quad . \tag{4.11}$$

At the end, the final label L to be assigned to the new instance i will be the one among $L \in \mathscr{L}$ which maximizes $P(i^{class=L})$, being yielded by:

$$i^L = \mathscr{L}_{\arg\max_L P(i^{class=L})} \quad . \tag{4.12}$$

Therefore, the new instance i will belong to the class L which results in the highest positive impact on the network's modularity measure Q, when weighted by the balancing parameter ρ and also by the indicator E, which, by its turn, measures the level of "meaningfulness" of its connections in the network, so to speak, by weighting the number of edges generated per attribute by the respective correlation γ between each feature and the labels in the dataset.

4.3.4 Database

We evaluate the efficiency of the proposed modularity-based high level model (MBHL) by applying it to artificially generated data and also to well-known benchmark datasets intended for machine learning classification tests. A succinct meta-information of the selected datasets

used for testing purposes is given in Table 7. For a detailed description of the real datasets, one can refer to Lichman (2013). Examples of the artificial datasets generated for the tests are provided in Figure 12. For splitting each dataset into 2 subdatasets, for training and testing purposes, we make use of a function which shuffles the data, through a random seed value, and returns a train-test split with 75% and 25% the size of the inputs, respectively. As preprocessing, all real datasets have their features treated through a quantile transformation, such that their values are adjusted to follow a uniform distribution, ranging from 0 to 1.

Table 7 – Meta information of the classification datasets used for evaluating and comparing the MBHL model

		N^o of Samples	**N^o of Features**	**N^o of Classes**
Artificial	Circles_0	100	2	2
	Circles_01	100	2	2
	Moons_0	100	2	2
	Moons_01	100	2	2
Real	Bankruptcy	250	6	2
	Haberman	306	3	2
	Hayes-roth	132	5	3
	Iris	150	4	3
	Wine	178	13	3
	Zoo	101	16	7

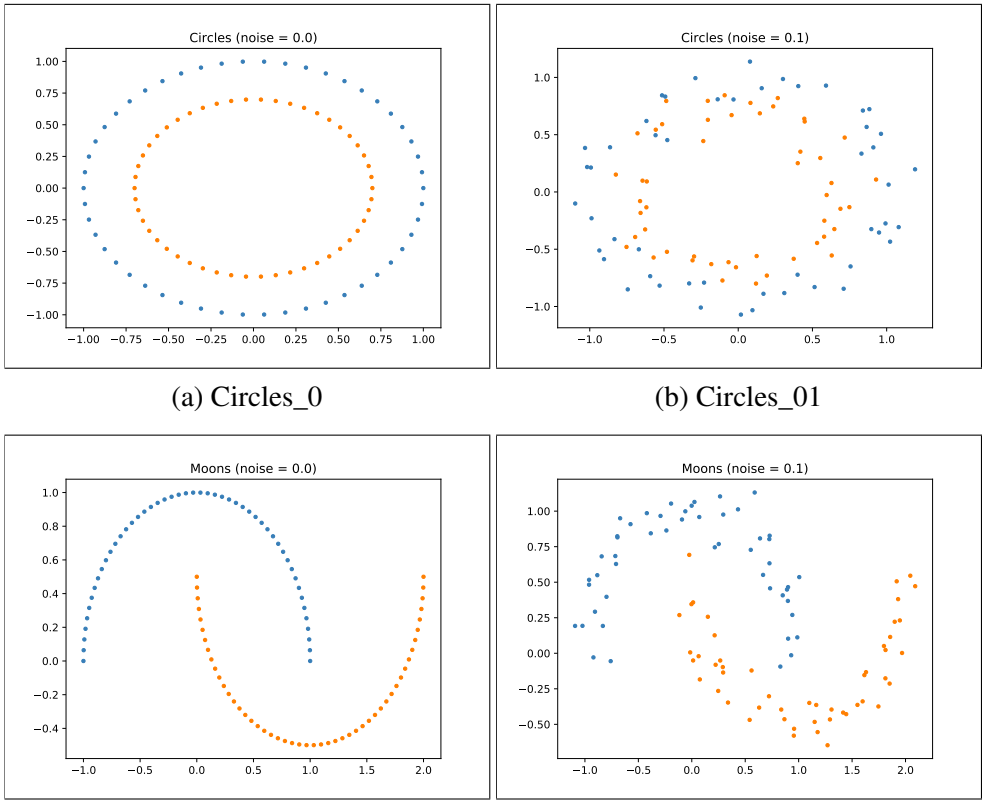
Source: Research data.

For the sake of comparison, the following traditional classification models are applied on the same datasets listed in Table 7: Decision Tree (SAFAVIN; LANDGREBE, 1991), Logistic Regression (GELMAN; HILL, 2007), Multilayer Perceptron (HINTON, 1989), Support Vector Machines with an RBF kernel (VAPNIK, 2000) and Naive Bayes (RISH, 2001). We also apply the following ensemble methods: Bagging of Decision Tree and Bagging of MLP (BREIMAN, 1996), Random Forest (BREIMAN, 2001) and AdaBoost (FREUND; SCHAPIRE, 1995). All traditional models are implemented through (PEDREGOSA *et al.*, 2011) and we keep their respective default parameters values, in all tests performed. As for the proposed MBHL model, we set the parameter $\theta = 0.1$ for tests with artificial datasets and $\theta = 0.5$ for tests with real datasets. The value of λ, for the Ridge Regression, was set to 1.0. For the tests with artificial datasets, the model generates edges among nodes from different instances only when all distances between them are within the respective range yielded by ε_d. Each dataset is processed 50 times by all models, each time having their data items shuffled by using a different randomly generated seed. The final accuracy scores are the averaged ones achieved by each model, on each dataset.

4.4 Results and Discussion

In this section, we present the obtained results when applying the proposed MBHL model to artificial and real benchmark classification datasets, along with a comparison of its

Figure 12 – Artificial datasets generated for the evaluation and comparison of the model. Above: (a) two concentric circles without noise and (b) two concentric circles with a noise of 0.1. Below: (c) two moons without noise and (d) two moons with a noise of 0.1.



(a) Circles_0 (b) Circles_01

(c) Moons_0 (d) Moons_1

Source: Elaborated by the author.

performance with the ones achieved by traditional classification models on the same data.

The results obtained from the application of the proposed MBHL model, along with other traditional classification models, both on real and artificial datasets, are summarized in Table 8. The Average Rank, in the last row, indicates the averaged rank position achieved by each model considering all datasets, according to their respective rank achieved on each of them, in terms of mean accuracy values.

Regarding the results obtained on artificial datasets, although the MBHL model was not ranked so well for the Circles_01 dataset (two concentric circles with 0.1 noise), being ranked on fifth place, it was still able to achieve the second place in the average rank. This is because its performance on the other three datasets was very stable, having achieved second place in all of them. Note that, for this database, the RBF SVM model is ranked as first, in all datasets, and the MBHL model is ranked right after it in the average rank, followed by the AdaBoost model, on third place.

Both Circles and Moons datasets can be challenging to classify due to their non-linearity property, specially when noise is inserted in the problem. Classifiers which are based mainly on

Table 8 – Results: mean accuracy rates for each dataset obtained by the following models, in that order: MBHL, AdaBoost, Bagging of Decision Tree, Bagging of MLP, Decision Tree, Logistic Regression, MLP, Naive-Bayes, Random Forest and SVM. The values between parenthesis indicate the rank achieved by each model on each dataset.

		MBHL	**Ada**	**BagDT**	**BagMLP**	**DT**	**LR**	**MLP**	**N-B**	**RF**	**SVM**
Artificial	Circles_0	0.985 (2)	0.983 (3)	0.976 (5)	0.725 (9)	0.982 (4)	0.388 (10)	0.797 (7)	0.784 (8)	0.969 (6)	1.0 (1)
	Circles_01	0.796 (5)	0.820 (3)	0.818 (4)	0.696 (9)	0.795 (6)	0.396 (10)	0.750 (8)	0.762 (7)	0.831 (2)	0.896 (1)
	Moons_0	0.988 (2)	0.979 (3)	0.922 (6)	0.857 (8)	0.933 (5)	0.844 (10)	0.855 (9)	0.864 (7)	0.968 (4)	0.998 (1)
	Moons_01	0.973 (2)	0.955 (3)	0.912 (5)	0.852 (9)	0.909 (6)	0.845 (10)	0.860 (7)	0.859 (8)	0.951 (4)	0.976 (1)
	Average Rank	2nd	3rd	5th	9th	6th	10th	8th	7th	4th	1st
Real	Bankruptcy	0.993 (4)	0.996 (2)	0.995 (3)	0.961 (8)	0.995 (3)	0.962 (7)	0.963 (6)	0.957 (9)	0.997 (1)	0.986 (5)
	Haberman	0.594 (4)	0.465 (9)	0.496 (7)	0.651 (2)	0.489 (8)	0.700 (1)	0.649 (3)	0.534 (5)	0.500 (6)	0.496 (7)
	Hayes-roth	0.651 (4)	0.592 (7)	0.712 (2)	0.553 (9)	0.690 (3)	0.513 (10)	0.569 (8)	0.633 (5)	0.721 (1)	0.613 (6)
	Iris	0.938 (5)	0.930 (7)	0.950 (2)	0.918 (8)	0.930 (6)	0.913 (10)	0.918 (9)	0.945 (3)	0.940 (4)	0.965 (1)
	Wine	0.961 (5)	0.700 (10)	0.927 (8)	0.932 (7)	0.894 (9)	0.964 (3)	0.940 (6)	0.961 (4)	0.971 (2)	0.977 (1)
	Zoo	0.927 (2)	0.801 (9)	0.904 (7)	0.925 (4)	0.860 (8)	0.928 (1)	0.925 (5)	0.925 (5)	0.926 (3)	0.918 (6)
	Average Rank	2nd	9th	4th	8th	7th	6th	7th	5th	1st	3rd

Source: Research data.

the linear distance among instances tend to perform poorer in this type of scenario, since testing instances from different labels get more mixed, and the classes oftentimes overlap each other in the decision space. In this sense, the relative good performance of MBHL on these type of datasets, finishing on second place overall, indicates that the model is able to correctly detect the topological patterns formation, for the selected datasets, and to adjust its inference process according to these identified patterns.

If we look at Figure 11e and Figure 11f, which show the attributes networks resulted from the training phase for Circles_0 and Moons_0 datasets, respectively, we can note that only the nodes representing instances immediately adjacent to each other in these datasets become connected in the training phase. This happens due to two factors: (1) The form with which the parameter ε is calibrated when building the network, by keeping it at a minimum value in order to generate only the enough number of edges for allowing the connection between all nodes from a same class in the network, and (2) The rule where the model will connect the nodes of different data instances only if all distances between them are within their respective thresholds yielded by the parameter ε, for all dimensions considered. Hence, the capacity of the model to identify the topological patterns in a dataset comes from these two factors combined. In this way, the network is more sensitive to the disturbances in its topological structure provoked by the insertion of new instances during the testing phase. Also, nodes representing instances close to each other in one of the dimensions, but far from each other in other dimensions, do not get connected, since all features of the dataset are considered when generating the edges between different instances.

Regarding the obtained results when applying the proposed MBHL model on real datasets, we can note, in Table 8, that it achieves its best relative performance on Zoo dataset, being ranked as second best, right after the Logistic Regression model and followed by Random Forest model. Likewise in the tests with artificial datasets, although its relative performance do not really stand

Table 9 – "Meaningfulness" levels (γ values) provided by the model for each feature in the Zoo dataset.

Hair	Feathers	Eggs	Milk	Airborne	Aquatic	Predator	Toothed	Backbone	Breathes	Venomous	Fins	Legs	Tail	Domestic	Catsize
0.07	0.09	0.07	0.1	0.03	0.06	0.05	0.03	0.11	0.08	0.03	0.04	0.07	0.11	0	0.07

Source: Research data.

out from the other classifiers, when it comes to the other real datasets, it was nevertheless able to achieve second place in the average rank. This is because the MBHL model was overall more stable than others, in terms of relative performance, being ranked as 4th or 5th on the other real datasets.

The good relative performance of the MBHL model on the Zoo dataset – which has a total of 16 features, with most of them being binary ones – is a sign that the parameter γ is properly fulfilling its role, of measuring the "meaningfulness" of each attribute in the classification task. The γ values for this specific dataset, along with their respective feature description, are listed in Table 9. As one can note, the use of Ridge Regression resulted in smaller values for the weights, with a maximum value of 0.11, and only one of them is set as 0 (the "Domestic" feature), which means that the model identified this attribute as not meaningful for the classification task. It is also worth noting that the Zoo dataset has 7 classes and only 101 samples, which indicates that the model can also learn well even when the number of data instances per class is limited. As for the results obtained on the Haberman dataset, we would like to point out that, after 50 random train-test data splits, half of the models (5 out of 10) still achieved a mean accuracy of less or equal to 50% on it. Considering that this dataset has only 2 classes, then it means that this classification problem is a challenging one, and thus the mean accuracy achieved by the MBHL model on it, of 59.4%, can be considered quite satisfactory.

4.5 Chapter Remarks

In this chapter, we have introduced a high level classification model that maps each attribute of the input dataset as a node of a network, and the labels assigned to testing instances are based mainly on the modularity measure. Additionally, we make use of an attention mechanism, by giving more weights to those features identified as the most important – or meaningful – ones in the dataset for classification purposes. To evaluate the model, we applied it on both artificial and real benchmark datasets, and compared its performance to the ones achieved by other traditional classification models. The obtained results on artificial datasets indicate that the model is able to correctly detect topological patterns in the data, including those with non-linearity properties, and to adjust its inference process accordingly. The results obtained from its application on real datasets were also encouraging, with the model being able to achieve competitive mean accuracy rates when compared to traditional classifications models.

CHAPTER

5

ANALYZING VOTING DATA AND PREDICTING CORRUPTION AMONG BRAZILIAN CONGRESSMEN

In this chapter, we approach the problem of analyzing voting data from Brazilian congressmen using a network-based model. For this end, a database is built especially for this study, comprising almost 30 years of legislative work in the Brazilian House of Representatives. Two different types of analyses are made on this database. The first analysis involves the generation of a temporal network, where each node represents a congressman and the edges between each pair of nodes are created according to their voting record similarities. Afterwards, we investigate how the changes in the congressmen temporal network's topological structure might be related to some of the main political transitions happened in Brazil during the same period, such as the alternations of PSDB and PT parties in the presidency, and the impeachment of president Dilma, in 2016. The second analysis concerns the investigation of whether it is possible to conceive a model to predict that a congressman will be convicted of corruption or other financial crimes in the future, solely by considering his voting history.

5.1 Introduction

In this study, we propose a network-based approach for analyzing voting data in the form of representatives' temporal networks to capture the topological structural changes in time and reveal how these changes may be reflected in (or *by*) some of the main political events happened during the same period in Brazil, from 1991 until 2018. Our analysis starts by converting each voting session into a static network, in which each node represents a congressman and each edge represents the accumulated similarity between a pair of congressmen based on their voting history. Afterwards, these static networks are converted to temporal networks, by considering all of them as being an evolving network. We apply this technique to official data from the Brazilian

House of Representatives, comprising the votes of 2,455 congressmen in a total of 3,407 voting sessions from 1991 until 2019, hence covering a range of almost 30 years of legislative works. The obtained results are able to capture the main political transitions happened during the period in terms of the relative positions occupied by each political party in the network. We also find out that, surprisingly, the proposed technique is capable of identifying convicted representatives in the network with high precision and most of them are for corruption charges. This method can be used to predict cases of corruption or other financial crimes. Such a feature comes out unexpectedly since the networks' edges are generated only based on the representatives' legal public activities (voting history), without any financial or other relative information of any sort.

In summary, this study makes use of specific dynamical measures for analyzing the Brazilian legislators' networks. Moreover, it shows how the network-based framework can be applied to identify future cases of corruption or other financial crimes among congressmen with high accuracy, just based on the voting data. Therefore, we believe this work makes an important advance in the large scale public data study using complex networks.

Regarding the organization of this chapter, besides this introduction, we discuss, in section 5.2, the motivations for this study. In section 5.3, we explain the methodology used in the analyses, showing how the congressmen temporal network is generated, and also introduce the methods tested for conviction prediction purposes. The database especially built for this study is introduced in this section as well. In section 5.4, we present the results obtained from our analyses, along with some relevant discussions. At the end, in section 5.5, we close this chapter by adding the final remarks.

5.2 Motivation

In the last years, governments around the world have been trying to increase their transparency by making large amount of public administration data available to the population (ARMSTRONG, 2005; JAEGER; BERTOT, 2010). This phenomenon had triggered the development of new methods specifically designed for the analysis of such kind of data. Within this context, network-based techniques have been applied to politics-related data, such as on the analysis of the legislators' relations through bill co-sponsorship data (KIRKLAND; GROSS, 2014; NEAL, 2018) and through roll-call voting data (ANDRIS *et al.*, 2015; MASO *et al.*, 2014; MOODY; MUCHA, 2013; WAUGH *et al.*, 2009). A comprehensive review on this topic has been made by Victor, Montgomery and Lubell (2017).

There are also relevant applications of network-based approaches on the analysis of networks for crimes-related purposes. Wachs *et al.* (2019) studied the social aspects of corruption by relating the social capital of Hungarian settlements to the risk of corruption in its local government, using large-scale social network data, finding that settlements with high bonding social capital tend to award contracts with higher corruption risk, while settlements with high

bridging social capital tend to award lower corruption risk contracts. Berlusconi *et al.* (2016) tested link prediction techniques on the identification of missing links among an Italian mafia group and Ribeiro *et al.* (2018) made use of the same techniques on politicians cited on corruption scandals in Brazil. In a more recent work (LUNA-PLA; NICOLÁS-CARLOCK, 2020), a network-based approach has been applied to modeling a major corruption scandal occurred in Mexico involving embezzlement activities, thus contributing to provide an objective perspective of the systemic nature of events where companies are abused for corrupt purposes. The study from Ribeiro *et al.* (2018), regarding networks originated from corruption scandals in Brazil, is the one that originally inspired us to conceive the investigation methods presented in this chapter.

5.3 Materials and Methods

The methodology used in this study is summarized below. In subsection 5.3.1, we present the database constructed from Brazilian congressmen voting data, used in the analyses performed in this work. In subsection 5.3.2, we explain how the voting data are mapped as static networks, where each node represents a congressman. In subsection 5.3.3, we describe how the previously generated static networks are then analyzed in the form of a singular temporal network, whose nodes and edges evolve in time. At the end of the section, in subsection 5.3.4, we present the corruption prediction method tested in the final resulting congressmen network, based on the assumption that convicted representatives tend to vote alike in legislative sessions.

5.3.1 Database

The data are collected from the official website of the Brazilian House of Representatives (CÂMARA, 2018) within their transparency section. These datasets comprise the outcome of 3,407 voting sessions of legislative bills deliberated in the House of Representatives, from May 22, 1991 until Feb 14, 2019. We made a thorough data cleansing process in this database in order to detect and fix possible mistakes, such as duplicated names or votes and also typographical errors. Each voting session contains the following attributes: the bill to be voted, the voting date, and for each representative who attended the session: IDE (a unique number for each of them), Name, Political Party and Vote. The voting data are similar to roll call votes, except that here there are four different types of votes: (1) *Yes*, if the representative approves the bill; (2) *No*, if the representative disapproves the bill; (3) *Abstention*, if the representative deliberately chooses to not take part in the voting; and (4) *Obstruction*, similar to abstention, with the difference that abstention counts for *quorum* effects, i.e., the minimum number of voting members who must be present at the session, while obstruction does not count for it.

After extracting and cleaning the data from the 3,407 voting sessions, we end up with a total number of 2,455 representatives and 1,656,547 votes. For analyzing these data, we opt for making use of a network-based technique, specially developed for this purpose. Firstly,

we convert each voting session into a separated static network. Afterwards, we select some of these static networks to generate temporal networks and then perform some analyses in order to examine how their *topology* – in terms of network temporal measures – evolve in time.

As for the conviction classification task, also tested in this study, we add an additional attribute, for all representatives, which indicates whether he or she is currently convicted or have been arrested for corruption or other financial crimes, such as money laundering, peculation, embezzlement or misappropriation of public funds, improbity and crime against the Public Administration. This information has been confirmed from Brazilian judiciary official sources, such as the Federal Supreme Court (Supremo Tribunal Federal) (STF, 2019). At the end of this research, we were able to identify a total of 33 representatives in our database who currently have been either arrested or convicted for corruption (21 congressmen) or for other financial crimes (12 congressmen).

5.3.2 Static Network Generation

A network can be defined as graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is a set of nodes and $\mathscr{E}$ is a set of tuples representing the edges between each pair of nodes $(i, j) : i, j \in \mathscr{V}$. The process of mapping each voting session in the database into a network is made according to their respective date attribute, sorted in ascending order, strictly. For the first voting, when $t = 0$, its data items are initially converted to a square votes matrix M^t of size $d \times d$, where d is the total number of representatives who participated in the session. Each element M^t_{ij} is a binary value: it assumes 1 if the vote of representative i is equal to the vote of representative j; otherwise, it assumes -1. These values are accumulated in a separated weight matrix W^n, in which each element W^n_{ij} is equal to the sum of values of M^t_{ij} in all votes matrices M^t until voting session n. Hence, each item W^n_{ij} of this matrix represents the accumulated weight between representatives i and j. The time steps t are measured in terms of voting sessions. Mathematically, the current value of each weight W^n_{ij} is given by:

$$W^n_{ij} = \sum_{t=0}^{n} M^t_{ij} \quad . \tag{5.1}$$

Therefore, from Equation 5.1, the values in each row W^n_i may range from $-n$, in the case that the representative i always voted differently from representative j, until n, which is the case when i and j always voted alike. The former case implies that, up to the current instant, representatives i and j have complete opposite political views, while, in the latter case, i and j are very aligned up to now, politically speaking. Another possibility here, in this technique, would be binning the votes similarities per predetermined periods of time, such as per presidential term or per year. After some preliminary processing of the database, we have noted that it takes varying voting time to emerge a clear topological pattern in the networks, therefore, in our case it is more suitable to take all historical votes into consideration for generating the weight matrix W^n.

After generating the matrices M^t and W^n, the next step is to generate a network $\mathscr{G}^t$, for

each voting session t, such that each representative becomes a node in $\mathscr{G}^t$. The edges in $\mathscr{G}^t$ are created according to the following rule:

$$\mathscr{G}^t_{ij} = \begin{cases} W^n_{ij}, \text{ if } W^n_{ij} = \max\limits_{\forall x \in W^n_i} x \\ 0, \text{ otherwise .} \end{cases} \tag{5.2}$$

As a result of Equation 5.2, the great majority of the vertices in $\mathscr{G}^t$ will have only one outbound edge, connecting it to the most politically aligned vertex. Vertices with more than one outbound edge may only occur when the function $\max_{\forall x \in W^n_i} x$ returns more than one value. The most connected vertices in the network (*hubs*) will be the ones with the highest number of inbound edges.

The main procedures of our technique can be summarized in the following steps:

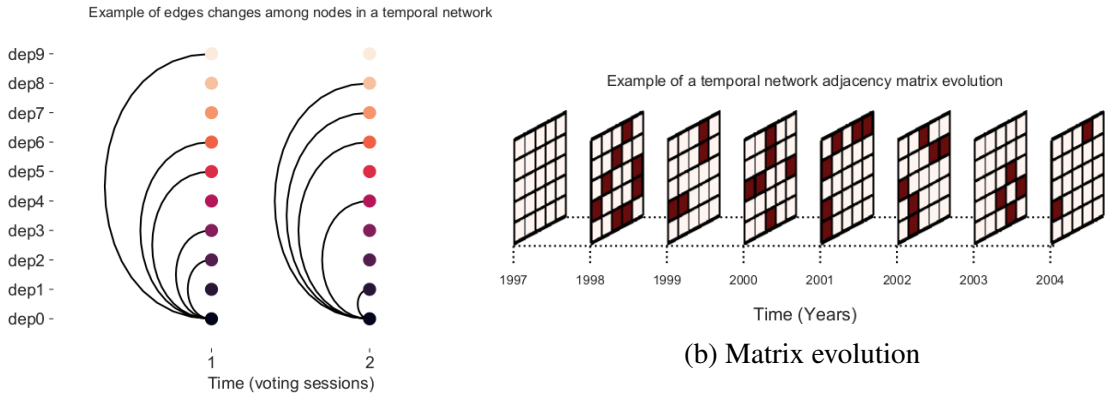
1. build votes matrix M^t from data of voting session t;
2. update weight matrix W^n, also inserting new representatives in it, if any;
3. build network $\mathscr{G}^t$, whose values come from the weight matrix W^n; and
4. repeat the procedure for next voting session $t+1$, until the last one in the dataset.

As a consequence of this process, the networks $\mathscr{G}^t$ evolve in time, as their edges are determined by the accumulated weights between pairs of representatives from matrix W^n, which is updated at each step t. The vertices, representing the congressmen, may also be replaced by new ones along the process, as new representatives appear in the voting session lists, in such a way that the nodes, in this case, can be seem as the seats in the House. When a new congressman is inserted into the network (because he/she has been elected or for any other reason), he/she does not inherit any voting information from the congressman who previously occupied its seat in the House (or node in the network). In this case, the model adds a new row and a new column in matrix W^n to store the voting similarities between the node of the new congressman and all other nodes in W^n. It is also worth noting that the attribute "political party" is not taken into account by the model to generate the network's edges. We have opted to proceed this way because, in this study, our aim is to capture the affinities among representatives beyond their political party affiliations, i.e., by only taking into account their votes on legislative bills for network generation. This makes sense whereas, in the case of Brazil, there are currently as much as 33 different political parties, and this excessive number of parties ultimately tends to make the ideological differences between them to diminish substantially.

5.3.3 Temporal Network Generation

After running our algorithm for all bills in the database, we end up with a total of 3,407 networks, each one with around 500 nodes and representing a different voting session during

Figure 13 – (a) Illustration showing how the temporal network edges, or *graphlets*, evolve in time, here measured in terms of voting sessions. When time slice $t = 1$, representative 0 is connected to representatives 2, 3, 5, 6 and 9. In the next time slice $t = 2$, it loses the connections with representatives 2, 3, 5 and 9 and receives edges from representatives 1, 4, 7 and 8. (b) Example demonstrating the adjacency matrix evolution in a temporal network, whose dimension $\mathscr{D}$ is measured in units representing years. The network edges are generated according to this matrix.



(a) Edges evolution

(b) Matrix evolution

Source: Elaborated by the author.

the last 28 years. Thus, we can also say that all these networks, in fact, represent different moments of the Brazilian congressmen network. At this point, we already have shown how to generate these networks in a static form, each $\mathscr{G}^t$ representing a moment at time t. For the sake of converting these networks into a single temporal network $\mathscr{G}$, we need then to insert a new dimension $\mathscr{D}$ in the static network definition, such that it becomes $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{D})$, where $\mathscr{D}$ stands for the network temporal slices or, in our case, the voting sessions. To achieve this, we generate a matrix for representing each edge $\mathscr{E}$ in the static networks' slices in the form of a triplet $(i, j, t) : i, j \in \mathscr{V}, t \in \mathscr{D}$. These triplets are also known as dynamic *graphlets* (HULOVATYY; CHEN; MILENKOVIĆ, 2015) and an illustration of their dynamics is shown in Figure 13a. The final result of this conversion process is a *multilayer network*, in which each layer represents a static temporal slice of a single main graph (Figure 13b). In this case, since the dimension $\mathscr{D}$ is a set of indices ordered by time, we can therefore also call this graph a *temporal network* (THOMPSON; BRANTEFORS; FRANSSON, 2017; HOLME; SARAMÄKI, 2012), and perform analyses on it by extracting some specific measures.

Extracting temporal measures from a network with over 3,000 time slices, each one having around 500 nodes, is a time-consuming process. Therefore, in this work, we decide to make use of only one time slice per year for generating the temporal network. The selected voting sessions, as well as the current presidency at each period and his/her corresponding political party, are described in Table 10. It is worth noting that the voting sessions sampling (with one session in each year being selected as a temporal network slice) has little effect on the overall results, since our network formation technique certifies that the weight of each edge, stored in

Table 10 – Voting sessions used for generating the temporal network slices, yearly

Year	Bill voted	Session date	Presidency
1991	PL 638/1991	1991-08-28	Collor (PRN)
1992	PL 2747/1992	1992-04-29	
1993	PL 1258/1988	1993-04-01	Itamar (PRN)
1994	PDC 413/1994	1994-04-20	
1995	PL 233/1995	1995-04-04	FHC I (PSDB)
1996	PL 824/1991	1996-04-10	
1997	PEC 173/1995	1997-04-09	
1998	PEC 33/1995	1998-04-29	
1999	PL 1/1995	1999-05-12	FHC II (PSDB)
2000	PEC 96/1992	2000-04-05	
2001	PLP 23/1999	2001-04-03	
2002	MPV 14/2001	2002-04-10	
2003	MPV 86/2002	2003-04-01	Lula I (PT)
2004	PEC 101/2003	2004-05-19	
2005	MPV 242/2005	2005-06-07	
2006	MPV 269/2005	2006-04-04	
2007	MPV 339/2006	2007-04-10	Lula II (PT)
2008	MPV 415/2008	2008-04-23	
2009	MPV 452/2008	2009-04-14	
2010	MPV 475/2009	2010-05-04	
2011	REQ 343/2011	2011-04-06	Dilma I (PT)
2012	PEC 153/2003	2012-04-10	
2013	PEC 544/2002	2013-04-03	
2014	PLP 221/2012	2014-05-07	
2015	MPV 660/2014	2015-04-07	Dilma II (PT)
2016	REQ 4250/2016	2016-04-04	
2017	PL 5587/2016	2017-04-04	Temer (MDB)
2018	PL 3734/2012	2018-04-11	

Source: Research data.

the weight matrix W^n, already carries in itself the information regarding all bills previously voted until present.

Besides generating one main temporal network, which includes all 28 time slices in Table 10, we also generate one temporal network per presidential term, for the sake of comparison purpose. The measures extracted from the resulting temporal networks are listed below.

- *Temporal degree centrality* (D^T): the number of overall connections in time per node.
- *Temporal participation coefficient* (P^T): a measure of diversity of connections across communities for individual nodes (GUIMERA; AMARAL, 2005). The communities are detected by using the Louvain method (MEO *et al.*, 2011).

We also calculate a "proportional" version of each temporal measure M^T, grouped by the political

party p of each node i, defined as:

$$M_p^T = \frac{\sum_i M_{i_p=p}^T}{\sum_i M_i^T}, \tag{5.3}$$

where p is a political party and i_p returns the party of node i. These proportional versions of the measures are used for comparison among parties.

5.3.4 Conviction Prediction

Now, let us proceed to describe how we assess whether a representative is more likely to be convicted or arrested in the future by analyzing the voting agreements among congressmen. Two different methods have been tested for accomplishing this task: the first one is based on the matrix W^n values, while the second one is based on the network link prediction model. Following, we describe the two methods with more details.

5.3.4.1 Conviction Prediction Based on the Weight Matrix

After finishing the processing of all voting sessions, we end up with the network resulting from the final weight matrix W^n. This network has 2,455 nodes, representing all congressmen who voted in at least one legislative bill from 1991 until 2019, along with their respective pairwise voting history similarities. While browsing this main network, we note that the highest weighted neighbors of a node labeled as convicted are more likely convicted ones as well, apparently forming some sort of "corruption neighborhoods" in the network. Hence, we decide to investigate this aspect further by running a very simple algorithm, which basically takes the n highest weighted neighbors of a convicted representative, according to the weights stored in W^n, and labels all of them as also convicted ones. Thus, we have that the "convicted" label c of a node i is defined as follows:

$$i^c = \begin{cases} \text{True, if } j^c = \text{ True }, \forall j \in k\text{NN}_i \\ \text{False, otherwise }, \end{cases} \tag{5.4}$$

where $k\text{NN}_i$ returns the n neighbors with the highest weights associated to node i. We assess the efficiency of this model by measuring its prediction accuracy for different values of n. The rationale behind this model is that arrested or convicted representatives, for some reason, tend to vote similarly on legislative bills.

5.3.4.2 Conviction Prediction Based on Link Prediction

Given that the simple model described above does not consider the network topological structure for prediction purposes (only considers the weight matrix W^n), we thus also test another method for accomplishing this task, which makes use of models for predicting missing links in networks. The guidelines of this method are described below:

1. generate subgraph from an *undirected* version of the network resulting from matrix W^n, containing only arrested or convicted representatives and their neighbors;
2. remove all existing links between convicted labeled nodes from this network (subgraph);
3. apply the link prediction model to the network; and
4. take the top n link predictions whose source is a convicted labeled node and classify their target nodes as also convicted ones.

One of the models tested for this task is Rooted PageRank (LIBEN-NOWELL; KLEINBERG, 2007), which is based on an algorithm developed for ranking the importance of website pages (PAGE *et al.*, 1999). It defines the $score(x,y)$ as the expected number of steps required for a *random walk* on the network starting from node x, moving iteratively with a probability α to return to x (or "reset") and a probability $1-\alpha$ to move forward to a random neighbor until it reaches the node y. The lower the *score* for each pair of nodes x and y is, the higher the pair is ranked among the model's list of predicted links. Besides Rooted PageRank, other 5 link prediction models are also applied to this task: Pearson (AHLGREN; JARNEVING; ROUSSEAU, 2003), Cosine (SALTON; MCGILL, 1986), NMeasure (EGGHE; LEYDESDORFF, 2009), MinOverlap (ESQUIVEL; ROSVALL, 2011) and Random (for comparison purposes). By making use of a link prediction model, we are now taking into account the congressmen network topological structure for conviction prediction purposes.

5.4 Results and Discussion

In this section, we present the obtained results when applying our method of analysis to the database comprising almost 30 years of legislative voting data from Brazilian representatives. We start, in subsection 5.4.1, by presenting the analyses regarding the congressmen temporal network evolution, in terms of topological features. Then, in subsection 5.4.2, we present the results obtained when testing a corruption prediction technique, based on the assumption that corrupt representatives tend to vote alike in legislative sessions.

5.4.1 Political Scenario Through the Analysis of the Representatives' Networks

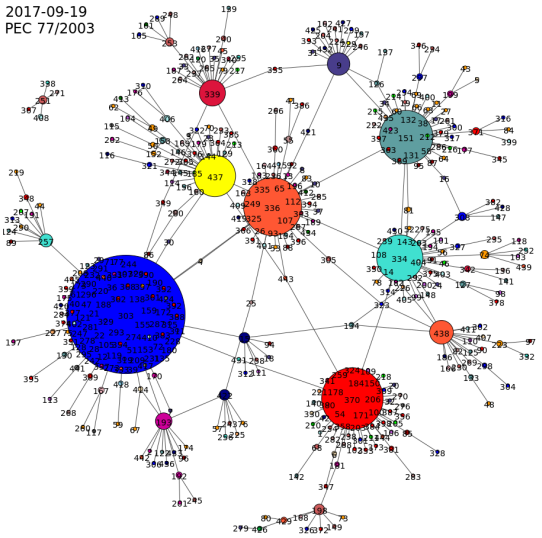
As mentioned earlier, our initial task involves the generation of over 3,000 static networks in total, then, a comprehensive temporal network is built. We start this subsection by presenting an example of one of these static networks, shown in Figure 14a, built from the voting session of bill PEC 77/2003, occurred on September 19, 2017. The outbound edges connect each node to the one with highest accumulated weight associated with it. One feature that called our attention in most of these networks is that, even though the *political party* attribute, represented by the

color of the nodes in the figure, has not been taken into account explicitly by the algorithm, we still can note the formation of neighborhoods based on political parties in the networks, centered at *hubs*. This feature confirms that representatives from the same party tend to vote alike in legislative bills, thus the formation of party clusters occurs. If a node is connected to a neighborhood different from its own party's, then the congressman represented by this node has been voting more similarly to the representatives from other parties. As expected, still in Figure 14a, the colors of the biggest hubs in the network coincide to those from parties with most members in the House of Representatives at that time. The colors in *blue*, *red*, *cadet-blue* and *orange* represent parties PSDB, PT, PP and MDB, respectively, which were main parties in the Brazilian congress in September 2017. The hubs, within this context, represent the congressmen who voted according to each "local majority" in the network, i.e., the majority within a local neighborhood.

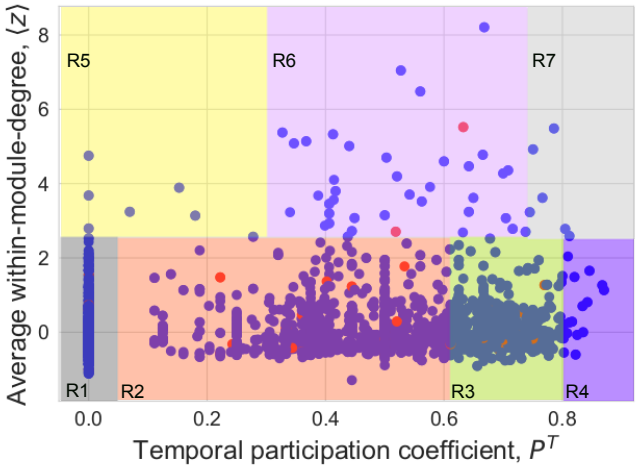
Alternation of power is an important and expected condition of democratic systems. Within this context, we analyze the temporal networks segmented by each presidency, with the aim of measuring the evolutionary strength of the two main political parties (PSDB and PT) in Brazil during the considered period, in terms of the position they occupy in the network, and examine how these changes may be related to the main political events happened in the same period. We initially extract two centrality measures from each network: temporal degree centrality D^T and temporal participation coefficient P^T, which give us centrality scores for each node. Afterwards, we calculate the ratio of each of those measures for the parties PSDB and PT, according to Equation 5.3, in each presidential term. For cases when a representative switched parties during the period, we then consider the party to which he belonged at the time of each voting session, i.e., each time slice. The results of this process are shown in Figure 14c and Figure 14d. Observe that the ruling political party presents higher values for both centrality scores measured in the congressmen temporal network, and such a feature strictly follows the respective alternation of power between PSDB (FHC governments, from 1995 until 2001) and PT (Lula and Dilma governments, from 2002 until 2016). It is also worth noting that, in these figures, there is a sudden drop in both measures for PT party in the second term of Dilma (2015-2016), which coincides with the turbulent political scenario in Brazil at that time, when many demonstrations were held against Dilma – specially after her predecessor Lula was charged by federal prosecutors with corruption accusations against him and his party – and end up in her impeachment, in the end of 2016. A similar behavior could also be observed for the PTC party (former PRN) in 1992 (the impeachment of former president Collor occurred at that time), although in a much smaller scale since this is a minor political party in Brazil. This event is not included in these figures for the sake of visibility.

Following, we generate what is known as the *network cartography* (GUIMERA; AMARAL, 2005) for the temporal network which includes all 28 time slices (from 1991 until 2018, yearly). This framework helps us to better understand the network topological structure by grouping the nodes into some "universal roles", according to their level of connectivity in-

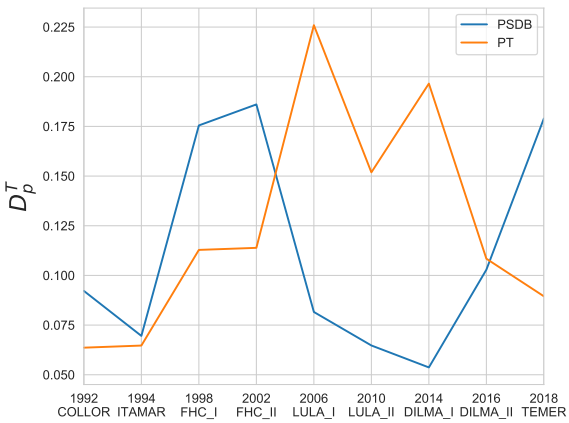
Figure 14 – (a) Example of a static network generated by our algorithm for the voting session occurred on 2017-09-19 of legislative bill PEC 77/2003. Each node represents one of the 513 congressmen who voted this bill and each color represents a different political party. (b) Node roles based on the network cartography framework, with the adaptation that, here, we use the temporal version of the participation coefficient (P^T) with averaged within-module-degree, z-scores, from each temporal network slice t. Each point represents a congressman and the *red* color denotes convicted ones. (c) Proportional temporal degree centrality D_p^T and (d) proportional temporal participation coefficient P_p^T measures evolution, calculated for all representatives and grouped by political party p, for each presidential term. The evolution of both measures coincide precisely with the respective alternation of the ruling parties PSDB (FHC) and PT (Lula and Dilma).



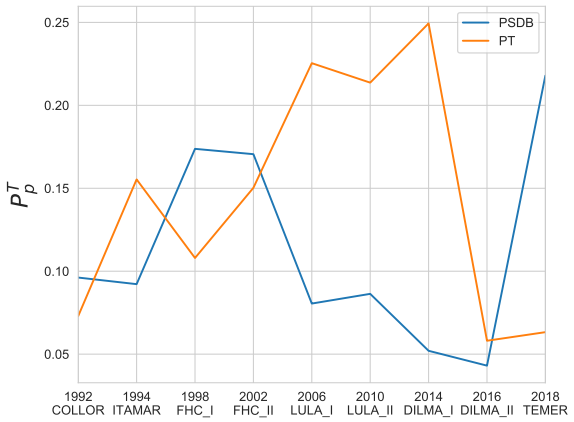
(a) Example of static network



(b) Temporal network cartography



(c) Proportional temporal degree centrality D_p^T



(d) Proportional temporal participation coefficient P_p^T

Source: Elaborated by the author.

side the network. It depends on two measures: the *within-module degree* z_i, which shows how "well-connected" a node i is to other nodes within its module, and the *participation coefficient* P_i, which shows how "well-distributed" the links of node i are among different modules. For accomplishing this task, we make a slight adaptation from the original technique. For static networks, the within-module-degree returns a single value z_i for each node i. As for temporal networks, instead, it returns a 2-d array in the form of z_{it} with one value of z_i for each time slice t. Therefore we opt here for averaging these values, such that $z_i = \overline{z_{it}}$, in order to generate the network cartography. We also make use of the temporal participation coefficient P_i^T, instead of its static version P_i. The output can be seen in Figure 14b. Each point in this plot represents a congressman and the *red* color denotes those nodes labeled as convicted ones. The distribution of their network roles is summarized in Table 11, grouped by convicted and the others (those who have not been officially convicted). It shows that around 98% of them are non-hubs (roles R1 to R4) and only about 2% of them are module hubs (roles R5 to R7), also indicating that convicted representatives tend to have a slightly higher incidence of *connector hubs* (R6), which are hubs with links to most of the other modules.

Table 11 – Network cartography: node roles distribution (%)

Role	Convicted	Others	Description
R1–ultra-peripheral	62.0	63.1	nodes with all their links within their module
R2–peripheral	26.0	25.4	nodes with most links within their module
R3–non-hub connector	9.8	9.7	nodes with many links to other modules
R4–non-hub kinless	-	0.5	nodes with links homogeneously distributed among all modules
R5–provincial hubs	-	0.2	hubs with the vast majority of links within their module
R6–connector hubs	2.2	1.0	hubs with many links to most of the other modules
R7–kinless hubs	-	0.1	hubs with links homogeneously distributed among all modules

Source: Research data.

5.4.2 Prediction of Conviction Among Representatives

The incidence of corruption impacts the society negatively in many ways, such as holding back businesses, wasting public spending and undermining the democratic system. Predicting the incidence of corruption and other related financial crimes, specially at the individual level, is a challenging task. Nowadays, a prediction system with an average accuracy around 0.2 is already considered useful by public investigators all over the world (RIBEIRO *et al.*, 2018). Here, we make use of a network-based approach to identify hidden connections among convicted congressmen linked to bribing schemes or other financial crimes in Brazil. Two methods are tested for detecting future convictions among representatives. The first method is based on the nearest neighbor of convicted congressmen using the weight matrix W^n and the second one is based on link prediction. The former achieves prediction accuracy about 0.24, while the latter achieves accuracy beyond 0.5, even up to 0.9. Consequently, the accuracy obtained by the link prediction model can be considered quite satisfactory. The reason why the prediction accuracy by the two methods are so different is simple: In the first method, a prediction to a congressman

is made by considering only his/her labeled nearest neighbor, i.e., a prediction is conditioned on only one node of the network. On the other hand, in the second method, a prediction is made by link prediction methods, which considers the local or global network structure conditioned on more than one nodes, i.e., a finer filtering is performed.

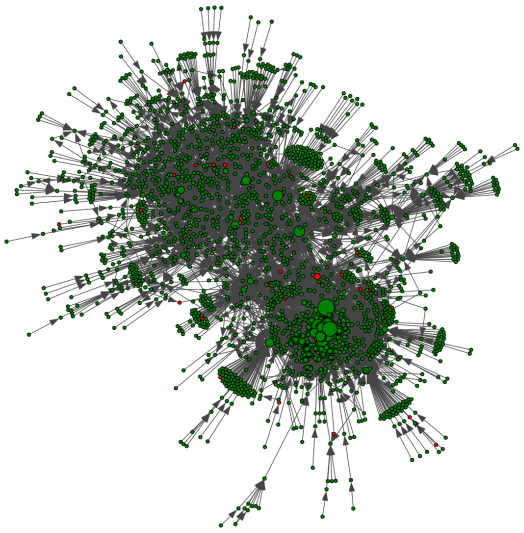
5.4.2.1 Results Based on the Weight Matrix

While browsing the nodes of the network resulting from the final weight matrix W^n (Figure 15a) – the one formed by all representatives, regardless the time factor – the first speculation in mind may be that the highest weighted neighbors of a convicted corrupt representative are possibly convicted ones as well. Therefore, we investigate whether the nodes of convicted representatives tend to stay close to each other in this network, and thus forming some sort of "corruption neighborhoods", so to speak. For this purpose, we build n separated networks composed only by nodes labeled as convicted ones, along with their respective n highest weighted neighbors according to the final weight matrix (these neighbors can be labeled as convicted or not). Afterwards, we run a simple algorithm, as specified in Equation 5.4, which classifies all n neighbors of an already convicted labeled node as being convicted ones as well (whether in the present or in the future). In Figure 15b, it is possible to see the network resulted from $n = 1$, i.e., with the 33 convicted representatives along with the highest weighted neighbor of each of them. It indicates that there is, indeed, the formation of some sort of "corruption structures" in the network. Note that Figure 15b is actually a subgraph of Figure 15a, which has 2.455 nodes and only 33 of them labeled as convicted. So the odds of a convicted node having a neighbor who is also convicted would be very low, if it is not for the incidence of the corruption neighborhoods. The emergence of this feature is something surprising to us, considering that none of the input attributes in our data are related to the congressmen financial income or expenditures and that the edges are generated solely based on their voting history. The conviction prediction results for n in [1, 5] are shown in Figure 15c. From this figure, we see that the optimal value of n is 1, with an average accuracy of 0.24.

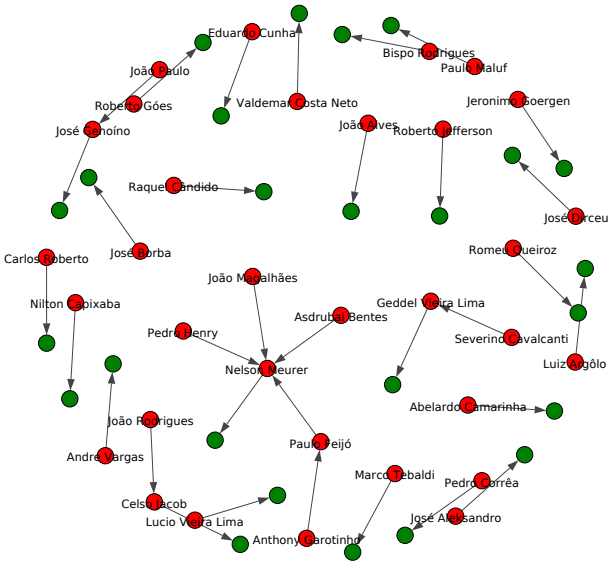
In order to confirm whether there is indeed a correlation between voting similarity and convictions for corruption and other financial crimes among representatives, we run another test by using the same rationale explained above with the difference that, now, instead of selecting the highest weighted neighbor of each convicted node for prediction purposes, we took its $n-$st highest weighted neighbor determined by its outgoing edges, therefore decreasing the voting history similarity between the original convicted node and its neighbor, as n increases. The obtained results, in Figure 15d, show that, in this case, the higher the value of n, the smaller is the accuracy achieved by the algorithm, which contributes to confirming our initial suspicion that convicted representatives indeed tend to vote alike in legislative bills.

The prediction accuracy achieved by our first prediction model is about 0.24 and it is very close to the accuracy achieved by Ribeiro *et al.* (2018), which is around 0.26, when predicting

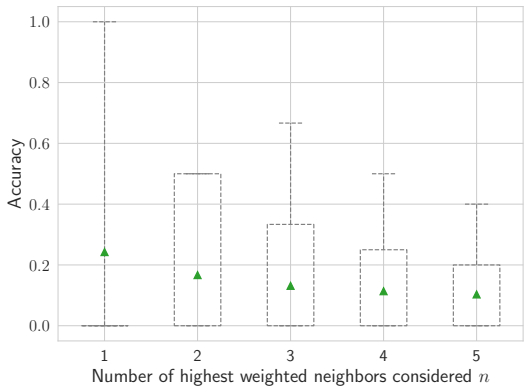
Figure 15 – (a) Representation of the network resulted from the final matrix W^n, with all 2,455 congressmen in the database, disregarding the time factor. Each node is connected to its highest weighted neighbor, in terms of voting similarity. The red color denotes convicted representatives (33 in total). (b) A subgraph of the consolidated network, shown in (a), displaying only the 33 already arrested or convicted representatives (in red) and their respective highest weighted neighbors. We opted for not displaying the names of representatives who currently have not been officially convicted in this graph (in green). (c) Predictions based on the n highest weighted neighbors, in terms of votes similarity, resulted in an average accuracy of 0.243 when $n = 1$. (d) Tests made by considering the $n-$st highest weighted neighbor of a convicted node show that, as we increase the value of n, the lower is the average accuracy.



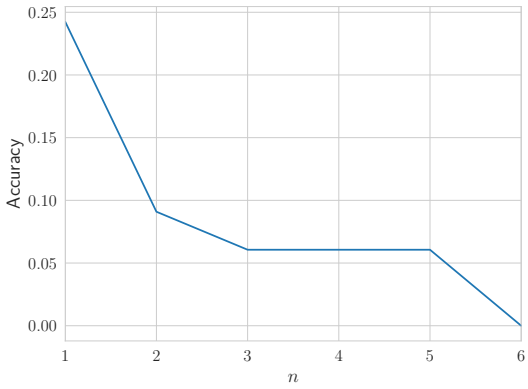
(a) Consolidated network



(b) Convicted congressmen and their respective highest weighted neighbors



(c) Prediction results based on the n highest weighted neighbors



(d) Prediction results based on the n-st highest weighted neighbor

Source: Elaborated by the author.

missing links among politicians cited on corruption scandals in Brazil. Following, we show how the prediction rate can be considerably improved when we take into account the overall network topological structure for making the predictions.

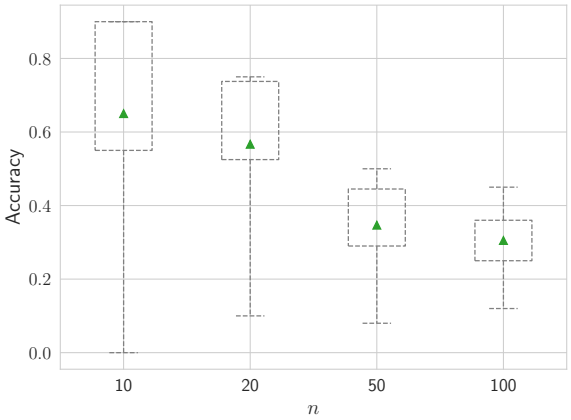
5.4.2.2 Results Based on a Link Prediction Model

The last step in our analyses involves the application of link prediction techniques for the sake of predicting new conviction cases among representatives. For accomplishing this task, we apply a total of 5 link prediction models plus a Random method (for comparison purposes) in the congressmen network. The method based on link prediction differ from the simple one presented in the previous subsubsection because the former makes a prediction considering the network's topological structure (excluding, of course, the random technique from this list), while the latter just takes into account certain neighbors. As in the previous test, the models are also applied to a subgraph of the network resulted from the final weight matrix W^n, formed by convicted representatives and their respective highest weighted neighbors, with the difference that, at this time, neighbors from both incoming and outgoing edges are considered, and also that the network is previously converted to an *undirected* one. This final subgraph contains 211 nodes (33 of them being convicted) and 1,374 edges. As preprocessing, we remove all existing links between two nodes labeled as convicted from the network (5 in total). After running the link prediction models, we took the top n predicted links with convicted nodes as sources and label their target nodes as being convicted ones as well. All tests are performed using the tool introduced by Guns (2014), with default parameters values for all models.

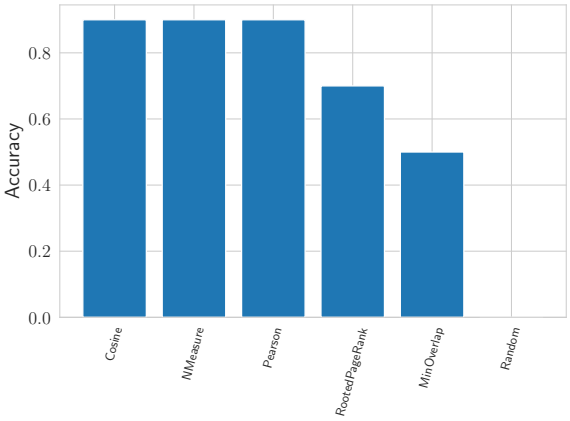
The obtained results of all 6 link prediction models under consideration are shown in Figure 16a and Figure 16b. Figure 16a shows how the value of n, in this case, may affect the overall results, where $n = 10$ is the most indicated among the tested values, with an average accuracy of 0.65 (around 6 correct ones out of every 10 predictions, then). Figure 16b brings the accuracy achieved by each model, with Cosine, NMeasure and Pearson showing an impressive performance with an accuracy of 0.9, followed by Rooted PageRank and MinOverlap, with an accuracy of 0.7 and 0.5, respectively. It is worth noting that the Random predictor scored 0 in this task, which contributes to highlighting the effectiveness of applying the graph-structure-based predictors.

Comparing between the first prediction model with the average accuracy of 0.24 and the link prediction models with accuracy beyond 0.65, we perceive how the performance of a model can be improved whereas one considers the topological structure of the input dataset for classification purposes. This feature becomes more evident given the good results achieved by the first 5 link prediction models shown in Figure 16b. The performance of link predictors, overall, may vary significantly, with some methods being more suitable than others according to the input dataset (LIBEN-NOWELL; KLEINBERG, 2007). In our case, given the technique used for building the congressmen network, two features have emerged from it: (1) the more

Figure 16 – (a) Performances achieved by 6 link prediction models on the task of predicting conviction cases among representatives by considering the top n predicted links whose source node is a convicted one, indicating that the highest scores are achieved when $n = 10$, with an average accuracy of 0.65. (b) Performances achieved by each model, when considering their top 10 predictions, showing Cosine, NMeasure and Pearson with the highest score, with an impressive accuracy of 0.9.



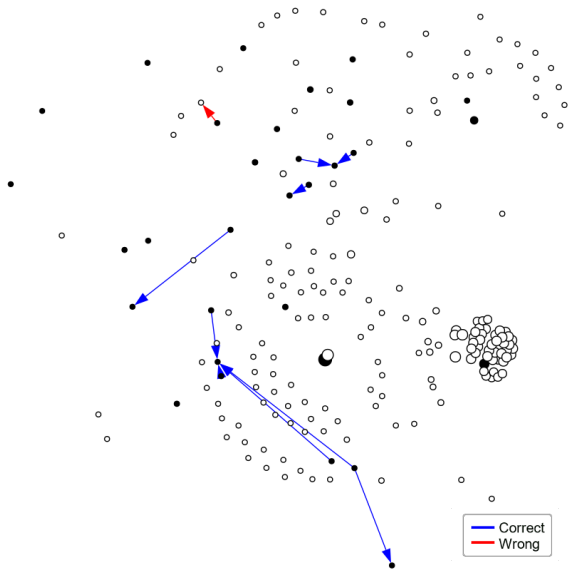
(a) Link predictors accuracy according to their top n predictions



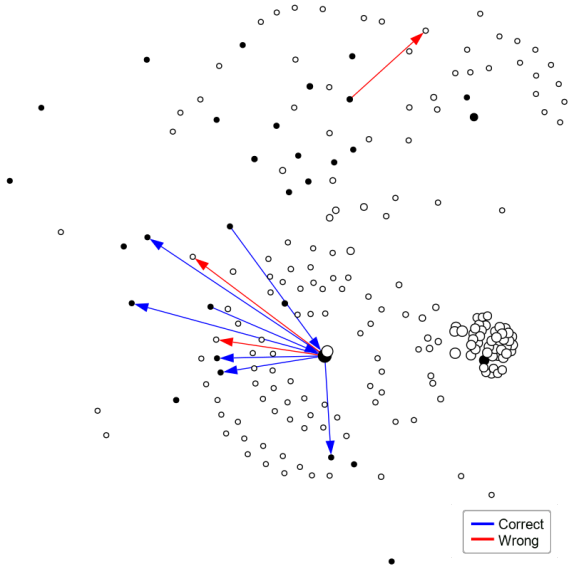
(b) Results based on link prediction, for $n = 10$

Source: Elaborated by the author.

Figure 17 – Comparison of two link prediction outputs for the network formed by convicted representatives and their neighbors: top 10 links having a convicted node as source predicted by (a) Pearson and (b) Rooted PageRank models. Black nodes indicate convicted ones. A link prediction is considered correct if its target node is also labeled as convicted. Remembering that the models do not take the node labels into account for prediction purposes. All other links are removed from the network only for the sake of visibility.



(a) Links predicted by Pearson (9 correct, 1 wrong)



(b) Links predicted by Rooted PageRank (7 correct, 3 wrong)

Source: Elaborated by the author.

politically aligned two representatives are (in terms of their voting history), the nearer they are in the network (in terms of number of links); and (2) only long term representatives are able to become hubs in the network, since a higher number of votes is needed for that. In Figure 17, we show a comparison of the top 10 link predictions from the Pearson and Rooted PageRank models. This figure may help us to better understand why some link-prediction-based methods performed different than others in the task of predicting new convicted nodes. Methods such as Pearson, Cosine and NMeasure have in common the fact of being *local predictors*, i.e., solely based on the neighborhoods of the two nodes considered. Hence, they presented very similar results, also achieving the best accuracy when compared to other methods. This may be related to the feature where convicted nodes tend to stay close to each other in the network, as we saw earlier. As for the Rooted PageRank, which achieved the second best accuracy of 0.7, it is a *global predictor*, such that even if two nodes do not share any common neighbors, they still may be related and form a link in a later stage. One may observe that all 7 correct links predicted by Rooted PageRank have the largest network hub (the one in black, in the center) either as its source or as its target and, in this case, it also happens that the largest hub in this network is a convicted one himself. This feature favors models based on random walks, such as Rooted PageRank, since many of the other convicted nodes are close to this hub.

5.5 Chapter Remarks

Fighting and preventing corruption and other financial crimes are challenging tasks, because criminals constantly develop increasingly advanced mechanisms to cover their infractions. In this study, we present a technique to reveal the hidden relationships between voting behavior and condemnations for corruption and other financial crimes among politicians. We also show how this information can be used to detect those individuals which are more likely to be convicted in the future. To our knowledge, this work is one of the first endeavours to accomplish such task through a network-based methodology. An interesting feature of this work is that the high conviction-prediction accuracy can be obtained using voting data, which implies that it is possible to reveal politicians' illegal behavior through just their legal public activities. Such kind of systems, once developed, can be certainly quite useful to many countries, specially to the countries like Brazil, which seriously suffer from corruption, for a long time.

Our work is inspired by Ribeiro *et al.* (2018), which predicts missing links among politicians cited on corruption scandals in Brazil. Both works (the one of Ribeiro *et al.* (2018) and our work) deal with a similar problem – the incidence of corruption among individuals by using network-based techniques. However, there is a fundamental difference between the two works: The former is based on a dataset composed of 404 politicians cited on at least one corruption scandal and aims to predict citations on future scandals, while our study is based on a dataset comprising the voting history of 2,455 representatives on bills and only considers those already found officially guilty for prediction purposes. Therefore, the dataset used in this work is

not only larger, but also is always available and easy to access. The use of regular public data, as the dataset we use here, presents big facility to develop politician monitoring system in the future. Besides of this, the prediction accuracy achieved by our prediction model, of about 0.9, is much higher than that obtained in Ribeiro *et al.* (2018), which is of around 0.26. We hence believe that the accuracy rate achieved in this work is quite satisfactory. Another related work, of Berlusconi *et al.* (2016), tested link prediction techniques based on a similarity score on the identification of missing links among an Italian mafia group, obtaining a link reliability of up to around 0.9 for predictions made based on common neighbors. However, the prediction accuracy has been counted, in some cases, by considering informal relationships among the members of the mafia, for example, the existence of a phone call between two members (two nodes), which presents certain level of subjectivity. On the other hand, in our work, the corruption prediction accuracy is calculated using official judiciary sources, such that we are certain whether a congressman is convicted or arrested. It means that we are sure about the prediction accuracy of our model.

CHAPTER

6

TREND DETECTION AND AUTOMATIC DECISION-MAKING IN THE STOCK MARKET

In this chapter, a network-based model for price trend detection and automatic decision-making in the stock market is presented. The model starts by mapping the historical prices of a financial asset to a network, where each node represents a variation range and two nodes are connected by a link if their respective variation ranges have ever occurred consecutively in the past. Then, communities are detected in the price variations range network, for characterizing *down* and *up* trends in the asset's price. Afterwards, in the model's operating phase, spot prices are inserted in the network, according to their respective current variations, and the connector hubs in each community are used as indicators of a possible *trend reversal* pattern for the price, possibly triggering a buying or a selling order for the stock.

6.1 Introduction

Many *complex systems* in nature and society can be described in terms of networks to capture the intricate web of connections among the units they are made of (PALLA *et al.*, 2005; WATTS; STROGATZ, 1998; BARABÁSI; ALBERT, 1999; DOROGOVTSEV; MENDES, 2013). A salient feature of networks is the presence of community patterns, one of the examples is the biological neural networks. Data on both anatomical and functional connectome of human (animal) brain has shown the small-world structure with highly clustered modules at different scales (AKIKI; ABDALLAH, 2019; GLEISER; SPOORMAKER, 2010; HAGMANN *et al.*, 2008). These communities are known to represent subsystems of neurophysiological functions, e.g., visual cortex. Besides the brain (SPORNS, 2002), other examples of real-world networks include the internet (FALOUTSOS; FALOUTSOS; FALOUTSOS, 1999), food chains (MONTOYA; SOLé, 2002), blood distribution networks (WEST; BROWN; ENQUIST, 2009)

and power grid distribution networks (ALBERT; ALBERT; NAKARADO, 2004).

Stock market prediction assumes that it is possible to determine the future value of a company stock or other financial asset traded on an exchange. Although there are numerous works published on this subject, the existence of such possibility in the stock market is still a controversial matter. In economics, the origin of the *random walk hypothesis* can be traced back to the works of Regnault (1863) and Bachelier (1900). These works argue that stock market prices evolve according to a random walk and thus cannot be predicted. This idea, along with concepts proposed by Hayek (1945), has been later incorporated in the *efficient market hypothesis* (MALKIEL; FAMA, 1970), which states, broadly speaking, that the price of an asset fully reflects all available information in the market and, consequently, predicting market movements would be an impossible task to be accomplished. *Technical analysis*, on the other hand, is known as the study of financial market's historical data with the purpose of forecasting future price trends (WANG; CHAN, 2007). It, hence, contradicts the efficient market hypothesis (ROBERTS, 1959) by believing that it is indeed possible to identify trending patterns within the historical prices of an asset in short-term and even in long-term periods.

The use of "network thinking" (WATTS, 2004; BARABÁSI, 2002) when dealing with complex systems in the real world can help to better understand the phenomenon, by analyzing its topological features (MITCHELL, 2006). In vaccination strategies, for instance, the immunization of hubs in the network is more likely to slow the spread of a disease than choosing random individuals to vaccinate (PASTOR-SATORRAS; VESPIGNANI, 2002). A similar type of reasoning can also be applied to other important tasks, such as managing public policies for controlling epidemics (PASTOR-SATORRAS; VESPIGNANI, 2001), mitigating the effects of power failures (MOTTER; LAI, 2002) and viruses spread in computers (WANG; CHEN, 2003). Additionally, there is the concept of *functional cartography*, introduced by Guimera and Amaral (2005), based on observations that metabolic networks in organisms may also present community structure, in which each node plays a different "role" or "sub-role", according to their pattern of intra- and inter-module connections. In this manner, according to this concept, a hub could be classified into three sub-roles: *provincial* (the vast majority of the node's links are within the node's module), *connector* (the node is both a hub in its module and has many links to most other modules) or *kinless* (the node's links are homogeneously distributed among all modules).

Inspired by the concept of functional cartography and the formation of community structures, observed in natural and man-made systems, we present a model which makes use of connector hubs to detect *price trend reversals* in the market, thus allowing the model to detect the starting point of *up* or *down* trends for a stock, as well as triggering a buying or a selling operation accordingly. It starts by, in the *trend detection phase*, mapping the stock price variation ranges into a network and then classifying theses nodes as being more or less likely to indicate an *up* or *down* trend in the stock price, according to the network's community structure. Afterwards, in the *operating phase*, the model propagates these labels to future prices, also triggering buying

or selling operations for the stock, accordingly. Through the identification of hubs connecting the network's communities, the model detects a pattern formation representing what is known in the stock market as a trend reversal, such that the return obtained from each trade can be improved. For evaluating its efficiency, the model is applied to the historical prices of 10 of the most traded stocks from NYSE and from Bovespa Stock Exchange, and the obtained results are encouraging, with the best returns of the proposed model being able to outperform the stock price returns for the same period in 15 out of the 20 considered cases.

The proposed model is nature inspired in two aspects: (1) in the data representation, by mapping the stock prices as a network and making use of the functional cartography concept, and (2) in the data processing, inasmuch as the decision making technique proposed in this work is also inspired by human (animal brain) in such a way that the classification is made according to the pattern formation of the data beyond the physical features. In this case, the community structure of the price variations' network is used to detect the price trend patterns for the stock.

Regarding the organization of this chapter, besides this introduction, we discuss, in section 6.2, the aspects involving the motivation for this study. In section 6.3, we provide a detailed description of the model, showing how it generates the network and the labels from the input data and also the algorithms used in its tasks, both for the trend detection and the operating phases. We also introduce, in this section, the database used for obtaining the experimental results. In section 6.4, we present the results obtained by applying the model to real financial time series from NYSE and Bovespa Stock Exchange. In section 6.5, we conclude the chapter with some final remarks.

6.2 Motivation

In recent years, the continuous advances in the area of *artificial intelligence* contributed to increasing the interest in using different approaches, such as *machine learning*, to treat the challenging problem of predicting financial data. In the work of Huang, Nakamori and Wang (2005), the authors have compared the performances of Support Vector Machines (VAPNIK, 2000) and other traditional classifiers on predicting the tendency of the NIKKEI index, obtaining favorable results by SVM. In the work of Adebiyi, Adewumi and Ayo (2014), the authors have compared the performance of the statistical model ARIMA and artificial neural networks on forecasting the future direction of stock prices of NYSE, obtaining better results by the latter. Another interesting approach was made by Lee and Jo (1999), in which an expert system based on a knowledge base comprising candlestick's technical analyses was developed to predict future price movements for stocks, with the experimental results achieving an average hit ratio of 72% tested with Korean stock market data.

A network-based machine learning technique offers the advantage to not only analyze physical features of the input data (e.g., distance or distribution), as in traditional machine learning

techniques, but to also consider the pattern formation in their topological properties (COLLIRI *et al.*, 2018; SILVA; ZHAO, 2012a). Besides, as this is intrinsically a graphical approach, it has the convenience of being *interpretable*, which is not the case with other machine learning approaches, such as neural networks (GURESEN; KAYAKUTLU; DAIM, 2011). In the work of Cao *et al.* (2019), a network was built from S&P 500, NASDAQ and DJIA indexes data and the topological measures are extracted and used as attributes for predicting the next-day patterns, obtaining an accuracy rate of over 70%. Another application in this sense was made by Anghinoni *et al.* (2019), which forecasts trends in stochastic time series from the Bovespa index through *unsupervised learning* techniques, based on community detection and network walk observations, and obtained an accuracy rate of over 90% in the predictions, outperforming traditional classifiers such as naive Bayes (RISH, 2001), Decision Tree (SAFAVIN; LANDGREBE, 1991) and Multilayer Perceptron (HINTON, 1989).

The main motivation behind the development of such methods and techniques, aiming to predict future prices in the stock market, lies in the possibility of using them to correctly determine the exact time for buying and selling a stock. Although the determination of these "timings", when it comes to stock market decisions, may be subject to cultural differences among the investors (JI; ZHANG; GUO, 2008), it is possible to affirm that they all share the same goal, which is to increase the return obtained from each trading operation. Within this context, the investor who makes use of the model presented in this study has the benefit of not only being able to detect price trends, but also of being advised about when will be the most adequate moment to buy or to sell a stock, in order to improve his returns.

6.3 Materials and Methods

The methodology used in this study is summarized below. In subsection 6.3.1, we provide an overview of the proposed model. In subsection 6.3.2, we demonstrate how the model detects the price oscillation trends from the training data, as well as how these trends are represented in the form of a network. In subsection 6.3.3, we explain the model's operating phase, in which buying or selling operations are performed for an asset, according to the position of its current price in the previously generated network. At the end of the section, in subsection 6.3.4, we introduce the price histories of stocks from NYSE and Bovespa, used for evaluating the model's performance.

6.3.1 Model Overview

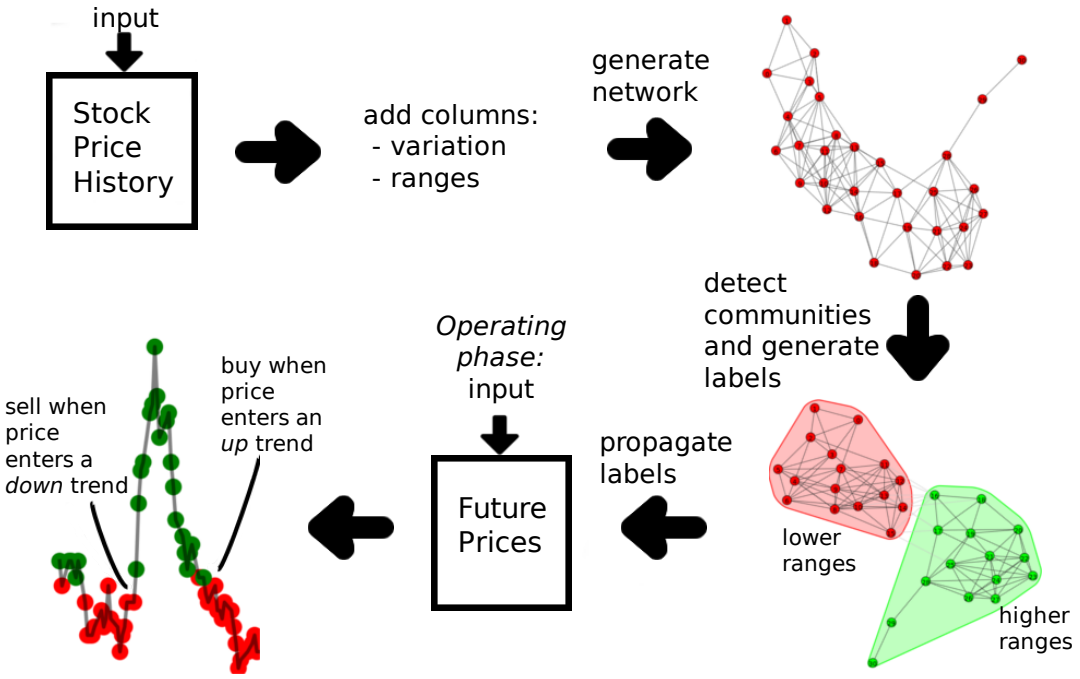
In *supervised learning* models, initially we have an input dataset comprising an array of attributes X_{train} and an array of labels Y_{train}, and the model then has to learn from these two arrays to predict (or classify) the instances from the array X_{test} in the testing phase. In our case, X_{train} is provided and it comprises the price history of a stock, while Y_{train}, containing the labels, is

not provided and must comprise the respective stock price trends for each closing price in X_{train}. So the first task of the model is to generate the labels Y_{train} based on the data provided in X_{train}, i.e., the model must detect the price trends for the stock according to its historical prices. This is achieved in the *trend detection phase*, by first adding two columns (or attributes) in X_{train}: the price variation within a v-days sliding window and its respective discretized version, consisting of n price variation ranges. Each of these ranges is then mapped as a node in a network, and edges are created between each pair of variation ranges if they ever appeared consecutively in X_{train}. Next, a community detection algorithm is ran, and the labels Y_{train}, i.e., the stock price trends, will be the respective community to which each node belongs. Nodes from the lower ranges' community represent a *down* trend, while nodes from the higher ranges' community represent an *up* trend. Later, in the testing phase (here named as the *operating phase*), these labels are used to predict the future stock prices, triggering possible *buying* or *selling* operations for the stock, whenever the current price is represented by a connector hub from an upper or lower community in the network, respectively.

Note that it is possible to exist more than two communities in the network and, in this case, the model takes into consideration only the one with lower ranges and the one with upper ranges, for triggering buying or selling operations. A summary of the model's processing is illustrated in Figure 18. It is also worth noting that the model only triggers a buying or selling operation when it detects what is known as a *trend reversal* pattern for the stock price, i.e., when the prices movement changes from a *down* trend to an *up* trend (triggering a *buying* operation) or, alternately, when the prices movement changes from an *up* trend to a *down* trend (hence triggering a *selling* operation).

In technical analysis, there is the known concept among investors of *support* and *resistance*, when analyzing chart patterns (CHIANG *et al.*, 2016; OSLER, 2000). The main idea is that these two indicators tend to act as "barriers", preventing the price of an asset from going up or down beyond these levels. Additionally, given the market dynamics, in the medium and long terms, a previous resistance level may become a support level, and vice-versa, when the price goes up or down beyond that level. The rationale behind the proposed model has some similarities with this type of analysis, with the main difference that, instead of measuring the support and resistance levels in terms of prices, the model determines them in terms of a sliding-window price variation, with the connector hubs of the network acting as some sort of support and resistance indicators, in this case. During the operating phase, the model assigns a label to the asset's current price according to the respective node that its variation range occupies in the network, which is built during the trend detection phase. Since the trend reversal patterns identified and learned by the model from this stock's historical price data in the training phase continues to reoccur in the stock's future prices, therefore, it is used for predicting purposes in the operating phase. For this reason, the model can get good results.

Figure 18 – Overview of the model. From the input stock price history, two columns are added: the price variation within a v-days sliding window and its respective discretized values, in the form of n variation ranges. Then, a network is generated, where each node represents a variation range and the edges denote whether the ranges ever appeared consecutively in the stock price history, pairwise. The community detection algorithm has the role of labeling the ranges into an *up*, *down* or one of the possible in-between trends for the stock price (in this example there are only two possibilities: *up* or *down*, because there are only two communities). In the operating phase, the labels are propagated to future prices, possibly triggering a buying or a selling operation accordingly, when it identifies a *trend reversal* pattern in the price series.



Source: Elaborated by the author.

6.3.2 Trend Detection Phase

For conducting our experiments, we split each stock price history dataset X into two subdatasets: X_{train} and X_{test}, according to a time range parameter γ. The other two parameters required by the model are: v (measured in days, used as a sliding window for generating the price variations) and l (also measured in terms of days, is the length of the ranges for the discretization of the price variations). The trend detection phase can be summarized in the following steps below, in that order:

1. start with a dataset X, containing attributes *date* and *closing price* for the stock, and insert column *var-v* showing the discretized closing price variations within a v-days sliding window;

2. split dataset X into two parts according to the previously stipulated number of training days γ, such that we have $X_{train} = X_{[:\gamma]}$ and $X_{test} = X_{[\gamma+1:]}$;

3. sort X_{train} in ascending order by attribute *var-v*;

4. stipulate n variation ranges r of the same length l and categorize each variation value in $X_{train}^{var\text{-}v}$ into its respective range r, generating attribute R in X_{train};

5. revert X_{train} to its original order by *date* attribute, and generate adjacency matrix A, representing the constructed network $\mathcal{G}$, such that each range r in X^R may become a node in a network (Algorithm 3);

6. detect communities in $\mathcal{G}$.

Algorithm 3 – Adjacency matrix A generation.

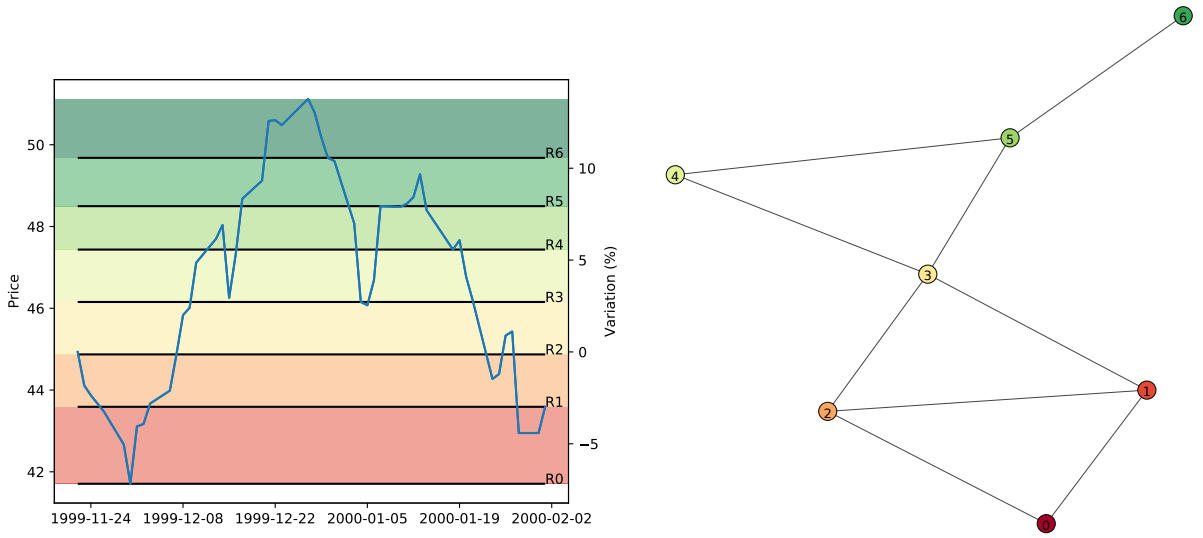
Input: price history with variation ranges X
Output: adjacency matrix A

```
1:  procedure GET ADJACENCY MATRIX A(X)
2:      n ← number of ranges stipulated for the variations in X
3:      A ← zero matrix (n x n)
4:      t ← length of X
5:      for i = 1 to t do
6:          r1 ← X^R_{i-1}
7:          r2 ← X^R_i
8:          if r1 ≠ r2 then
9:              A_{r1,r2} = 1
10:             A_{r2,r1} = 1
11:         end if
12:     end for
13: return A
14: end procedure
```

The rules for generating the adjacency matrix A are depicted in Algorithm 3, in which X, in this case, stands for X_{train}. According to these rules, two vertices v_1 and v_2 in the network $\mathcal{G}$, which represent ranges r_1 and r_2 in X^R, will be connected by a link only if they are immediately next to each other at any point in X^R. Since the values in X^R are ordered by dates in ascending order and given that they indicate how distant the current price is from the price of v days ago, then the position of the nodes in the network will be affected by the overall stock price oscillation, with subsequent and similar ranges tending to be connected and closer to each other (Figure 19). This pattern will later be recognized by the community detection algorithm, which will group the nodes according to the "momentum" they represent in the current stock price oscillation trend (so to speak). It is worth noting that the Algorithm 3 was originally applied as the first step in a network-based model for detecting periodicity in time series, such as the ones from meteorological data (FERREIRA; ZHAO, 2014), and here we are taking advantage of its rationale by applying it in the stock market context. During the operating phase, depending on the community to which the node of the current price variation range belongs, the model then infers whether it is an indication that the stock price is currently in an *up*, *down* or maybe in one of its *in-between* trends, when compared to its price from v days ago. If the community of the current price is different from the community of the previous day price, then a trend reversal may be detected, and a buying or selling operation is triggered for the stock.

The trend detection phase is illustrated in Figure 20, where we have a stock EXMP3 with a sliding window v of 30 days for the price variation attribute. In this example, the training

Figure 19 – Illustration showing how the price variation ranges are defined and later mapped as nodes in the network. For this example, we use the first 50 days of stock GE, from NYSE, and, for the sake of simplification, the daily price variations are calculated based on the initial price, instead of using a sliding window. (a) After being sorted in ascending order, the price variations are split into 7 equal parts (the square root of 50), thus delimiting the ranges R0 to R6. (b) Each range becomes a node in the network, and two nodes are connected if they ever appeared consecutively in the stock price time series. Note that node 2 is connected to node 0 due to the price drop happened on 2000-01-28.



(a) Price variation ranges

(b) Network resulted

Source: Elaborated by the author.

process resulted in 31 ranges r (labels 0 until 30) for categorizing the 30-days price variations, and the community detection algorithm returned 2 communities (or *clusters*) to be used as labels for the price trends. It is also important to observe in this figure that, here, the indexes and range values start at 0 instead of at 1, and so we proceed when writing the algorithms. Therefore, in case one wants to replicate our model and uses different indexing values, then he has to adapt the code indexes correspondingly.

Regarding the processing time, all processes required by the model in the trend detection phase have a linear time, hence with a time complexity of $O(n)$, with the only exceptions being the sorting operations, which are of type $O(n \log n)$, where n may represent the size of the input dataset X, of subdataset X_{train} or of the network $\mathscr{G}$, depending on the task.

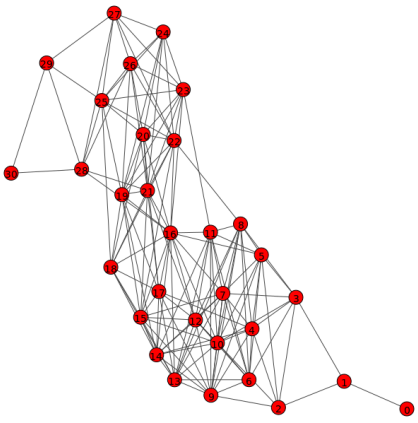
6.3.3 Operating Phase

After having detected the trending patterns for the stock and properly assigned them as labels for the subdataset X_{train}, we then proceed to the testing phase, in which we propagate these labels to the future prices of the stock in X_{test}. Here, this phase is named as the *operating*

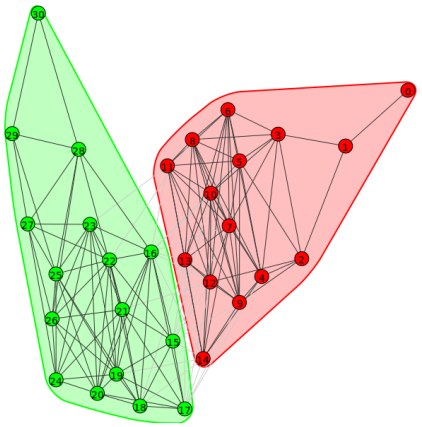
Figure 20 – An example of the trend detection phase for the stock EXMP3. (a) Fragment from the subdataset X_{train}. The columns *var-30*, *R* and *cm* represent the discretized 30-days closing price variations, the ranges used for categorizing the variations and the community to which each range *r* belongs (added later from $\mathscr{G}$), respectively. (b) The network $\mathscr{G}$ resulted from the application of Algorithm 3, to generate the adjacency matrix *A*. (c) The communities identified in the network $\mathscr{G}$. Note that, in this case, the green community represents the higher ranges (from 15 to 30) and hence indicates an *up* trend, while the red community contains the lower ranges (from 0 to 14) and hence indicates a *down* trend.

date	ticker	close	var-30	R	cm
1998-10-02	EXMP3	1.46	-35	1	0
1998-10-05	EXMP3	1.39	-34	1	0
1998-10-06	EXMP3	1.46	-27	1	0
1998-10-07	EXMP3	1.44	-26	1	0
1998-10-08	EXMP3	1.45	-20	2	0
1998-10-09	EXMP3	1.55	-2	12	0
1998-10-13	EXMP3	1.64	5	17	1
1998-10-14	EXMP3	1.63	9	19	1
1998-10-15	EXMP3	1.73	5	17	1
1998-10-16	EXMP3	1.74	-1	13	0
1998-10-19	EXMP3	1.78	7	18	1
1998-10-20	EXMP3	1.91	20	25	1
1998-10-21	EXMP3	2.09	38	29	1
1998-10-22	EXMP3	2.10	42	29	1
1998-10-23	EXMP3	1.96	60	30	1
1998-10-26	EXMP3	1.83	29	28	1
1998-10-27	EXMP3	1.80	13	21	1
1998-10-28	EXMP3	1.83	-3	12	0
1998-10-29	EXMP3	1.74	3	16	1
1998-10-30	EXMP3	1.88	30	28	1

(a) X_{train}



(b) Network



(c) Communities

Source: Elaborated by the author.

phase, since depending on the labels assigned to the current prices, an operation of buying or selling may also be triggered for the stock. We stipulate some rules for buying or selling the stock according to which node, in the previously generated network $\mathscr{G}$, the current stock price variation range is. There is only one parameter required by the model in this phase – the number of connector hubs *h* taken into account to identify possible reversals in the price trend. The operating phase of the model can be summarized in the following steps below, in that order:

1. add columns *R* and *cm* in X_{test} dataset by using the same conversion values between attributes *var-v*, *R* and *cm* (community) obtained from the trend detection phase;
   - in case a value from $var\text{-}v_i^{test}$ is higher than $max(var\text{-}v^{training})$, then its r_i value should be $max(R^{training})$;
   - in case a value from $var\text{-}v_i^{test}$ is lower than $min(var\text{-}v^{training})$, then its r_i value should be 0;
2. perform buying and selling operations for the stock according to the rules described in Algorithm 4; and
3. in case there is still an open position after processing the last row of X_{test} with Algorithm 4, this position then must be "closed" (sold) and its respective return should also be computed in the overall model's return for the stock.

Algorithm 4 – Perform buy and sell operations.

Input: variation ranges network G, number of hubs considered h, prices list to operate X
Output: list with trading operations performed in X

```
1:  procedure GET HUBS(G, h, cm1)
2:      if cm1 = 0 then
3:          cm2 ← 1
4:      else
5:          cm2 ← cm1 − 1
6:      end if
7:      nodes ← []
8:      for v in G^vertices do
9:          if v in G^clusters_cm1 then
10:             nodes ← append v
11:             v^k ← 0
12:             for j in v^neighbors do
13:                 if j in G^clusters_cm2 then
14:                     v^k += 1
15:                 end if
16:             end for
17:         end if
18:     end for
19:     nodes ← sort descending by v^k
20:     return nodes_[:h]
21: end procedure
22: procedure TRADE(X, G, h)
23:     n ← number of communities in G
24:     downhubs ← GetHubs(G, h, 0)
25:     uphubs ← GetHubs(G, h, n − 1)
26:     t ← length of X
27:     list ← []
28:     flat ← True
29:     for i = 0 to t do
30:         if flat then
31:             if X_i^R in uphubs then
32:                 list ← add dict with buying info (price, date)
33:                 flat ← False
34:             end if
35:         else
36:             if X_i^R in downhubs then
37:                 list ← add dict with selling info (price, date, return, length)
38:                 flat ← True
39:             end if
40:         end if
41:     end for
42:     return list
43: end procedure
```

The rules for buying and selling the stock during the operating phase are described in Algorithm 4, in which X, in this case, stands for X_{test}. The first step of the *Trade* procedure, in Algorithm 4, is to identify the h *downhubs* and the h *uphubs* in the network $\mathscr{G}$, through the *GetHubs* procedure. The *downhubs* are the h vertices within the lowest community (label 0) with most links to vertices from its immediately higher community (label 1). These connector hubs thus indicate the beginning of a *down* trend for the stock price, according to our model. On the other hand, the *uphubs* are the h vertices within the highest community ($n-1$) with most links to vertices from its immediately lower community ($n-2$), where n is the number of communities in $\mathscr{G}$. These connector hubs thus indicate the beginning of an *up* trend for the stock price, according to our model.

In the second step of the *Trade* procedure, in Algorithm 4, the model may execute a

buying or a selling order according to the price's variation range (represented by a node in the trade data network), as well as its current position in the stock. The model performs a buying operation when it is currently *flat*, i.e. it is not already positioned in the stock, and the variation range (X_i^R) of the current price (X_i^{close}) is among one of the *uphubs*. Conversely, it performs a selling operation for the stock when it is currently positioned in the stock and the variation range (X_i^R) of the current price (X_i^{close}) is among one of the *downhubs*. When the current price variation range reaches one of the h "entry" connector hubs from the higher ranges' community, then it is a sign of *trend reversal* with the stock price entering in an *up* trend, so the model advises to buy the stock at that moment. On the other hand, when it reaches one of the h "entry" connector hubs from the lower ranges' community, then it is also a sign of *trend reversal*, this time with the stock price entering in a *down* trend, so the model advises to sell the stock at that moment. Taking Figure 20c as an example, in this case, when $h = 3$, the *downhubs* in the network would be the connector hubs 11, 13 and 14, while the *uphubs* would be the connector hubs 15, 16 and 17.

Note that, by proceeding this way, we are here also assuming that the price oscillation patterns identified for the stock during the trend detection phase will still remain during the operating phase, i.e., after the γ days previously stipulated for splitting the dataset into X_{train} and X_{test}. It is also important to emphasize that the model only buys the stock when it is "flat", i.e., it is not currently already positioned in the stock (hence there are no "averaging down" strategies here) and, conversely, it only sells the stock when it is currently positioned in it (hence "short selling" strategies are not allowed here either).

Regarding the time complexity of the model during the operating phase, there are two tasks with linear time ($O(n)$), where n represents the size of subdataset X_{test}, and one task with logarithmic time ($O(n \log n)$), which is the *GetHubs* procedure from Algorithm 4, where n then represents the size of network $\mathscr{G}$.

6.3.4 Database

In the stock market, the returns of trading operations depend on specific timing strategies for buying and selling the stock. The proposed model defines these timings strictly according to the trending patterns detected through a network-based technique, for each stock. Hence, in order to evaluate the model, we compare its performance, in terms of financial return, with the one provided by the stock price during the same period of the tests, i.e., with the returns obtained by the "buy and hold" strategy. The database we build for the tests comprises the daily closing price histories of 10 of the most traded stocks from NYSE and 10 of the most traded stocks from the Brazilian Stock Exchange (Bovespa). These stocks are listed in Table 12, along with their respective time ranges considered. Eventual stock splits and consolidations are also taken into account and appropriately reflected into the price histories for generating the model's input data.

In order to obtain the results, we run the model using different values for the sliding window parameter v and for the number of hubs parameter h, when processing all stocks in

Table 12 – Time series used as database, obtained from NYSE and Bovespa Stock Exchange

	Ticker	Company	d_0	d_f	Days
NYSE	BAC	Bank of America	1999-11-22	2019-11-20	7303
	EFX	Equifax	1999-11-22	2019-11-20	7303
	F	Ford Motor	1999-11-22	2019-11-20	7303
	GE	General Electric	1999-11-22	2019-11-20	7303
	JPM	JPMorgan	1999-11-22	2019-11-20	7303
	M	Macy's	1999-11-22	2019-11-21	7304
	SCHW	Charles Schwab	1999-11-22	2019-11-21	7304
	PFE	Pfizer	1999-11-22	2019-11-20	7303
	T	AT&T	1999-11-22	2019-11-20	7303
	X	United States Steel	1999-11-22	2019-11-20	7303
Bovespa	ABEV3	Ambev	1999-09-17	2019-11-19	7368
	BBAS3	Banco do Brasil	1998-03-16	2019-11-19	7918
	BRFS3	Brasil Foods	2009-12-10	2019-11-19	3631
	BRKM5	Braskem	2002-09-02	2019-11-19	6287
	CIEL3	Cielo	2009-12-18	2019-11-19	3623
	CSNA3	CSN	1998-03-16	2019-11-19	7918
	ITSA4	Itaúsa	1998-03-16	2019-11-19	7918
	JBSS3	JBS	2007-03-29	2019-11-19	4618
	PETR4	Petrobras	1998-03-16	2019-11-19	7918
	VALE3	Vale	1998-03-16	2019-11-19	7918

Source: Research data.

the database, such that $v \in (5, 10, 30, 60, 120, 180)$, and $h \in [1, 10]$. The number of days γ for splitting X into X_{train} and X_{test} is fixed as 1000, which represents almost 4 years of trading days. The values of *var-v* are discretized through a simple rounding function, for the sake of converting them into integers, and the length l of each range r is defined as the square root of the length of X_{train}, which results in $n = l$, for all cases. For detecting communities in the network, we make use of the *fast greedy* algorithm (NEWMAN, 2004).

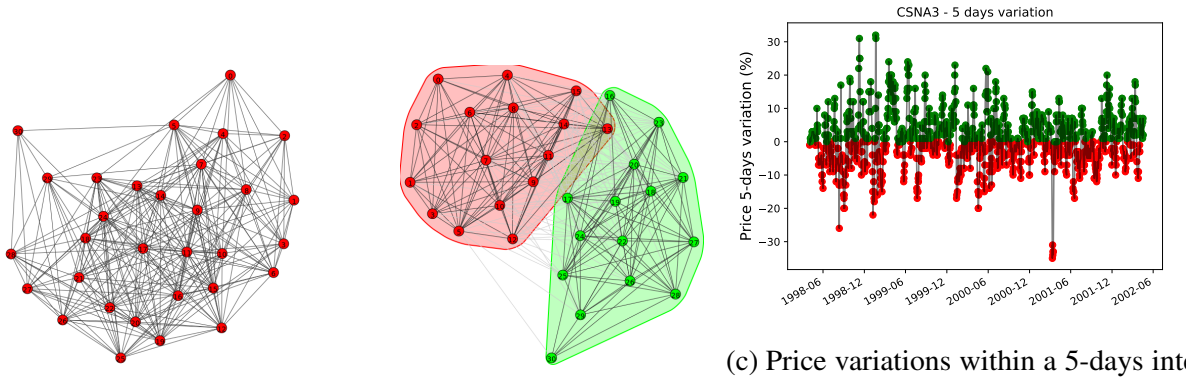
6.4 Results and Discussion

In this section, we present the obtained results when applying the proposed model on historical data from NYSE and Bovespa stocks.

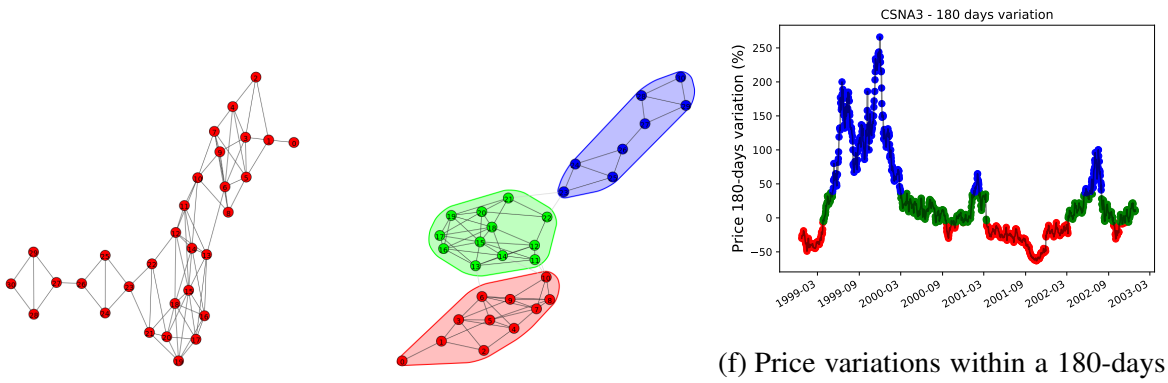
6.4.1 Generated Networks

We start by showing, in Figure 21, the networks, detected communities and the evolution of the values in $X_{train}^{var\text{-}v}$ obtained from the trend detection phase for the stock CSNA3, for $v = 5$ and $v = 180$, respectively. We present the plots for the same stock with these specific values of v with the aim of demonstrating how the parameter v may affect the training process' output.

Figure 21 – Output from the trend detection phase for stock CSNA3, when the sliding window $v = 5$ days (above) and $v = 180$ days (below). Left column: Network $\mathscr{G}$. Middle column: Communities detected. Right column: Price variation ranges evolution. The colors in (c) and (f) denote the communities to which each variation range belongs. Note that, when comparing the price variations, in the third column, the ranges for $v = 180$ follow much longer trends and also reach higher values, both on the positive and on the negative sides.



(a) Network (b) Communities (c) Price variations within a 5-days interval



(d) Network (e) Communities (f) Price variations within a 180-days interval

Source: Elaborated by the author.

For lower values of v, the network $\mathscr{G}$ tends to have a higher average degree (Figure 21a) and the price variations tend to present a lower autocorrelation (Figure 21c). In opposition, higher values of v tend to result in a network $\mathscr{G}$ with lower average degree (Figure 21d) and to incur a higher autocorrelation for the price variations (Figure 21f). A lower v also tends to increase the number of trades performed by the model during the operating phase, as the nodes get more interconnected and so there are more chances of the price variation ranges to reach the *uphubs* and the *downhubs* in the network. Another consequence of having lower values for the sliding window v is that it makes the average length of the trades shorter (Figure 22), for the same reason explained above. Therefore, we can say that if one is planning to make use of the model with the goal of optimizing short-term trading operations, then he is advised to choose lower values of v. While if one plans to optimize long-term trading operations, then a higher value of v is indicated. The parameter h may also affect the number of operations performed by the model, since lower

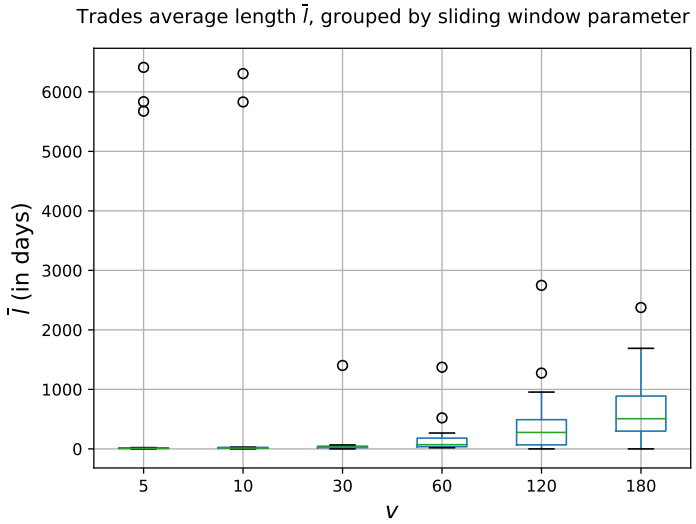


Figure 22 – Box plots of the average length of trades $\bar{l}$ (in days) according to sliding window parameter v (also in days) in the operating phase for both NYSE and Bovespa databases, considering all values of parameter h. Usually, as lower the value chosen for v, the shorter is the expected length of the trading operations performed by the model. The *outliers* shown in this figure occur for small values of h, such as 1 or 2.

Source: Elaborated by the author.

values of h, such as 1 or 2, tend to lessen its probability of triggering new operations, both for *buying* and for *selling*.

6.4.2 Obtained Returns

The returns achieved by the model in the operating phase, considering all values of parameters v and h, both for NYSE and Bovespa databases, are displayed in Figure 23 and summarized in Table 13, grouped by stock. The first column in Table 13, r_{stock}, shows the stock return in terms of its price variation within the period of the operating phase, i.e., the time range provided in X_{test}. The column $\bar{r}_{model}$ shows the average return achieved by the model, considering all values of v and h. The columns d_0^* and d_f^* display the initial and final dates taken into consideration by the model for achieving the best return r^*_{model} for each stock. The value of d_0^* may some times differ from the operating phase's initial date due to the sliding window parameter v^*.

For the NYSE database, the model's average return $\bar{r}_{model}$ is able to surpass the stock price return r_{stock} in the period for 6 out of the 10 cases considered, with the exceptions being EFX, JPM, SCHW and T. As for the model's best return r^*_{model}, it is able to outperform the stock price return in 8 out of the 10 cases considered, with the 2 exceptions being EFX and SCHW. The best case, in terms of return optimization, is achieved for stock F, in which the stock price return for the period is -33.46% while the model's best return is 673.3% in the same period. This return rate for F is achieved through 131 trading operations, with an average length of 26 days

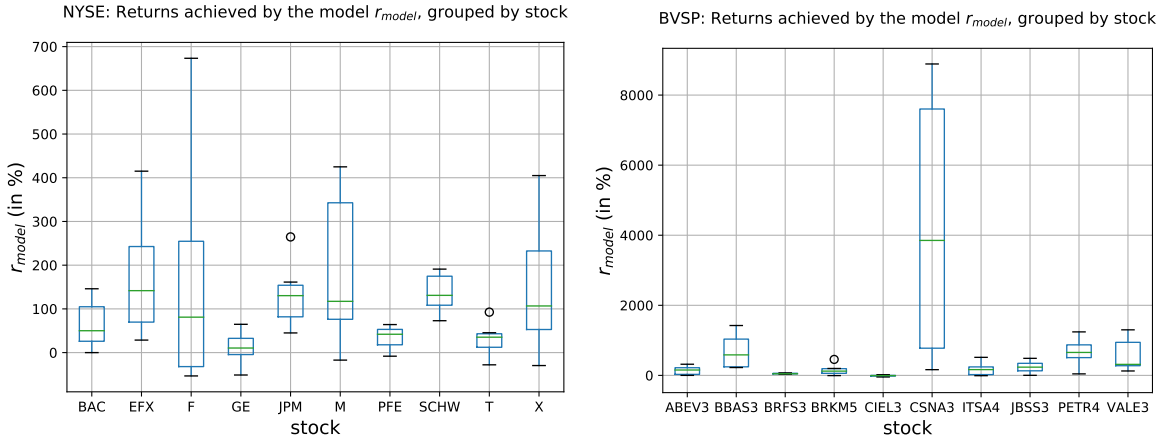


Figure 23 – Box plots of the returns achieved by the model r_{model} (in %) in the operating phase on the NYSE and Bovespa databases, grouped by stock, considering all values of parameters v and h. For NYSE, the highest return is obtained for stock F, of 673%. For Bovespa, the returns obtained for stock CSNA3 really stand out from the others, reaching more than 8000%.

Source: Elaborated by the author.

Table 13 – Main results obtained from the tests: the returns r are expressed in percentages (%). The mark * indicates the best result for each stock. The columns n^*, $\overline{l}^*$ and $\overline{r}^*$ show the total number of trades, average length per trade and the geometric mean return per trade, respectively.

		Model's average results		**Model's best results**					**Trades info**		
	stock	r_{stock}	$\overline{r}_{model}$	r^*_{model}	d^*_0	d^*_f	v^*	h^*	n^*	$\overline{l}^*$	$\overline{r}^*$
NYSE	BAC	-56.03	-5.46	146.04	2003-11-21	2019-11-20	5	6	193	21	0.47
	EFX	478.34	152.10	415.04	2003-12-30	2019-11-20	30	2	4	1402	50.65
	F	-33.46	3.55	673.30	2003-12-01	2019-11-20	10	3	131	26	1.57
	GE	-60.55	-15.60	64.72	2004-05-10	2019-11-20	120	2	14	275	3.63
	JPM	263.52	54.19	264.65	2003-12-01	2019-11-20	10	1	1	5831	264.65
	M	-35.74	135.43	424.95	2004-08-05	2019-11-21	180	2	3	1191	73.80
	SCHW	308.77	67.70	183.73	2003-12-30	2019-11-21	30	1	61	66	1.72
	PFE	17.06	18.13	64.12	2004-08-05	2019-11-20	180	2	12	253	4.21
	T	47.07	4.44	92.59	2004-08-05	2019-11-20	180	10	4	1222	17.80
	X	-64.00	57.32	404.96	2003-12-30	2019-11-20	30	3	155	19	1.05
Bovespa	ABEV3	360.16	71.14	318.10	2004-04-20	2019-11-19	5	2	1	5676	318.10
	BBAS3	906.37	394.20	1423.18	2002-04-29	2019-11-19	10	1	1	6307	1423.18
	BRFS3	-24.48	19.85	71.10	2014-01-13	2019-11-19	10	1	44	25	1.23
	BRKM5	87.82	84.67	455.75	2006-09-13	2019-11-19	5	2	185	15	0.93
	CIEL3	-77.06	-32.67	19.47	2014-01-14	2019-11-19	5	8	78	4	0.23
	CSNA3	451.51	2541.69	8887.88	2002-05-24	2019-11-19	30	10	120	28	3.82
	ITSA4	602.62	133.33	514.82	2002-12-20	2019-11-19	180	2	4	837	57.47
	JBSS3	390.74	179.46	354.17	2011-07-14	2019-11-19	60	1	5	521	35.35
	PETR4	313.28	382.56	1241.27	2002-04-15	2019-11-19	5	10	397	9	0.66
	VALE3	660.68	399.17	1300.08	2002-06-03	2019-11-19	10	2	151	28	1.76

Source: Research data.

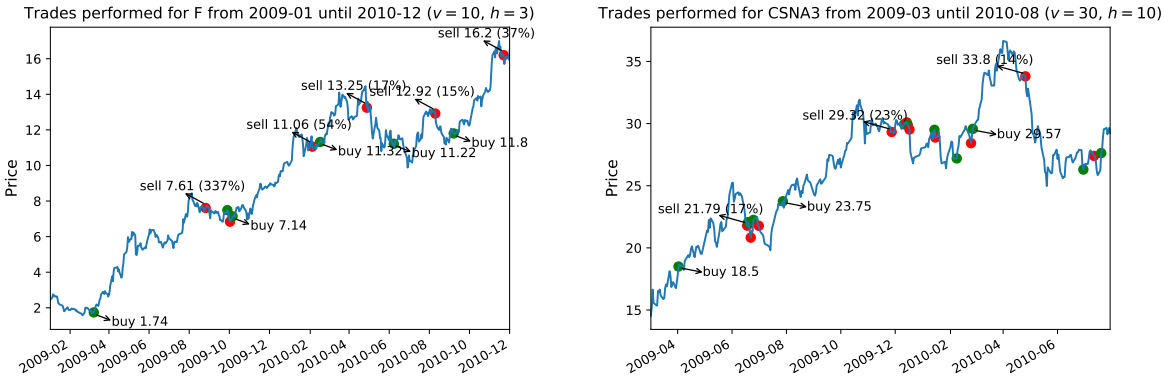


Figure 24 – Examples of trades performed by the model for stocks F and CSNA3 during the operating phase, using their respective optimal parameters v and h. *Buying* operations are indicated by the green color and *selling* operations by the red color. The values in parentheses show the return achieved in each trade. In these two cases, the model is more able to correctly detect a future *up* or *down* trend for the price, hence the higher overall returns obtained for these stocks when compared to others.

Source: Elaborated by the author.

and a geometric mean return of 1.57% per operation. For the stock JPM, the model's best return is only 1% higher than the stock price's return for the period, and this return is achieved through 1 operation alone. Looking at the trades list of that stock, for these parameters, we noted that the model in fact just performed the equivalent to a "buy and hold" strategy in this case, buying the stock on 2003-12-02 and having being forced to sell it just at the end of the simulation's period in order to compute its return.

The most surprising item in the results for Bovespa database is the high rates of return obtained for the stock CSNA3. Since these values really stand out from the others, we have double checked these numbers in order to be sure that the model's calculations and the input data are both correct. The average return from the model $\bar{r}_{model}$ for CSNA3 (2,541%) – which is the arithmetic mean of the returns obtained for all values of parameters v and h – is already much higher than the stock return r_{stock} for the same period (451%). The best return obtained by the model r^*_{model} (8,887%) though, when $v = 30$ and $h = 10$, is even higher. This high return is achieved from a total of 120 trades, with an average length of 28 days and a geometric mean return of 3.82% per trade. Overall, the average return from the model is able to outperform the stock return in 4 out of 10 cases: BRFS3, CIEL3, CSNA3 and PETR4. As for the model's best return, it is able to outperform the stock price return in 7 out of 10 cases, with the 3 exceptions being ABEV3, ITSA4 and JBSS3. For ABEV3 and BBAS3, the model achieved its best return with only 1 trade, being that in the case of BBAS3 it was nevertheless able to outperform the stock price return in the same period by a high margin (1,423% versus 906%). Looking at this individual trade in the stock's trades list for these parameters ($v = 10$ and $h = 1$), we observe that the model waited for the stock initial price to decrease before opting for buying it, on 2002-08-12.

Table 14 – Annualized Sharpe Ratio (SR) obtained by a "buy and hold" strategy and the one from the proposed model's best returns, for the NYSE and Bovespa stocks in the database.

	Buy & Hold	**Model**	
	SR	SR	t-stats
NYSE	0.17	2.14	8.46
Bovespa	0.56	1.99	7.33

Source: Research data.

To help us understand how the model is able to achieve such higher returns for stocks F, from NYSE, and CSNA3, from Bovespa, we generate Figure 24, in which it is possible to see some examples of the operations performed by the model when using the respective optimal values of the parameters h and v for these stocks. The first trading operation shown in this figure, for stock F, is the most profitable of all, achieving a return of 337% (by buying it at 1.74 on 2009-03-09 and selling it at 7.61 on 2009-08-26). As it can be noted from the remaining operations shown for this stock in the figure, the model is very successful at detecting the price trending patterns during this period, both when buying and selling the stock. It is worth remembering that, evidently, during the operating phase only the historical and current prices are acknowledged by the model, and not the future prices. Hence, we can also say that the model is accurately predicting the future prices for the stock during this period, thus the high return rates achieved in these trading operations. The same observations can also be made for the stock CSNA3 during the period shown in Figure 24.

In Table 14, we make a comparison between the annualized Sharpe ratio of a "buy and hold" strategy and the one from the proposed model's best returns, for the NYSE and Bovespa stocks in the database. The Sharpe ratio (SHARPE, 1966) is the average return earned in excess of the risk-free rate over the strategy volatility or standard deviation, and it is widely used in finance. Generally, the greater the value of the Sharpe ratio, the more attractive the risk-adjusted return. In this case, we assume that the risk-free rate is equal to zero. The statistical significance of the mean obtained returns is given by t-statistic = Sharpe Ratio $\times\sqrt{\text{number of years}}$. As it is shown in Table 14, the strategy from the proposed model's best returns outperforms the buy-and-hold one for both NYSE and Bovespa stocks in the considered periods.

6.5 Chapter Remarks

In this chapter, we introduced a network-based model to detect price trends and also to automate decision-making processes in stock market trading operations in order to optimize the returns. The model starts by mapping the stock's historical price variation ranges into a network and then generates the trend labels based on its topological structure. These labels, which represent the price's *up* and *down* trending patterns identified in this phase, are later propagated to the future prices, and the network's connector hubs are used as indicators of price

trend reversals, triggering operations for buying or selling the stock accordingly. The results obtained through the application of the model to real financial time series, both from NYSE and the Brazilian Stock Exchange, are promising, and future studies should be made with the aim of applying it to larger databases as well as of possibly improving its efficiency, in terms of return rates optimization.

CHAPTER

7

PREDICTING COVID-19 NEW CASES AND DEATHS IN A REGION

In this chapter, we approach the problem of predicting the evolution of new COVID-19 cases and deaths in a region through a network-based regression model. The proposed model works as a multiple regression analysis, and starts by mapping each time series as node in a set of static networks, each one representing a time step t, with the connections between each pair of nodes being based on how similar are their respective variations, at the time t. Then, a community detection algorithm is ran, in each static network, and time series from nodes which have shared a same community in the past are considered by the model to be correlated, and this information is later used for prediction purposes. Hence, we can also say that the model treats the set of generated static networks in the form of a temporal network. The proposed model is evaluated by applying it to predict new COVID-19 cases and deaths for the 27 federal units of Brazil, on a weekly basis, and the obtained results are encouraging, when compared to other similar studies, in terms of mean absolute percentage error.

7.1 Introduction

Since the first human populations started to live in groups, the epidemic diseases have been one of the greatest problems faced by humanity. In the modern era, this problem has been aggravated by two combined factors : (1) the fact that human beings have been living more and more in concentrated urban spaces, and (2) that these great concentrations of people, by their turn, are increasingly more interconnected through faster and more efficient worldwide transportation routes (HARARI, 2014). The most recent example regarding this problem involves the COVID-19, which was officially characterized as a pandemic only around four months after its first cases were reported, in China. The fast dissemination of this disease can be particularly challenging for policy makers around the world, which have to face the difficult task of developing efficient

strategies for controlling the spread of the disease among the population, oftentimes by taking severe measures to restrict the local activities, both socially and economically. Within this context, models which can help on predicting the spread evolution of the disease in each region would surely help the authorities on their resources planning.

Currently, the most common strategy to deal with the COVID-19 pandemic, among policy makers, is to implement isolation policies in the population, through quarantines or lockdowns. These measures have the aim of delaying the peak of the dissemination curve, in order to avoid a suddenly rise on COVID-19 hospitalizations. In this sense, it is more desirable for a given region that its dissemination curve can be more similar to the curves of countries which were less affected by the disease, such as Japan, South Korea and Germany, for instance, than to the curves of countries more severely affected by the disease, such as Italy, Spain and US, for instance. In this study, we take this type of reasoning one step further, by introducing a semi-supervised regression model which detects the correlations between the COVID-19 curves from different regions, and makes use of these detected correlations for predicting future values for new confirmed cases and deaths on a given region. The model starts by mapping the COVID-19 time series to static networks, where each network represents the current correlations between the time series, at each time step t. Each node in these networks represents the time series for a specific region, and the edges are generated according to the detected correlations between each pair of nodes. Afterwards, these static networks are analyzed in the form of a temporal network, in order to predict the evolution of a specific time series, for the period considered.

The novelty of this model consists in the use of a form of multiple regression analysis, in which it does not take into account the predicted time series' own prior curve for making the predictions. Instead, the model considers, as input attributes, only the curves from other time series in the dataset identified as more correlated to the series to be predicted, and whose lengths are greater or equal to the predicted time step t. Hence, this model is indicated for cases when the time series in the dataset present different initial dates, and one wants to investigate whether it is possible to generate predictions based on the correlations detected between the series. We evaluate the proposed model by applying it on preliminary COVID-19 data from different world regions for predicting the future confirmed cases and deaths in the 27 federal units of Brazil. The obtained results are promising, with the model being able to predict, on a weekly basis, the number of new confirmed cases in each state with a median and mean absolute percentage error of 21% and 24%, respectively, and the new confirmed deaths in each state with a median and mean absolute percentage error of 16% and 23%, respectively.

Regarding the organization of this chapter, besides this introduction, we discuss, in section 7.2, the motivations for conceiving this study. In section 7.3, a description of the methodology and the model is provided, showing how the networks are generated from the input data, and how the predictions are made. In section 7.4, we present the results obtained by applying the model to real preliminary COVID-19 spreading data, from world regions and from Brazilian

federal units. In section 7.5, we conclude this chapter with some final remarks.

7.2 Motivation

Networks (or *graphs*) are powerful modeling tools for exploring a dataset in terms of the relations between the data instances, both in a static or in a dynamic way. Mathematically, a network can be defined as a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is a set of nodes and $\mathscr{E}$ is a set of tuples representing the edges between pairs of nodes $(i, j) : i, j \in \mathscr{V}$. A *temporal network* is a specific type of *multilayer network* or *multiplex* (DOMENICO *et al.*, 2013), in the form of $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \mathscr{D})$, in which the additional dimension $\mathscr{D}$ contains an ordered set of temporal indices that represents time (KIVELÄ *et al.*, 2014). Among the phenomena which have already been modeled through temporal networks are brain connectivity (THOMPSON; BRANTEFORS; FRANSSON, 2017), fires events in the Amazon (XUBO *et al.*, 2020), political parties (COLLIRI; ZHAO, 2019; AREF; NEAL, 2020) and corruption scandals (LUNA-PLA; NICOLÁS-CARLOCK, 2020).

Although complex networks is a relative new field of study, there are also already well-known network-based models developed specifically to approach problems regarding the spread of epidemic diseases. These models do not necessarily need to make use of very advanced mathematical calculus since, oftentimes, simple models can help to further understand the transmission of infectious agents within human communities (ANDERSON; ANDERSON; MAY, 1992). The SI, SIS and SIR models (PASTOR-SATORRAS; VESPIGNANI, 2001; BARTHÉLEMY *et al.*, 2005), for instance, allow one to estimate what would be the critical threshold, in terms of the percentage of infected individuals in a population, for an infectious disease to become endemic. These models can also help on determining which immunization strategies are expected to be more effective, according to the topological characteristics of the network formed by the individuals susceptible to the disease (PASTOR-SATORRAS; VESPIGNANI, 2002; DEZSŐ; BARABÁSI, 2002). A very interesting survey on this topic was made by Costa et al. (COSTA *et al.*, 2011).

Some recent studies have applied machine learning techniques to predict the spread of epidemics, on a weekly basis. The study made by Al-qaness *et al.* (2020) forecasts the number of weekly new confirmed cases of influenza in China, based on the previously confirmed cases, by using an improved adaptive neuro-fuzzy inference system (ANFIS), achieving a mean absolute percentage error of 32% and 45%, respectively, on the two datasets analyzed. In the work from Arora, Kumar and Panigrahi (2020), an artificial recurrent neural network (RNN) was applied on a 7-days testing dataset to predict new confirmed cases of COVID-19 in the states of India, obtaining a mean absolute percentage error of around 6% when predicting the weekly new confirmed cases in 4 states.

7.3 Materials and Methods

The research methodology used in this study is summarized below. We start by, in subsection 7.3.1, describing the database used for the model's application and evaluation. Then, in subsection 7.3.2, we explain the proposed prediction model in details, showing how the networks are generated from the input time series dataset. Afterwards, in subsection 7.3.3, we provide a simple application example, for illustrative purposes.

7.3.1 Database

The database used in this study is built from the preliminary data made publicly available by Dong, Du and Gardner (2020). This dataset comprises the daily evolution of COVID-19 confirmed cases and deaths in 261 different regions or countries, from 01-22-2020 until 05-26-2020. We also make use of another dataset (BRASIL.IO, 2020), this one comprising the daily evolution of COVID-19 confirmed cases and deaths in each of the 27 federal units of Brazil, from 02-25-2020 until 05-26-2020. The real values from the later dataset are the ones to be predicted by the model and, for this reason, the future values from each federal unit are suppressed, prior to each prediction, and these values are used later exclusively for performance measuring purposes.

7.3.2 Description of the Time Series Prediction Model

Regression analysis is a statistical method building a mathematical model that best fits the data for the prediction of the output variable (KOSTOPOULOS *et al.*, 2018). In simple regression, there is only one independent variable x that affects the value of the dependent variable y, while in multiple regression there are more than one. In this work, for predicting the evolution of the spread of COVID-19 in a given region, we conceive a *semi-supervised* regression model which predicts future values for a time series in a dataset, i.e., the dependent variable, based on the detected similarities between this time series and the other time series in the same dataset, through a correlations *temporal network*. In this case, we are aiming to detect hidden evolution patterns among groups of time series in the dataset, in order to make use of these patterns for prediction purposes. Therefore, this model can be applied in cases when the time series in the dataset present different initial dates, and one wants to investigate the existence of possible correlation patterns between them and, if that is the case, to estimate future values for a given time series based on these detected correlations.

In *machine learning* applied to time series, initially we have an input dataset comprising n instances (or time series) and m time steps t, in the form of $\mathscr{X} = \{X_1, X_2, X_3, ..., X_n\}$, where each instance i consists of m elements, such that $X_i = (x_{i,t=0}, x_{i,t=1}, x_{i,t=2}, x_{i,t=3}, ..., x_{i,t=m})$, as in

the following 2d array:

$$\begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & \dots & x_{1,m} \\ x_{2,1} & x_{2,2} & x_{2,3} & \dots & x_{2,m} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n,1} & x_{n,2} & x_{n,3} & \dots & x_{n,m} \end{bmatrix} \quad . \tag{7.1}$$

The model proposed in this work starts by bringing all time series in $\mathscr{X}$ to a same starting point $t = 1$. Next, it calculates the variation $\delta X_{i,t}$ for each time series i at each time step t, which is yielded by:

$$\delta X_{i,t} = \frac{X_{i,t} - X_{i,t-1}}{X_{i,t-1}} \quad . \tag{7.2}$$

This provides us with a 2d variations array $\delta\mathscr{X}$, containing n instances (the time series), and a maximum of $m-1$ columns (the time steps) for each time series. Afterwards, each row $\delta\mathscr{X}_t$ in this array is mapped as a network G_t, where each node represents a time series and the edges between each pair of nodes are generated according to the similarities between their variations at the time t. The neighbors connected to each vertice i, in each network G_t, are given by:

$$N(i_t) = \varepsilon\text{-radius}(i_t), \tag{7.3}$$

where ε-radius(i_t) yields a set of instances whose variations $\delta X_{i,t}$ are within the range $[-\varepsilon, +\varepsilon]$. Following, a community detection algorithm is ran in the networks, in order to group the time series with more similar variations, at each time step t. For this end, one can use, for instance, the *fast greedy* algorithm (NEWMAN, 2004), which detects the community structure based on the greedy optimization of the modularity measure, or also the *walktrap* algorithm (PONS; LATAPY, 2005) which, roughly speaking, is based on the idea that short random walks tend to stay in the same community in the network. At this point, we end up with a set of static networks $\mathscr{G} = \{G_{t=2}, G_{t=3}, G_{t=4}, ..., G_{t=m}\}$, with each of them representing the topological space emerged from the current relations between the variations in δX at the time slice t. Hence, we can also say that this set forms the temporal network $\mathscr{G}$, and each element in the set represents a different time slice t of the temporal network $\mathscr{G}$.

Following, a dictionary D_i is created, for each time series i in the dataset, in the form of a list $\mathscr{D} = \{D_1, D_2, D_3, ...D_n\}$. The set of keys K in D_i are given by all instances j in the dataset which have shared the same community with i, at any time step t, i.e., in any of the networks in $\mathscr{G}$. The set of values V in D_i, for any key j, are given by the respective number of times that the instances i and j have shared the same community in $\mathscr{G}$. Mathematically, we have that a dictionary can be defined as:

$$D \subseteq \{(k,v) \mid k \in K \wedge v \in V\} \wedge \forall (q,w) \in D : k = q \rightarrow v = w \quad , \tag{7.4}$$

and, in the proposed model, the set of keys K and the set of values V in D_i are yielded by:

$$D_i^K = \{j \mid j \in G_{t,i}^C\} \wedge t \in [2,m] \wedge j \in [1,n] \wedge j \neq i, \quad \text{and} \tag{7.5}$$

$$D_i^V = u(\{G_t \mid j \in G_{t,i}^C\}) \wedge t \in [2,m] \wedge j \in [1,n] \wedge j \neq i \quad , \tag{7.6}$$

Table 15 – Time series example dataset

	X_1	X_2	X_3	X_4	X_5
1/1/2020	10	2	-	-	-
1/2/2020	15	4	-	-	-
1/3/2020	20	8	3	-	-
1/4/2020	25	16	6	6	-
1/5/2020	30	32	12	9	1
1/6/2020	35	64	24	12	2

Source: Research data.

where $G^C_{t,i}$ provides a set with all instances that share the same community with i in the network G_t.

Finally, the predicted variation for a time series i, at the time step $t > 2$, is equal to the averaged variations of the time series which are in the keys of the dictionary D_i whose length is equal or longer than t, weighted by their respective values in D_i. Thus, the predicted variation $\hat{\delta}X_{i,t}$ is given by:

$$\hat{\delta}X_{i,t} = \begin{cases} \frac{\sum_j D_i^V[k \to j]\ \delta X_{j,t}}{\sum_j D_i^V[k \to j]}, \forall j \in D_i^K \mid j \in G_t \wedge u(j) \geq t, \\ \qquad\qquad \text{if } \{j \mid j \in D_i^K \wedge j \in G_t \wedge u(j) \geq t\} \neq \emptyset \\ \emptyset, \text{ otherwise} \end{cases} \tag{7.7}$$

where $u(j)$ returns the length of array j. In this way, the model is able to predict variations for a time series i, on a time step t, only if at least one of the time series in D_i has a length equal or longer than t. Note that the model, hence, performs a form of multiple regression analysis, in which the prior evolution curve of the time series i (which is the dependent variable) is not taken into account in the predictions, and the independent variables (or predictors) considered in this case are the evolution curves from longer time series in the dataset which are more correlated to i. In this sense, it is worth highlighting the important role played by the ε-radius threshold parameter in the model, used for generating the edges in the networks. Smaller values of ε make the predictions performed by the model more sensitive to local averages in the dataset, with the risk of *overfitting*, while, conversely, higher values of ε result in the model considering broader averages when making the predictions, with the risk of *underfitting*. It is also worth noting that, in Equation 7.7, by weighting the time series variations more correlated to i by the number of times this correlation has occurred, prior to the time t of the prediction, we are here assuming that the time series i tends to preserve these same correlations in the future.

7.3.3 Model's Demonstration Through a Simple Example

In order to illustrate how the proposed model works, we now present its application on a simple dataset, to be used as example. Let us consider a dataset $\mathscr{X}$, comprising five time series:

Table 16 – Daily variations (%) for the example dataset

	X_1	X_2	X_3	X_4	X_5
1st day	50.0	100.0	100.0	50.0	100.0
2nd day	33.3	100.0	100.0	33.3	
3rd day	25.0	100.0	100.0		
4th day	20.0	100.0			
5th day	16.7	100.0			

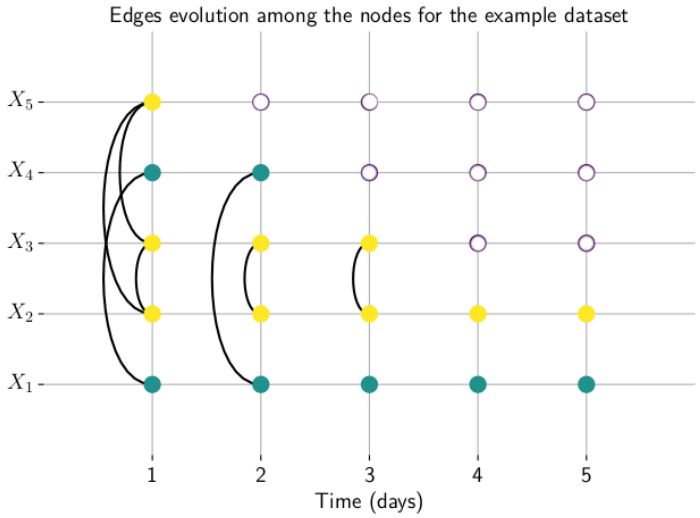
Source: Research data.

X_1, X_2, X_3, X_4 and X_5, with different initial dates and different lengths, as shown in Table 15. Since the model makes use of data from the longer time series in the dataset, in order to perform the predictions, then in this case it will be able to predict future values only for X_3, X_4 and X_5. One can observe, from Table 15, that X_1 and X_4 follow an arithmetic progression, while X_2 and X_3 follow a geometric progression, and X_5 – which is the most recent one, having only 2 observations so far – could either follow an arithmetic or a geometric progression in the future. However, let us suppose that, in the current context, we are not aware of these evolution patterns for any of these time series, and hence we expect the model to correctly detect these evolution patterns for us, and to also estimate the future values for X_3, X_4 and X_5 accordingly.

The first step in the proposed model is to bring all time series to a same starting point $t = 1$, and then to calculate the daily variations, as it is shown in Table 16. Next, the model generates 5 networks, i.e., the maximum length of the series in $\mathscr{X}$ subtracted by 1, where each node represents a time series in $\mathscr{X}$ and the edges between them are created according to the similarities of their daily variations, on each time step t. For accomplishing this task, in this example, we make use of the nearest neighbor technique based on a radius $\varepsilon = 10$. Then, the *fast greedy* community detection algorithm is ran in the networks. This results in a temporal network $\mathscr{G}$, formed by the set $\{G_t \mid t \in [1,5]\}$. Note that, in this step, the model groups the time series with more similar variations, at each time step t, and, as t gets bigger, only the nodes from time series with longer lengths are left in $\mathscr{G}$. The edges evolution among the nodes of $\mathscr{G}$ is shown in Figure 25, where each row represents one time series in the dataset and each column represents one time slice of the temporal network. The colors denote the community to which each node belongs, at each time slice. In this case, if a node has a white color, it means that this node is not in the temporal network at this time slice.

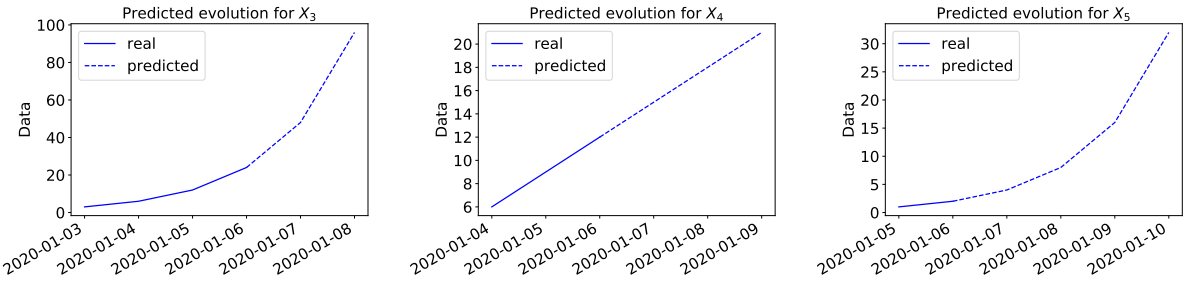
Following, the model creates a dictionary for each time series i, with a set of keys containing the instances that have shared the same community with i, and a set of values equal to the number of times this community sharing has occurred. Hence, in this case, we have that the dictionaries for X_3, X_4 and X_5 are: $D_3 = \{X_2 : 3, X_5 : 1\}, D_4 = \{X_1 : 2\}$ and $D_5 = \{X_2 : 1, X_3 : 1\}$, respectively. Therefore, according to Equation 7.7, the variation predicted for X_5 at the time step $t = 2$, is given by: $(1\delta X_{2,2} + 1\delta X_{3,2})/(1+1)$. In Figure 26, we show all predictions made by the model for the example dataset. As one may observe, the model is capable to correctly detect the

Figure 25 – Time slices showing the edges evolution in the temporal network, for the example dataset. Each row represents one time series and each column represents one time slice of the temporal network. The colors denote the community to which each node belongs, at each time slice. In this case, if a node has a white color, it means that this node is not in the temporal network at this time slice.



Source: Elaborated by the author.

Figure 26 – Predictions performed by the model for the time series (a) X_3, (b) X_4 and (c) X_5, from the example dataset. The blue line shows the series data provided in the dataset, while the blue dashed line indicates the predicted data.

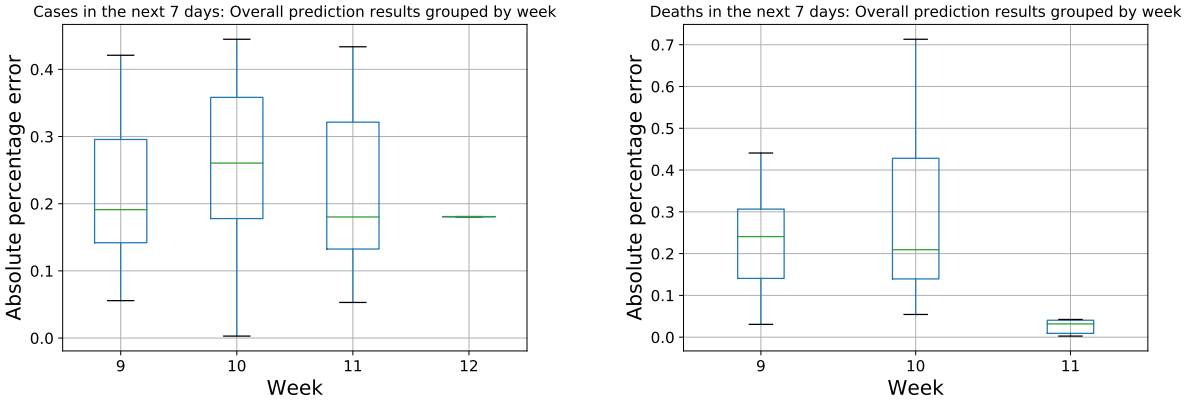


(a) X_3 (b) X_4 (c) X_5

Source: Elaborated by the author.

evolution patterns between X_1 and X_4 and between X_2 and X_3, and to make use of these detected correlations for predicting future values for X_3 and X_4. In the case of X_5, which could either evolve as an arithmetic progression or as a geometric one, the model ends up correlating it to X_2, and therefore the predictions for X_5 follow a geometric progression.

Figure 27 – Boxplots of the overall prediction absolute percentage errors, grouped by week, for the (a) confirmed new cases in the next 7 days (51 predictions in total) and (b) confirmed new deaths in the next 7 days (25 predictions in total), for each of the 27 federal units of Brazil.



(a) Confirmed new cases in the next 7 days: overall prediction results.

(b) Confirmed new deaths in the next 7 days: overall prediction results.

Source: Elaborated by the author.

7.4 Results and Discussion

In this section, we present the obtained results when applying the proposed model to COVID-19 preliminary data, in order to predict the future number of confirmed cases and deaths in the 27 federal units of Brazil.

In order to apply the proposed time series prediction model on the COVID-19 datasets, we start by converting the confirmed cases and deaths daily variations in the database to weekly variations. In this manner, given that the first confirmed case of COVID-19 in Brazil dates from 02-25-2020, in the state of SP, and that the final date in the database is 05-26-2020, we end up with a maximum of 12 weeks for the COVID-19 time series, considering all federal units of Brazil. We predict the number of new confirmed cases and deaths for the next 7 days in each federal unit, for all units which presented at least 20 confirmed cases or deaths in the considered period. Additionally, we opt for not considering the time series from regions located in China from the database as predictors, since these time series may present later corrections in their data, which affects the prediction outputs. We set the value of the ε radius threshold parameter, used for generating the edges in the networks, as $\varepsilon = Q(\delta X_t, .2)$, where $Q(A, n)$ stands for the n-th quantile of the array A. For detecting communities in the network, we use the *fast greedy community detection* algorithm.

In Figure 27, we show the boxplots of the absolute percentage errors for the obtained results when predicting the number of COVID-19 confirmed new cases and deaths in the next 7 days, for each of the 27 federal units of Brazil, from weeks 9 to 12. To avoid the inclusion of outliers, we did not consider in this figure predictions made for regions with a current total

Table 17 – Confirmed cases in the next 7 days: most recent predictions performed by the model for the federal units of Brazil.

Label	Week	Date	Predicted	Real	Error	Labels included in the prediction
AC	10	2020-05-26	1121	2019	0.44	Oman, PB, Japan
AL	11	2020-05-24	2003	2398	0.16	Oman, Afghanistan, Kuwait
AM	9	2020-05-15	6256	7665	0.18	Dominican Republic
AP	9	2020-05-22	1865	2025	0.08	Bahrain, RS, Mexico
BA	11	2020-05-22	3840	4429	0.13	Algeria, MG, Belarus
CE	8	2020-05-11	4389	6559	0.33	Denmark, Panama
DF	11	2020-05-23	2384	2108	0.13	RS, Bahrain, Bulgaria
ES	11	2020-05-21	2465	3065	0.20	Algeria, BA, DF
GO	10	2020-05-21	514	695	0.26	Guatemala, Iraq, Senegal
MA	9	2020-05-22	6690	7175	0.07	Pakistan, PE
MG	11	2020-05-24	1948	2057	0.05	RS, Armenia, DF
MS	10	2020-05-23	251	350	0.28	Singapore, Congo (Brazzaville), Gabon
MT	9	2020-05-22	405	479	0.15	Kenya, El Salvador, Jordan
PA	9	2020-05-20	4971	8585	0.42	Alberta, Qatar, Belarus
PB	10	2020-05-21	2187	2877	0.24	Sudan, Bolivia, AC
PE	10	2020-05-21	5341	8323	0.36	Belarus, Pakistan, Qatar
PI	9	2020-05-21	1048	1169	0.10	PB, Sudan, Bolivia
PR	10	2020-05-21	444	764	0.42	Cuba, MG, Bosnia and Herzegovina
RJ	11	2020-05-21	7149	12622	0.43	Qatar, Belarus, Bangladesh
RN	10	2020-05-21	1084	1465	0.26	Cote dIvoire, DF, Bahrain
RO	9	2020-05-22	772	980	0.21	Afghanistan, Bahrain, PB
RR	9	2020-05-23	629	733	0.14	GO, El Salvador, Guatemala
RS	11	2020-05-26	1165	2987	0.61	DF, Bahrain, Armenia
SC	10	2020-05-21	1371	1278	0.07	Ghana, MG, Bahrain
SE	10	2020-05-23	2351	1996	0.18	Bolivia, AL, Afghanistan
SP	12	2020-05-19	14972	18276	0.18	Quebec, Mexico
TO	9	2020-05-20	766	947	0.19	Belarus, Gabon, PI

Source: Research data.

number of confirmed cases or deaths less than 50, in each week. After applying this filter, we end up with a total of 51 predictions of new confirmed cases and 25 predictions of new confirmed deaths included in the boxplots. The median and mean absolute percentage error for the new cases prediction results, in Figure 27a, are 21% and 24%, respectively. This mean absolute percentage error (MAPE) is smaller than the ones obtained by Al-qaness *et al.* (2020), when forecasting the weekly new confirmed cases of influenza in China, which are of 32% and 45%, respectively, on each dataset analyzed. The accuracy obtained by Arora, Kumar and Panigrahi (2020), with a MAPE of around 6%, is higher than the one we obtained. However, it is worth mentioning that the testing data used in their study comprised only 4 Indian states and a 7-days time range, while in our case we consider 51 different weekly predictions, for more than 20 states. The median and mean absolute percentage error for the new deaths prediction results, in Figure 27b, are 16% and 23%, respectively.

We believe the prediction accuracy is relatively higher for the number of new confirmed deaths for the reason that these numbers tend to be more reliable than the statistics regarding

Table 18 – Confirmed deaths in the next 7 days: most recent predictions performed by the model for the federal units of Brazil.

Label	Week	Date	Predicted	Real	Error	Labels included in the prediction
AC	10	2020-05-26	31	24	0.29	Afghanistan, Israel, Belarus
AL	11	2020-05-24	103	106	0.03	Israel, Belarus, BA
AM	10	2020-05-22	579	338	0.71	Peru, PE
AP	9	2020-05-22	41	54	0.24	MG, Luxembourg, PR
BA	11	2020-05-22	123	118	0.04	Egypt, MG, Israel
CE	10	2020-05-25	642	745	0.14	Peru
DF	11	2020-05-23	19	39	0.51	Afghanistan, MG, Belarus
ES	11	2020-05-21	118	114	0.04	South Africa, BA, Moldova
GO	10	2020-05-21	22	18	0.22	North Macedonia, Iraq, Bolivia
MA	9	2020-05-22	156	198	0.21	Romania, Austria
MG	11	2020-05-24	32	70	0.54	Bulgaria, Pakistan, Slovenia
PA	6	2020-04-29	18	113	0.84	PB, Bosnia and Herzegovina
PB	10	2020-05-21	59	85	0.31	Saudi Arabia, MG, Bosnia and Herzegovina
PE	10	2020-05-21	524	627	0.16	AM, Peru, Mexico
PI	9	2020-05-21	28	33	0.15	PR, Afghanistan, Saudi Arabia
PR	10	2020-05-21	17	22	0.23	Saudi Arabia, RS, Slovenia
RJ	11	2020-05-21	1168	1165	0.00	SP
RN	10	2020-05-21	44	59	0.25	Cuba, MG, PB
RO	9	2020-05-22	30	44	0.32	Afghanistan, Israel, Belarus
RS	11	2020-05-26	55	43	0.28	Saudi Arabia, US, Slovenia
SC	10	2020-05-21	16	20	0.20	PR, Tunisia, Saudi Arabia
SE	10	2020-05-23	22	33	0.33	MG, Bulgaria
SP	11	2020-05-12	1095	1098	0.00	RJ

Source: Research data.

the new confirmed cases, since the later are subject to some difficulties, such as limited testing capabilities in each region and also the lack of symptoms on some individuals infected by the disease. One can also note, still in Figure 27, that the boxplot for week 12 is smaller than the ones from previous weeks. This is due to the fact that, up to this date, only the state of SP in Brazil has been infected by COVID-19 for more than 11 weeks, hence this boxplot actually includes only the prediction of new cases performed for the state of SP.

In Table 17 and Table 18, we present the most recent predictions performed by the model for the Brazilian federal units, for new confirmed cases and deaths in the next 7 days, respectively, along with their respective real values and absolute percentage errors. The last column in these tables shows the labels with the highest weights considered in each prediction, i.e., the regions in the database whose time series the model identified as being most correlated to the predicted time series and whose lengths are greater or equal to the predicted period. The states of MS, MT, RR and TO do not appear in Table 18 either because they presented less than 20 confirmed deaths caused by COVID-19 in the considered period or because their weekly deaths variations did not match any other in the period, and hence the model did not attempt to perform predictions of new confirmed deaths for these states.

One can note, in Table 17 and Table 18, that the model is able to predict both smaller

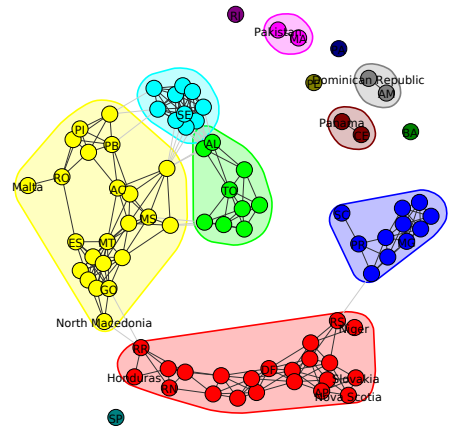
values, as for the new confirmed deaths in GO, PR and SC, and also larger values, such as the new confirmed deaths in SP and RJ. This is because the predictions are made in terms of variations, and not in terms of the actual numbers. It is interesting to analyze these two tables along with Figure 28, in which we have examples of the correlations networks built by the model for weeks 5, 7 and 9 for the federal units of Brazil and their respective neighbors, i.e., their most correlated world regions in each week. The left column in Figure 28 shows the correlations networks for new confirmed cases, while the right column shows the correlations networks for the new confirmed deaths. In this sense, one can note, for instance, that the states of RJ and SP presented similar new deaths variations, on week 5, and thus formed the community denoted by the aquamarine color in Figure 28b. On weeks 7 and 9, the new deaths variation in these same states did not match the variation from any other region, hence they both appear isolated in Figure 28d and Figure 28f. In Table 18, the model took into account only the correlation detected in Figure 28b for predicting the new deaths in these two states on week 11, and both predictions achieved a very high accuracy in this case, with an absolute percentage error of less than 1%.

Although the model is able to perform relatively accurate predictions for some regions, as it is shown in Table 17 and Table 18 and in the overall performance shown in Figure 27, it also has some limitations. The most recent predictions made for the state of PA, for instance, listed in the mentioned tables, obtained a high absolute percentage error, of 42% and 84%, for the prediction of new cases and deaths, respectively. This happens when the time series to be predicted presents a drastic shift in its curve, such that this shift does not match the variation from any other time series in the dataset, for that same time step. That is the case for the state of PA, which suffered from a suddenly rise in both of its COVID-19 curves, after the 5th week, causing it to appear isolated in Figure 28c, Figure 28d, Figure 28e, and Figure 28f, and which also prevented the model from predicting new deaths for this state after the 6th week, on 2020-04-29. This issue should be addressed in future versions of the model, by allowing it to adapt to such situations, in order to improve its prediction capabilities.

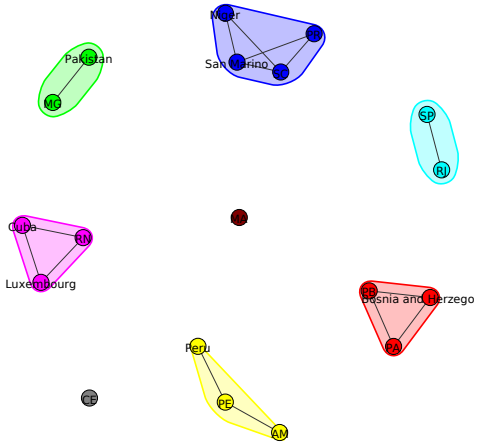
7.5 Chapter Remarks

In this chapter, we introduced a semi-supervised regression model which predicts future values for time series based on a correlations temporal network. The obtained results, by applying the model to predict the new confirmed cases and deaths related to COVID-19, in the 27 federal units of Brazil, are promising, with a mean absolute percentage error of 24%, for the new cases prediction, and a mean absolute percentage error of 23%, for the new deaths prediction. We consider these preliminary results as satisfactory. Especially when we take into account that the data concerning the context of this application are subject to the influence of external conditions, such as isolation policies, for example, and that the model did not consider any of these external factors for generating the predictions. Thus, the fact that the model was still able to perform fairly well, despite the difficulties involved in this application, corroborates to demonstrate that

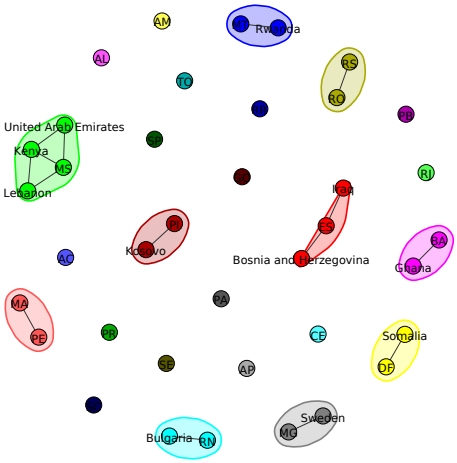
Figure 28 – Examples of correlations networks formed by the federal units of Brazil and their respective neighbors which, in this case, represent their most correlated regions in each week, in terms of COVID-19 confirmed cases (left) and deaths (right) weekly variations. The colors denote the network communities. For the sake of visibility, not all labels are shown in these figures.



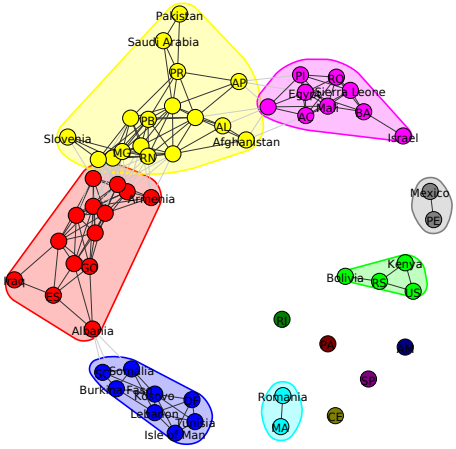
(a) Cases: week 5.



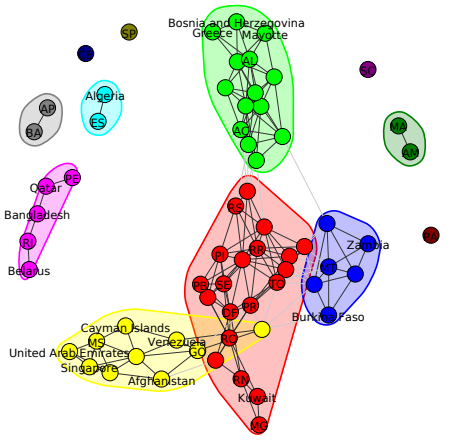
(b) Deaths: week 5.



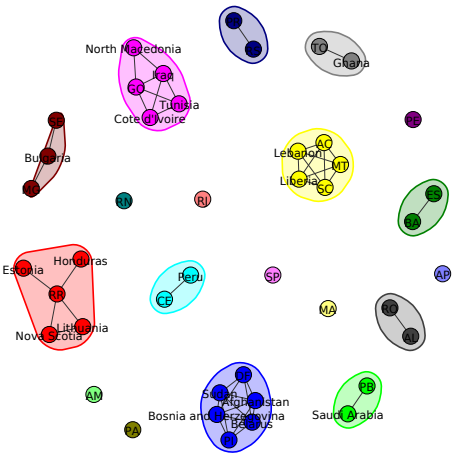
(c) Cases: week 7.



(d) Deaths: week 7.



(e) Cases: week 9.



(f) Deaths: week 9.

Source: Elaborated by the author.

the model's rationale is valid, and might as well be applied successfully to predict time series from different contexts and areas.

CHAPTER

8

DETECTING SIGNS OF HEPATIC, RENAL AND RESPIRATORY INSUFFICIENCY IN COVID-19 PATIENTS

In this chapter, we introduce a network-based classification technique to assist medical professionals on the COVID-19 severity assessment by detecting early signs of insufficiency in infected patients using only Complete Blood Count (CBC) test results. The method proposed in this study can be especially useful for those healthcare professionals who work in regions with scarce material and human resources, as it is the case in some regions of Brazil, where complementary tests to detect the severity of COVID-19 in patients at high risk of complications may not be available.

The proposed model consists of a modified high level classification technique, whose main improvements are: (1) the option to reduce the size of the network, during the training phase, for saving processing time and making the technique more efficient, and (2) the design of a parameter optimization criteria, based on a variant of Kullback-Leibler divergence, to make the technique more adaptable to each dataset. The model is evaluated firstly by using benchmark classification datasets, both artificial and real ones. Afterwards, we apply the model to detect early signs of hepatic, renal and respiratory insufficiency in COVID-19 patients of a dataset from one of the main hospitals in Brazil, based on their CBC test results. The model's performance is compared to those achieved by traditional and well-known classification models, when applied on the same data.

8.1 Introduction

Since the COVID-19 outbreak, within a short period of time, advanced machine learning techniques have been applied in the development of clinical decision support tools, to help

medical workers on guiding the monitoring and treatment plans for COVID-19 patients. Wang *et al.* (2020) presented a diagnosis and screening method for COVID-19 patients based on the analysis of breathing patterns. Metsky *et al.* (2020) provided assay designs for detection of viral species and subspecies, including SARS-CoV-2, through a CRISPR-based detection system. Gozes *et al.* (2020) and Shan *et al.* (2020) proposed an automated analysis of computed tomography (CT) images for the monitoring of COVID-19 patients over time. Yan *et al.* (2020) predicted the mortality of patients, by using three biomarkers, selected from blood samples.

In this study, we present a technique which may assist medical professionals on the patients' severity assessment by detecting early signs of insufficiency in COVID-19 patients using only CBC test results, before the consequences become irreversible. Specifically, our approach can help to identify the level of risk and the type of insufficiency for each patient. We start by building a dataset comprising results from both CBC and specific tests to detect signs of hepatic, renal and respiratory insufficiency, from a total of 2,982 COVID-19 patients who received treatment in the Israelite Albert Einstein Hospital, from Sao Paulo, Brazil (FAPESP, 2020). Next, we identify which CBC tests are more effective to be used as biomarkers to detect signs from each type of insufficiency. The training dataset resulting from this analysis is then delivered to a modified network-based high-level classification technique, which is also introduced in this study. The obtained classification results present competitive performance of the technique compared to classic and the state-of-the-art ones.

The main contributions are summarized as follows:

- Introducing a modified network-based high level classification technique with the following main improvements: (1) Presenting a network reduction method to make the technique more efficient, and (2) Designing a parameter optimization criteria, based on a variant of Kullback-Leibler divergence, to make the technique more adaptable to each dataset.

- Applying the technique to detect insufficiency signs in COVID-19 patients, along with a performance comparison to classic and the state-of-the-art classification techniques.

- Building a training dataset from the unlabeled original data, which can be also used in other relevant researches.

Regarding the organization of this chapter, besides this introduction, we discuss, in section 8.2, the motivations for this study. In section 8.3, the proposed technique is described in details. We also introduce in this section the datasets used for evaluating the model, including the COVID-19 dataset, which was especially built for this study. In section 8.4, we start by showing the performance of the model when applied to benchmark datasets and, afterwards, we present the results obtained when the model is applied to predict early signs of insufficiency in COVID-19 patients. At the end, in section 8.5, we close this chapter with some final remarks.

8.2 Motivation

When a new COVID-19 case is confirmed, the clinical protocol involves an initial assessment by healthcare professionals to classify its severity as a mild, moderate or severe case. This procedure usually includes a Complete Blood Count (CBC), which is a series of tests used to evaluate the composition and concentration of the cellular components of blood. Such type of testing, as the CBC, is widely used, because it can be easily collected at any place, even in situations of medical resource scarcity. Moreover, it is usually processed in a matter of minutes, up to two hours. Therefore, it is considered as a fast, low-cost and reliable resource to assess the patient's overall conditions. Based on the constant monitoring of these tests results, the healthcare professionals then need to decide whether or not to order complementary tests for detecting signs of hepatic, renal or respiratory insufficiency. Because the progressive respiratory failure is the primary cause of death of COVID-19, the monitoring activity hence is critical for the patient. However, we must face the problem that such complementary tests incur high costs and may not even be available in some specific places due to scarcity of material and human resources, as it is the case in some regions of Brazil. In this sense, the study presented in this chapter has the potential of helping healthcare workers, specially the ones who are based in regions with scarce material resources, on this monitoring activity, by introducing a risk assessment technique which is based solely on CBC test results.

Real-world datasets usually contain complex and organizational patterns beyond the physical features (similarity, distance, distribution, etc.). Data classification, which take into account not only physical features, but also the organizational structure of the data, is referred to as *high level* classification. Complex networks are suitable tools for data pattern characterization due to their ability of capturing the spatial, topological, and functional relationship among data. In this study, we present a modified high-level classification technique based on complex network modeling. It can perform classification tasks taking into account both physical attributes and pattern formation of the data. Basically, the classification process measures the compliance of the test instance to the pattern formation of each network constructed from a class of training data. The approach presented in this work is inspired by the high-level data classification technique introduced by Carneiro and Zhao (2017), Colliri *et al.* (2018) and Silva and Zhao (2012a).

8.3 Materials and Methods

In this section, we describe the modified optimized Network-Based High Level (MNBHL) classification technique, as well as introduce the datasets and indicators used for evaluating its performance. We start by providing an overview of the model, in subsection 8.3.1. Then, in subsection 8.3.2, we explain in details how the network is generated from the training data. In subsection 8.3.3, we demonstrate how the labels are assigned to new data instances, during the testing phase. In subsection 8.3.4, we provide more details regarding the cost function used

for optimizing the parameters of the model. At the end of the section, in subsection 8.3.5, the datasets and indicators used for evaluating the model are presented.

8.3.1 Model Overview

In classic supervised learning, initially we have an input dataset comprising n instances in the vector form $X_{train} = (x_1, x_2, ..., x_n)$, where each instance x_i itself is a m-dimensional vector, such that $x_i = (x_{i,1}, x_{i,2}, ..., x_{i,m})$, representing m features of the instance. Correspondingly, the labels of the instances are represented by another vector $Y = (y_1, y_2, ..., y_n)$, where $y_i \in \mathscr{L} = \{L_1 ..., L_C\}$ and L_i is the label of the ith class. The objective of the *training phase* is to construct a classifier by generating a mapping $f : X_{train} \xrightarrow{\Delta} Y$. In the *testing phase*, we use the classifier constructed so far to classify new data instances without label. The test dataset is denoted as X_{test}.

A network can be defined as a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, where $\mathscr{V}$ is a set of nodes, $\mathscr{V} = \{v_1, v_2, ..., v_n\}$ and $\mathscr{E}$ is a set of tuples, $\mathscr{E} = \{(i, j) : i, j \in \mathscr{V}\}$, representing the edges between each pair of nodes. In the proposed high-level classification model, each node in the network represents a data instance x_i in the dataset. The connections (or edges) between the nodes are created based on the similarity of their corresponding data instances in the attribute space, according to a combination of two rules: kNN and ε-radius, which will be detailed later.

In the *training phase*, we start by generating the balancing parameter α for dealing with unbalanced datasets. Then, each network component is constructed for the data instances of each class. Afterwards, we introduce a method to reduce the network, by maintaining only the $r\%$ most central nodes in each network measured by a selected centrality index. At the last step of training phase, the model calibrates the parameters, through a cost function optimization and updates the network by using these optimum values. In the *testing phase*, each data instance is inserted as a new node in the network, one at a time, by using the same rules of the training phase. In the case only one component from the network is affected by the insertion, then its label will be equal to the respective class of the affected component. For cases when more than one component is affected by the insertion, then its label will be given by the class whose component is the least impacted by the new node's insertion, in terms of the network topological measures.

8.3.2 Description of the Training Phase

8.3.2.1 Balancing

The training phase starts by normalizing the values in X_{train}. For dealing with unbalanced datasets, we introduce a quantity α, which has the role of balancing the different size effect

yielded for each class, during the testing phase. The values of α are calculated by:

$$\alpha_L = \frac{|X_{train}^{y=L}|}{|X_{train}|} \quad , \forall L \in \mathscr{L} \quad , \text{ and} \tag{8.1}$$

$$\alpha_L = \frac{\alpha_L}{\sum \alpha} \quad . \tag{8.2}$$

where $|X_{train}|$ is the total number of elements of the training set and $|X_{train}^{y=L}|$ is the number of elements with class label L.

8.3.2.2 Network Generation

The edges between nodes in the network are generated by measuring the pairwise similarity of the corresponding data instances in the attribute space and it is a combination of two rules: kNN and ε-radius. The ε-radius rule is used for dense regions, while the kNN is employed for sparse regions. The value of ε is yielded by:

$$\varepsilon = Q(D, p) \quad , \tag{8.3}$$

where D_i is a 1d vector containing the Euclidean distances between each element and element x_i in X_{train}, and Q is a function which returns the p-th quantile of the data in D. Thus, given that the value of ε depends on the p-th quantile of D and considering that the value of ε will be calibrated later still in the training phase, we fix the bounds for parameter ε in terms of the quantile $p \in [0.015, 0.025]$. Since we still do not know the optimum value for ε at the beginning, we set it as $Q(D, 0.02)$, which is the averaged value of its boundaries.

The second rule used for generating the edges in the network is the kNN. In the same manner of ε, this parameter will also be calibrated later in the training phase, so we decide to fix the boundaries of k at the beginning and we set its initial value as the average of its boundaries. The bounds are given by $k \in [3, 0.025\overline{v}]$, where v is a vector containing the number of data instances per each class in the dataset. Hence, the maximum possible value for k will be the average number of data instances per class in the dataset multiplied by the same upper bound of the quantile used for calculating the parameter ε. Note that, by proceeding this way, we are therefore linking the maximum value of k both to the characteristics of each dataset (here represented by $\overline{v}$) and to the maximum value of ε. This is a novelty introduced in this study, and it is aimed to avoid exaggerated disproportions between the number of edges yielded by the two rules. However, for cases when the number of data instances in one class is either smaller than the lower bound or is very large, e.g., bigger than one million, then one can also adjust the k bounds accordingly.

With the initial values of ε and k being set, the model then proceeds to generate the edges of the network $\mathscr{G}$. The neighbors connected to a training node x_i are yielded by:

$$N(x_i) = \begin{cases} \varepsilon\text{-radius}(x_i, y_i), \text{ if } \varepsilon\text{-radius}(x_i, y_i) > k \\ k\text{NN}(x_i, y_i), \text{ otherwise} \end{cases} \quad , \tag{8.4}$$

where y_i denotes the class label of the training instance x_i, ε-radius(x_i, y_i) returns the set $\{x_j, \forall j \in \mathcal{V} \mid dist(x_i, x_j) \leq \varepsilon \wedge y_i = y_j\}$, and kNN(x_i, y_i) returns the set containing the k nearest vertices of the same class of x_i. This combination of rules is the same one used by Silva and Zhao (2012a).

8.3.2.3 Network Reduction

At this point of the training phase, there is an option to reduce the number of vertices in the network, for the sake of saving processing time, both in the training phase and in the testing phase. There are several possible strategies for reducing a network (VALEJO *et al.*, 2020) and, in this work, we opt for a quite simple one, which consists in keeping only the nodes which occupy more central positions in it. For this end, we make use of the *betweenness centrality* measure (BRANDES, 2001). This measure, broadly speaking, estimates the extent to which a vertex lies on paths between other vertices, and is considered essential in the analysis of complex networks. In a network which does not present communities, the nodes with higher betweenness centrality are among the most influential ones (CHEN *et al.*, 2012), and in a network which presents communities, these nodes are the ones which usually work as links between the communities. We chose to use this measure for believing that it is able to depict the most representative nodes in the network, such that we can use only these selected nodes for the classification task.

For controlling this option, we add a reduction parameter $r \in [0, 1]$ in the model, such that when its value is set as less than 1.0, then only the ratio of r vertices with higher betweenness centrality measures are kept in the network. In this case, the values in X_{train} and Y are also reduced, accordingly. The value of ε is updated, and the network edges are generated again, by using the same rules provided in Equation 8.3 and Equation 8.4, respectively. If, after this procedure, the number of components in the network is higher than the number of classes in the dataset, then the value of k is increased by 1, and the edges are updated accordingly. This step can be repeated more times, if necessary, until we have, at the end, one component in the network per class in the dataset.

8.3.2.4 Parameters ε and k Calibration

The last step in the training phase is to find the optimum values for the parameters ε and k, to be used in the testing phase. This is made by minimizing the cost function estimated for the classifier, i.e., by forcing the model to classify the data instances from X_{train}, varying the values of ε and k between their respective boundaries, at each time, and also estimating the corresponding cost associated, by using the labels from Y. In this procedure, we associate once again the value of k to the parameter ε. This is because ε is a continuous variable and k is a discrete variable, and finding the minimum of a continuous function is less complex than finding the minimum of a mixed discrete-continuous function. Therefore, for optimization purposes, we

opt to yield the values of k according to:

$$k = K^{min} + \left\lfloor \frac{n - N^{min}}{N^{max} - N^{min}} \right\rceil (K^{max} - K^{min}) \,, \tag{8.5}$$

where $\lfloor x \rceil$ stands for x rounded to the nearest integer, n is the quantile used to generate the current parameter ε, and K and N are the possible values for k and n, respectively. The association of the value of k to the value of ε, besides saving processing time, also makes the optimization search more rational, since the values for these two variables are the ones responsible for generating the edges, with smaller values for both of them resulting in a less connected network and, alternately, higher values for both of them resulting in a more connected network. The details regarding the cost function optimization used in the model are provided in subsection 8.3.4. At the end of the parameters calibration procedure, the network $\mathscr{G}$ is updated for the last time, by using the optimum values achieved for ε and k.

The complete process of the training phase is outlined in Algorithm 5. In lines 2-4, we have the balancing tasks, in which the parameter α is generated and the values in X_{train} are normalized. Then, in line 5, the network $\mathscr{G}$ is built, by using the initial values for the parameters ε and k. In lines 6-11, there is the optional procedure to reduce the size of the network, by leaving only the $r\%$ most central nodes in it, and also the validation of whether the number of network components is equal to the number of classes from the training data. In line 12, the cost function optimization routine is called, and the optimum values for ε and k are defined. Finally, in line 13, the final network is generated, which is the one to be used in the testing phase.

Algorithm 5 – Training phase

```
1:  procedure FIT(X, Y)
2:      α ← fraction of items per class (balancing parameter)
3:      if normalize then
4:          normalize values in X
5:      end if
6:      G ← initial network, resulted from initial ε and k
7:      vs ← the r% most central nodes in G
8:      G ← reduced version of G, leaving only vs
9:      X, Y ← reduced versions of X, Y
10:     while len(G^components) > number of classes in Y do
11:         k += 1
12:         G ← network resulted from ε and k
13:     end while
14:     ε, k ← optimum values, using the cost function
15:     G ← network resulted from final ε and k
16: end procedure
```

8.3.3 Description of the Testing Phase

In the testing phase, the data instances from X_{test} are inserted as new vertices in the network $\mathscr{G}$, one at a time, and the model then needs to infer the label of each testing instance. The model extracts two groups of network measures for each component, before and after the insertion of a testing instance, respectively. The testing instance is classified to the class of the component which its insertion caused the smallest variation in the network measures. In other

words, a testing instance is classified to a certain class because it conforms the pattern formed by the training data of that class. In this way, the data instances from X_{test} are inserted in the network, to be classified, one by one.

The new node's insertion is made by using the same rules described in Equation 8.4 for generating the edges between the new node and the other nodes in the network. The only difference now is that, since we do not know the class of the new data instance x_i, the "same label" restriction is removed from the procedure, so that the nodes to be connected to x_i are yielded by:

$$N(x_i) = \begin{cases} \varepsilon\text{-radius}(x_i), \text{ if } \varepsilon\text{-radius}(x_i) > k \\ k\text{NN}(x_i), \text{ otherwise} \end{cases}, \tag{8.6}$$

where ε-radius(x_i) returns the set $\{x_j, \forall j \in \mathscr{V} \mid dist(x_i, x_j) \leq \varepsilon\}$, and kNN(x_i) returns the set containing the k nearest vertices of x_i. Note that, in this phase, the optimum values of ε and k are used in this procedure.

The model will assign the label for the new node according to the number of components affected by the node's insertion in the network. In case only one component is affected by its insertion, i.e., when all the target nodes of its edges belong to the same component, then the model will assign the class of this component for the new node. On the other hand, when more than one component is affected by the insertion, then the model will extract new measures of the affected components and assign a class label to the testing node as the label of the component which was less impacted by the insertion, in terms of measures. In this work, we use *betweenness centrality* for measuring the impacts on each network's component.

The overall impact I^L on each affected component of class L, caused by the insertion of the testing node, is calculated and balanced as follows:

$$I^L = \alpha^L \frac{(M_1^L - M_0^L)}{M_0^L} \quad , \tag{8.7}$$

where M_0^L and M_1^L are the extracted network measures for the component representing class L, before and after the insertion of the testing node, respectively. The probability $P(x_i^{\hat{y}_i=L})$ of the testing instance x_i to belong to class L is given by the reciprocal of the value in vector I in the normalized form, as in:

$$P(x_i^{\hat{y}_i=L}) = \begin{cases} \frac{1/I^L}{\Sigma_L 1/I^L}, \text{ if } C^L \text{ is affected} \\ 0, \text{ otherwise,} \end{cases} \tag{8.8}$$

where C^L is the network component representing class L. Finally, the label $\hat{y}_i$ to be assigned for x_i is yielded by:

$$\hat{y}_i = \mathscr{L}_{\arg\max_L P(x_i^{\hat{y}_i=L})} \quad . \tag{8.9}$$

The complete process of the testing phase is described in Algorithm 6. In lines 4-5, an initial empty list is created to store the predicted labels for each data instance in X_{test} and the

current measures of each network's component are extracted. At lines 6-15, the model generates a label for each testing instance, one by one. In case only one component is affected by the insertion of the testing node, then the label receives this component's class. In case more than one component is affected by the insertion, then the testing instance is classified to that class whose component is least impacted by the insertion in terms of the betweenness centrality measure. At the last row of the for-loop, in line 16, the classification list is updated with each predicted label.

Algorithm 6 – Testing phase

```
 1: procedure PREDICT(X)
 2:     if normalize then
 3:         normalize X, using original values from X_train
 4:     end if
 5:     Ŷ ← [ ]
 6:     M_0 ← initial measures of each component in G
 7:     for x in X do
 8:         insert x in G, using ε and k
 9:         if x affects only one component in G then
10:             ŷ ← class of affected component
11:         else
12:             M_1 ← new measures of each component in G
13:             I ← α · abs(M_1 − M_0)/M_0
14:             P ← 1.0/I
15:             P ← P/sum(P)
16:             ŷ ← class of argmax(P)
17:         end if
18:         Ŷ ← append ŷ
19:     end for
20: return Ŷ
21: end procedure
```

8.3.4 Cost Function Optimization

In order to calibrate the values of parameters ε and k, we here introduce a cost function for optimization based on a variant of the *Kullback-Leibler* (KL) divergence (KULLBACK; LEIBLER, 1951). The KL divergence between a true probability distribution p and the estimated probability distribution q is given by:

$$D_{KL}(p \,||\, q) = H(p,q) - H(p) \quad , \tag{8.10}$$

where $H(p)$ is the *entropy* (SHANNON, 1948) of the true probability distribution p and $H(p,q)$ is the *cross-entropy loss*. This cost function is commonly used in deep learning models for optimization purposes. However, in this work, we make a slight change in Equation 8.10 by switching its second term, so it becomes:

$$F(p \,||\, q) = H(p,q) - H(q) \quad , \tag{8.11}$$

which, in the case of one-hot vectors, i.e., when one class j has a probability of 100% and the rest have 0%, it is equivalent to:

$$F(p \,||\, q) = -\log(q_j) + \sum_i q_i \log(q_i) \mid p_j = 1 \quad . \tag{8.12}$$

In this manner, the first term of the cost function in Equation 8.12 continues to penalize smaller probabilities given for the true label, while the second term penalizes probability distributions with a smaller entropy. At first, the role of the second term in Equation 8.12 might not make much sense since, in machine learning models, probability distributions with a smaller entropy are usually more desirable. To understand this, we need to consider the fact that, in the proposed model, the estimated probability distribution q will assume the form of a one-hot vector, i.e., with zero entropy, exclusively in cases when only one network component is affected by the new data instance's insertion. Consequently, in these cases, the network measures will not be taken into account in the classification process. Hence, by inserting the second term in Equation 8.12, we are forcing the model to prioritize classifications which are based on the impacts of the new data instance on the network measures.

For the optimization search, we make use of the "brute force" method for estimating the minimum cost, i.e., we compute the cost function value at each point of a multidimensional grid of points, each time with a different value for ε and k, to find the global minimum of the function.

8.3.5 Database

We first evaluate the performance of the proposed modified network-based high level (MNBHL) model by applying it to well-known benchmark datasets, both artificial and real ones, which are intended for machine learning classification tests. Afterwards, we apply the MNBHL model to a dataset specially built for this work, obtained from the analysis of publicly available data of COVID-19 patients from one of the main hospitals in Brazil. In subsubsection 8.3.5.1 and subsubsection 8.3.5.2, we provide more information concerning these two databases.

Regarding the proposed network reduction parameter r, we set $r = 0.1$ (10%) when processing the Digits benchmark dataset and the COVID-19 dataset, and $r = 0.2$ (20%) when processing the Breast Cancer and Pima benchmark datasets. For the remaining datasets, we set $r = 1$, i.e., we do not make use of the network reduction option when processing them, since they are smaller ones. For the sake of comparison, the following traditional classification models are applied on the same datasets: Decision Tree (SAFAVIN; LANDGREBE, 1991), Logistic Regression (GELMAN; HILL, 2007), Multilayer Perceptron (MLP) (HINTON, 1989), Support Vector Machines (SVM) with an RBF kernel (VAPNIK, 2000) and Naive Bayes (RISH, 2001). We also apply the following ensemble methods: Bagging of Decision Tree and Bagging of MLP (BREIMAN, 1996), Random Forest (BREIMAN, 2001) and AdaBoost (FREUND; SCHAPIRE, 1995). All traditional models are implemented through the tool introduced by Pedregosa *et al.* (2011) and we kept their respective default parameters values, in all tests performed.

8.3.5.1 Benchmark Datasets

A succinct meta-information of the selected benchmark datasets used for obtaining the experimental results is given in Table 19. Circles_00 and Circles_02 datasets are two concentric circles without noise and with a 20% noise, respectively. Moons_00 and Moons_02 datasets are two moons (or bananas) without noise and with a 20% noise, respectively. For a detailed description of the real datasets, one can refer to Lichman (2013). For splitting each dataset into 2 subdatasets, for training and testing purposes, we make use of a function which shuffles the data, through a random seed value, and returns a train-test split with 75% and 25% the size of the inputs, respectively.

Table 19 – Meta information of the classification datasets used in the experimental results, for evaluating and comparing the MNBHL model

		N^o of Samples	**N^o of Features**	**N^o of Classes**
Artificial	Circles_00	100	2	2
	Circles_02	100	2	2
	Moons_00	100	2	2
	Moons_02	100	2	2
Real	Breast Cancer	569	30	2
	Digits	1,797	64	10
	Iris	150	4	3
	Pima	768	8	2
	Wine	178	13	3
	Zoo	101	16	7

Source: Research data.

8.3.5.2 The COVID-19 Dataset

We build a training dataset from the unlabeled COVID-19 dataset (FAPESP, 2020), which is collected from COVID-19 patients of the Israelite Albert Einstein Hospital, located at Sao Paulo city and one of the main hospitals in Brazil. The original database comprises a total of 1,853,695 results from 127 different tests, collected from 43,562 de-identified patients, who received treatment in the hospital from January 1, 2020 until June 24, 2020. We firstly identify the patients who tested positive in at least one of the COVID-19 detection tests. The tests considered for this end are:

- polymerase chain reaction (PCR),
- immunoglobulin M (IgM),
- immunoglobulin G (IgG), and
- enzyme-linked immunosorbent assay (ELISA).

Next, we filter the patients and left in the dataset only the ones who have made at least one complete blood count (CBC) test, in a date no earlier than the date to be tested positive for COVID-19. In the case that a patient has made more than one CBC test, we consider only the results of the first one for predicting signs of hepatic, renal or respiratory insufficiency. Afterwards, we run an algorithm to automatically label each patient of the dataset by the type of insufficiency detected from the results of additional specific tests of the patients and according to the reference values provided in the same database. After the data cleansing, we end up with a dataset with a total of 2,982 different patients. Each patient of the dataset belongs to one of the following 4 classes: Healthy, Sign of Hepatic Insufficiency, Sign of Renal Insufficiency, and Sign of Respiratory Insufficiency.

In Table 20, we provide an overview of the COVID-19 dataset. The dataset is split into 2 subdatasets, one for training and another for testing purposes. The training-testing splitting is half-half. Thus, the training set is composed of the first 1,471 data instances, and the remaining 1,471 data instances are used for testing purposes. All tests considered in this analysis are listed in Table 21. For determining the predictive attributes for each type of insufficiency, we make use of the *Pearson correlation coefficient* (PEARSON, 1895; BRAVAIS, 1844), selecting the CBC tests whose results are most correlated to the labels, i.e., to each type of insufficiency.

Table 20 – Overview of the patients in the COVID-19 dataset

Age	**Total**	**Healthy**	**With Signs of Insufficiency**		
			Hepatic	**Renal**	**Respirat.**
00-20	57	25	19	22	9
21-40	830	467	258	191	134
41-60	1279	516	554	404	373
61-80	688	135	388	377	404
80+	128	5	84	98	88
Total	2982	1148	1303	1092	1008

Source: Research data.

The models performances were evaluated by assessing the classification accuracy (ratio of true predictions over all predictions), precision, sensitivity/recall and F1 scores, as defined below:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{8.13}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8.14}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Sensitivity}}{\text{Precision} + \text{Sensitivity}} \tag{8.15}$$

where TP, TN, FP and FN stand for true positive, true negative, false positive and false negative rates, respectively.

Table 21 – Tests considered for identifying signs of hepatic, renal and respiratory insufficiencies in patients, along with the respective attributes used for detecting those signs

		Tests considered for the detection of signs (labels generation)	Attributes for each patient vertex v					
			Age	Neutrophils	Basophils	Lymphocytes	Eosinophils	RDW
Insufficiency	**Hepatic**	• transaminase TGO (AST) • transaminase TGP (ALT) • alkaline phosphatase • gamma-glutamyl transpeptidase (GGT)	x	x	x	x	x	
	Renal	• creatinine clearance (urine test) • creatinine (blood test) • blood urea nitrogen (BUN)	x	x		x		
	Respiratory	• arterial blood gas • respiratory pathogen panel • lactate • ionized calcium • potassium	x		x	x		
	Any		x	x	x	x		x

Source: Research data.

8.4 Results and Discussion

In this section, we start by presenting the obtained results when assessing the performance of the proposed MNBHL model when applying it to benchmark classification datasets, in subsection 8.4.1, and then, in subsection 8.4.2, we present the obtained results when applying it to the COVID-19 dataset.

8.4.1 Tests Performed on Benchmark Datasets

In this subsection, we present the obtained results when applying the proposed MNBHL model to both artificial and real benchmark classification datasets, along with a comparison of its performance with the ones achieved by traditional classification models, on the same data.

In Table 22, we present the obtained results, in terms of accuracy, from the application of the proposed MNBHL model to the benchmark datasets, as well as the comparison with the performances achieved by traditional models. These results indicate that the model's overall performance is competitive, being ranked as third one when compared to traditional models, in the average rank. We highlight the good performance achieved by the model on three datasets: Moons_00, Moons_02 and Zoo, in which it obtained an 100% accuracy in the classification task. We also would like to remember that we make use of the model's network reduction option for processing the Breast Cancer, Digits and Pima datasets, with a reduction to 20%, 10% and 20% of the training data, respectively. Therefore, these results also help to demonstrate that the rationale behind the network reduction process – which keeps only the most central nodes in the network, in terms of betweenness centrality – is valid, since the model was still able to achieve a relative good performance on those datasets, especially for the Breast Cancer dataset.

In Table 23, we show the running times of all considered models, when processing each benchmark dataset, for comparison purposes. Although the proposed model is overall slower

Table 22 – Experimental results: accuracy rates (%) for each dataset obtained by the following models, in that order: MNBHL, AdaBoost, Bagging of Decision Tree, Bagging of MLP, Decision Tree, Logistic Regression, MLP, Naive-Bayes, Random Forest and SVM. The values between parenthesis indicate the rank achieved by each model on each dataset.

	MNBHL	**Ada**	**BagDT**	**BagMLP**	**DT**	**LR**	**MLP**	**N-B**	**RF**	**SVM**
Breast Cancer	95.1 (3)	97.2 (1)	96.5 (2)	89.5 (7)	94.4 (4)	93.7 (5)	90.9 (6)	95.1 (3)	95.1 (3)	68.5 (8)
Circles_00	88.0 (2)	92.0 (1)	88.0 (2)	68.0 (5)	84.0 (3)	48.0 (7)	80.0 (4)	88.0 (2)	84.0 (3)	56.0 (6)
Circles_02	68.0 (4)	84.0 (1)	76.0 (3)	52.0 (6)	84.0 (1)	52.0 (6)	80.0 (2)	64.0 (5)	76.0 (3)	40.0 (7)
Digits	93.3 (6)	25.1 (10)	93.6 (5)	99.3 (1)	83.1 (8)	97.1 (3)	98.7 (2)	85.3 (7)	94.2 (4)	35.6 (9)
Iris	94.7 (3)	86.8 (4)	86.8 (4)	100. (1)	86.8 (4)	97.4 (2)	100. (1)	94.7 (3)	86.8 (4)	94.7 (3)
Moons_00	100. (1)	100. (1)	92.0 (3)	92.0 (3)	92.0 (3)	88.0 (4)	92.0 (3)	92.0 (3)	96.0 (2)	92.0 (3)
Moons_02	100. (1)	100. (1)	100. (1)	92.0 (2)	100. (1)	88.0 (3)	92.0 (2)	88.0 (3)	100. (1)	100. (1)
Pima	68.8 (6)	72.9 (1)	72.4 (2)	70.8 (4)	66.7 (7)	72.4 (2)	70.3 (5)	71.9 (3)	72.9 (1)	61.5 (8)
Wine	95.6 (3)	97.8 (2)	95.6 (3)	55.6 (4)	95.6 (3)	97.8 (2)	42.2 (5)	97.8 (2)	100. (1)	40.0 (6)
Zoo	100. (1)	69.2 (4)	100. (1)	100. (1)	100. (1)	96.2 (2)	100. (1)	100. (1)	100. (1)	84.6 (3)
Average Rank	3rd	2nd	2nd	6th	7th	8th	4th	5th	1st	9th

Table 23 – Running times, in seconds, on each dataset, measured for the following models, in that order: MNBHL, AdaBoost, Bagging of Decision Tree, Bagging of MLP, Decision Tree, Logistic Regression, MLP, Naive-Bayes, Random Forest and SVM.

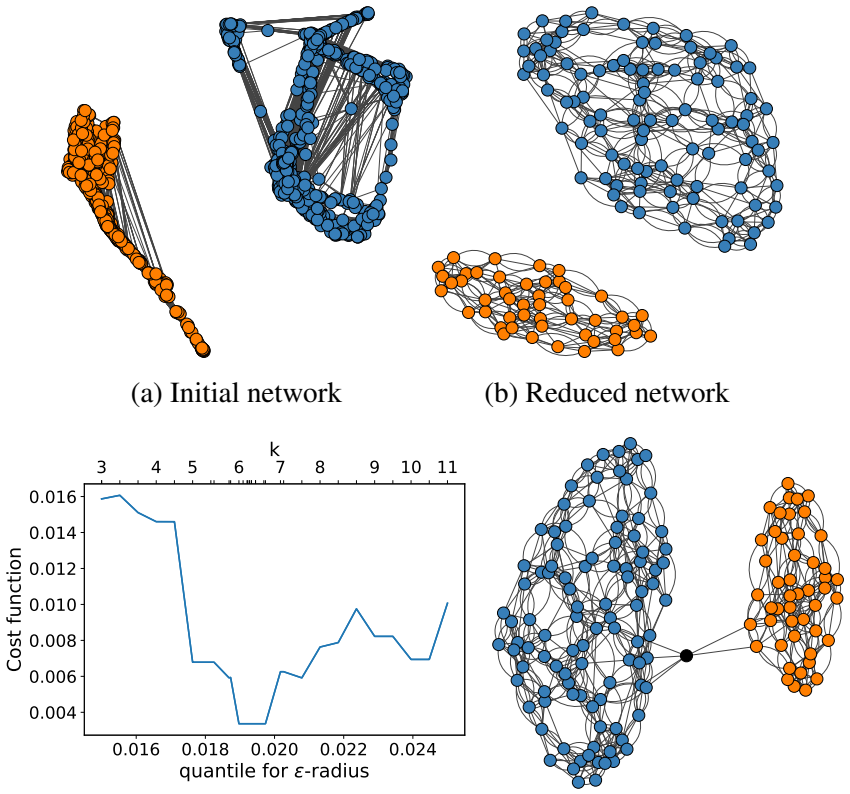
	MNBHL	**Ada**	**BagDT**	**BagMLP**	**DT**	**LR**	**MLP**	**N-B**	**RF**	**SVM**
Breast Cancer	2.296	0.248	0.083	5.551	0.022	0.021	0.529	0.002	0.081	0.064
Circles_00	1.926	0.193	0.032	1.167	0.001	0.001	0.120	0.001	0.027	0.001
Circles_02	1.864	0.124	0.029	1.176	0.001	0.001	0.123	0.001	0.028	0.001
Digits	13.495	0.333	0.189	21.581	0.026	0.300	2.887	0.005	0.120	0.803
Iris	3.601	0.126	0.028	1.401	0.001	0.001	0.146	0.001	0.026	0.001
Moons_00	1.469	0.128	0.026	1.158	0.001	0.001	0.135	0.001	0.027	0.001
Moons_02	1.578	0.129	0.028	1.173	0.001	0.001	0.126	0.001	0.029	0.001
Pima	5.290	0.180	0.063	7.561	0.008	0.020	0.959	0.001	0.075	0.071
Wine	5.058	0.137	0.032	0.677	0.002	0.003	0.033	0.001	0.029	0.005
Zoo	3.115	0.127	0.025	1.557	0.001	0.002	0.161	0.002	0.024	0.002

than the traditional ones, we believe its running time is still quite affordable. Note that, when compared to the Bagging of MLP model, the MNBHL model can even be faster on some datasets. We remember, again, that we opted for the network reduction when processing the Breast Cancer, Digits and Zoo datasets. Without this reduction, the running time for these three datasets would certainly be much higher than the timings shown in this table. However, as the results from Table 22 indicate, the network reduction does not seem to cause prejudice to the model's overall performance. In fact, when it comes to larger datasets, the network reduction option may actually increase the model's performance, by making use of only those data instances from the training dataset whose nodes the model identifies as being the most representative ones, for training purposes.

8.4.2 Experimental Results

In this subsection, we present the obtained results when applying the proposed MNBHL model to the COVID-19 dataset, along with a comparison of its performance with the ones achieved by traditional classification models, on the same data.

Figure 29 – Plots showing four different processing stages of the MNBHL proposed model, when applied to detect respiratory insufficiency signs.



(a) Initial network (b) Reduced network

(c) Cost function values (d) New data instance insertion

Source: Elaborated by the author.

We start by showing an example of the different processing stages from the proposed model, when applied to detect respiratory insufficiency signs. In Figure 29a, there is the initial network built by the model, i.e., the network formed by all training data instances. This network has 1,491 nodes in total (representing the patients), and two classes (with and without insufficiency signs), denoted by the blue and orange colors. In Figure 29b, we have the reduced version of this network, with only the 10% nodes (149 in total) with the highest betweenness values. The parameters ε and k used for building this network are the optimum ones, found by the model's cost function optimization (Figure 29c). In this case, the optimum values are the ε resulted from a 0.0192 quantile and $k = 7$, with a minimum cost of 0.00336. In the testing phase, the model makes use of these parameters when inserting a new data instance to be classified, which is denoted by the black color in Figure 29d. In this example, the new node's insertion affected both network components, and hence its label will be yielded by the class whose component is least impacted, in terms of the betweenness centrality.

In Table 24, we display the results obtained by all classification models under comparison, in terms of accuracy, on detecting signs of each type of insufficiency. Overall, all models are more successful on detecting signs of respiratory insufficiency, over other types, with most

Table 24 – Accuracy rates (%) for each insufficiency type, obtained by each classification technique. The values between parenthesis indicate the rank achieved by each technique, in each row.

	MNBHL	**Ada**	**BagDT**	**BagMLP**	**DT**	**LR**	**MLP**	**N-B**	**RF**	**SVM**
Any	67.9 (3)	67.9 (2)	63.5 (8)	67.1 (5)	59.9 (10)	68.1 (1)	67.0 (6)	67.5 (4)	64.4 (7)	62.6 (9)
Hepatic	64.2 (1)	62.7 (6)	58.4 (8)	63.7 (2)	54.7 (10)	63.2 (4)	63.0 (5)	63.5 (3)	59.4 (7)	56.1 (9)
Renal	70.6 (2)	68.3 (6)	65.1 (8)	69.8 (4)	59.6 (10)	71.0 (1)	69.4 (5)	70.2 (3)	65.3 (7)	64.4 (9)
Respiratory	76.1 (2)	75.8 (4)	71.3 (7)	76.1 (1)	66.2 (10)	76.0 (3)	75.3 (5)	75.2 (6)	70.8 (9)	71.2 (8)
Average Rank	1st	5th	8th	3rd	10th	2nd	6th	4th	7th	9th

Source: Research data.

of them achieving an accuracy of more than 70% on this task. The most difficult signs to be detected are those regarding the hepatic insufficiency, with an accuracy slightly over 60%, for most classifiers. The proposed MNBHL and Logistic Regression are the ones which achieved the best performances, considering all tasks, with an average rank of first and second places, respectively, followed by the BagMLP model.

Given the nature of the COVID-19 dataset, the input data may become more or less unbalanced in each age group. In this sense, younger patients, from age groups 00-20 and 21-40 years old, are expected to present a lower incidence of insufficiency cases than older patients, from age groups 60-80 and 80+ years old (see Table 20). For analyzing how such differences may affect the model's performance, we present, in Figure 30, the overall accuracy obtained by the MNBHL model, on each age group. The boxplots indicate that the model is able to achieve higher performances when analyzing data from older patients, reaching an accuracy of more than 90% for the 80+ years old age group.

People who are older than 60 years are in the high risk group for COVID-19. If any sign of insufficiency is detected in patients from this group, prompt measures should be taken by healthcare professionals, oftentimes by making use of mechanical ventilation. For this reason, we also evaluate, in Table 25, how the models perform, in terms of precision, sensitivity and F1 score, specifically on data from this group, when detecting signs from any type of insufficiency (hepatic, renal or respiratory). From this table, we see that the new MNBHL technique again obtains competitive results in comparison to classic ones.

8.5 Chapter Remarks

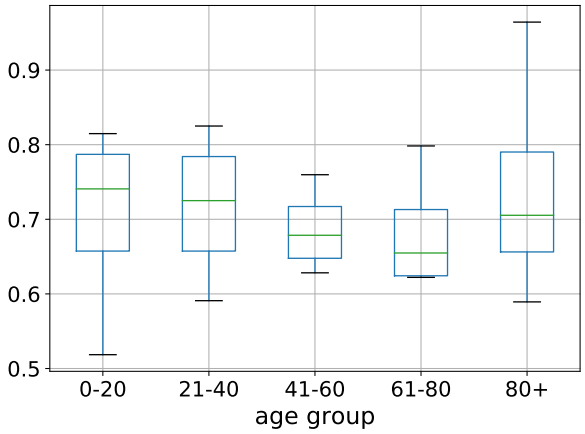
In this chapter, we present a modified network-based high-level classification technique that comprises two major improvements: (1) a network reduction method, to both reduce the model's running time and potentially enhance its performance, by selecting the network's most central nodes in the training phase, and (2) a cost function to optimize the network-building parameters ε and k. We initially evaluate the model by applying it to benchmark datasets and comparing it to traditional classification techniques, obtaining favorable results in this assessment. Afterwards, we applied the model to detect early insufficiency signs in COVID-19 patients, using

Table 25 – Precision, sensitivity and F1 score achieved by each technique, when detecting signs from any type of insufficiency (hepatic, renal or respiratory) in COVID-19 patients over 60 years old.

	Precision	**Sensitivity**	**F1 score**
MNBHL	0.891	0.982	0.933
Ada	0.884	0.993	0.934
BagDT	0.901	0.945	0.922
BagMLP	0.884	0.996	0.935
DT	0.894	0.869	0.882
LR	0.886	0.993	0.935
MLP	0.884	0.998	0.936
N-B	0.893	0.973	0.931
RF	0.897	0.963	0.928
SVM	0.882	0.975	0.925

Source: Research data.

Figure 30 – Boxplots of the overall performance, in terms of accuracy, achieved by the proposed technique on detecting insufficiency signs in COVID-19 patients, grouped by age.



Source: Elaborated by the author.

CBC test results as biomarkers. The experimental results obtained on this task are competitive, when compared to those achieved by classic and state-of-the-art classification techniques.

At this point, we would like to emphasize that the method proposed in this study cannot substitute specific medical tests for detecting signs of insufficiency in patients. Rather than that, the intention is to use the proposed method as an additional tool in the severity assessment of COVID-19 patients, thus helping healthcare professionals, specially those who work in regions with scarce material and human resources, to detect the severity of COVID-19 patients at high risk of complications, before the consequences become irreversible.

CHAPTER

9

CONCLUSIONS

9.1 Concluding Remarks

In this work, we have approached problems from very diverse fields through the development of novel machine learning techniques based on complex networks. We started by introducing two high level data classification techniques which were evaluated by applying them to benchmark datasets, both real and artificial ones, and by comparing their performances to those obtained by traditional classification models, on the same data. The first technique infers the label for a new data instance based on the detected impacts pattern from its insertion on the network topological structure, for each class. The main novelty of this model is in the fact that the low level method, which was required in the preceding technique, is no longer necessary for the classification process. The main novelty of the second technique proposed in this work is that it maps each data instance's attribute as a node in the network, instead of mapping each data instance as a node, as usual. Additionally, the technique is able to infer the labels for new data instances based solely on the modularity measure.

The other studies presented in this work are focused on offering solutions for specific and relevant problems encountered in the real world, also through the development of novel network-based machine learning techniques. The first problem concerns the analysis of voting data from Brazilian representatives, covering a time range of almost 30 years of legislative works. We demonstrate how the changes in the topological structure of the congresspeople temporal network reflect the main political changes happened in Brazil during the same period, in terms of the influence from each political party on the legislative decisions. Another finding of this research is with regard to the uncovering of an unexpected relationship between the voting history and convictions of corruption or other financial crimes among Brazilian representatives. This finding has motivated us to develop a predictive model for assessing the chances of a congressman for being convicted of corruption or other financial crimes in the future, solely based on how similar are his past votes and the voting record of already convicted congressmen.

In another investigation, we build an investment model to automate decision makings in the stock market, by adapting a model which was originally applied to detect periodicity in meteorological data. The model identifies up and down trends for a financial asset based on the price-variations network topological structure, and may automatically trigger a buying or selling order for the asset, accordingly. We have tested the proposed model by applying it to 10 of the most traded stocks from NYSE and Bovespa, and the preliminary obtained results were promising, with the model being able to outperform the "buy and hold" strategy, for the same period, in 15 out of the 20 considered cases.

The two remaining applications developed in this work are related to the COVID-19 pandemic, which arose during the period of our research. The first technique predicts new confirmed cases and deaths provoked by COVID-19 in a region through a temporal network built from the similarities between the COVID-19 variation curves in each region, pairwise. It works in the form of a multi correlation regression analysis, with the predicted values for a region being provided by considering the curve variation of regions within the same community in the temporal network. We evaluated the model by applying it to predict new COVID-19 cases and deaths for the 27 federal units of Brazil, on a weekly basis, in a total of 76 different predictions for a period of 3 weeks, including new cases and deaths. The obtained results indicate that, although our approach can be considered relatively simple – in the sense that it does not consider other important factors, such as local isolation policies, for instance – it was still able to outperform predictive models from similar studies, in terms of the mean absolute percentage error (MAPE).

In the last application, a technique for help monitoring COVID-19 patients was introduced. The model is able to detect early signs of hepatic, renal or respiratory insufficiency through the analysis of Complete Blood Count (CBC) test results. The main advantage of this technique is in the fact that CBC tests are easier to collect and cheaper than other additional tests, and hence the proposed model can be used as a supplementary tool to help healthcare professionals, specially those who work in regions with scarce material and human resources, to detect the severity of COVID-19 patients at high risk of complications. The high level data classification technique developed in this research is an improved version of the NBHL model, introduced in the first study, with two main novelties: (1) a network reduction method, to reduce processing time and, in some cases, also the noise in the input data, and (2) a parameters optimization procedure, based on a cost function. We believe these two improvements made in the high level data classification model are quite relevant, given that they provide a way of solving two issues left pending from the earlier models presented in this work: the parameters automation and the time complexity reduction.

9.2 Future Works

Regarding the first two high level network-based classification techniques presented in this work, the NBHL technique, presented in Chapter 3, can be extended by the addition of new network measures. Intuitively, it is expected that as more relevant measures are taken into account, the higher will be the classifier's robustness and efficiency. In the MBHL technique, presented in Chapter 4, although our current choice for adopting the Ridge Regression for the sake of generating the values of parameter γ has demonstrated, based on the obtained results, to be quite fair, we still consider that other forms of generating these values should definitely be explored in the future, such as the Elastic Net regression, for instance.

In the MNBHL model, presented in Chapter 8, we were able to successfully address two important points in both NBHL and MBHL models: (1) the network-building parameters optimization, through a cost function, so that they become adaptive, according to the features of each training dataset, and (2) the lowering of the model's memory consumption and running time, through a network reduction method used in the training phase, thus favoring the use of larger datasets for evaluation and application purposes. In the future, we plan to explore the possibility of extending the MNBHL classification technique by using additional centrality measures, in a combined form, both in the network reduction method and in the new node's insertion impact evaluation. We also plan to make applications on larger datasets, such as X-ray and CT images, to further assess the model's robustness.

With regard to the modeling and other application techniques, we plan to extract other measures from the congressmen temporal network, presented in Chapter 5, such as the temporal betweenness centrality, the temporal closeness centrality and bursting measure, to better understand its topological structure. Other network building methods can also be developed to include more relevant information from congresspeople, such as the federal state that each of them represents, original profession, sex, age, kinship among them, and so on. For the conviction prediction task, one can, for instance, filter the representatives voting records by types of bills and then identify which categories are more likely to be linked to corruption and other financial crimes. So we can alert people to pay more attention to those types of bills.

In the stock trading technique, presented in Chapter 6, one remaining open question is to uncover the main factors that contribute to a better performance of the model for a specific stock over others. The answer for it could be, for instance, in the topological properties of the network resulted from the trend detection phase, with some specific network structural patterns being more suitable to the model than others. In case there is such a correlation, then a preliminary stock filtering process, based on their resulting network topological properties, would help on determining the ones which are more likely to have their returns optimized by the model. Another possible improvement in this model, which would be interesting to test and check the results, is in turning its trend detection phase more dynamic. At the way it is now, it makes use of the first γ days of the stock's historical prices to generate the variation ranges network, which remains

unchanged during the whole operating phase. Additionally, updating the network regularly, as newer price variations are known during the operating phase, could lead to higher overall returns. Preferably, the network should be updated daily, but for testing purposes one can make use of a monthly update, i.e., around every 21 trading days. The parameter γ, in this case, could also be used as a sliding window, for saving processing time in the tests as well as for keeping the trend detection process concise and always up to date.

As for the COVID-19 prediction technique, presented in Chapter 7, we believe it can be applied to predict the evolution of the COVID-19 curve in each region more locally, such as in each city, for instance. The only requirement for such application is to have access to the COVID-19 evolution data from the cities to be predicted. Additionally, the proposed model can be improved, in order to allow it to perform predictions also in situations when the past variations of the time series to be predicted do not match any other time series in the dataset. Moreover, we also plan to explore other forms of analyzing the model's output data. One of the possibilities, in this sense, is to generate indexes for measuring how much the evolution of a given time series is correlated to the global and local averages evolution in the temporal network, for instance.

9.3 Publications During the Doctorate Period

During the doctorate period, eight articles have been generated, with two of them published in international journals, four of them in international conferences, one of them currently accepted as a book chapter and one published as a preprint research paper. The complete list of articles is provided below.

- Colliri, Tiago; Zhao, Liang. Predicting Corruption Convictions Among Brazilian Representatives Through a Voting-History Based Network. *Corruption Networks: Concepts and Applications*, 2021. Springer book chapter. (accepted)

- Colliri, Tiago; Zhao, Liang. Stock Market Trend Detection and Automatic Decision-Making Through a Network-Based Classification Model. *Natural Computing*, 2021, pp. 1-14, doi: 10.1007/s11047-020-09829-9.

- Colliri, Tiago; Delbem, Alexandre; Zhao, Liang. Predicting the Evolution of COVID-19 Cases and Deaths Through a Correlations-Based Temporal Network. *Cerri R., Prati R.C. (eds) Intelligent Systems. BRACIS 2020. Lecture Notes in Computer Science, vol 12320, pp. 397-411*. Springer, Cham. doi: 10.1007/978-3-030-61380-8_27.

- Colliri, T.; Weiguang, L.; Zhao, L. An Optimized Modularity-Based High Level Classification Model, 2020, Glasgow, United Kingdom. *2020 International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1-8, doi: 10.1109/IJCNN48605.2020.9206755.

- Colliri, Tiago; Zhao, Liang. Analyzing the Bills-Voting Dynamics and Predicting Corruption-Convictions Among Brazilian Congressmen Through Temporal Networks. *Scientific Reports*, v. 9, 16754 (1 - 11), 2019. https://doi.org/10.1038/s41598-019-53252-9.

- Colliri, Tiago; Zhao, Liang. A Network-Based Approach to Predict New Affected Regions and the Spread Evolution of COVID-19. *SSRN 3577663*, 2020.

- Colliri, Tiago; Zhao, Liang. A Network-Based Model for Optimizing Returns in the Stock Market, 2019, Salvador. *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, 2019, pp. 645-650, doi: 10.1109/BRACIS.2019.00118.

- Colliri, Tiago; Ji, Donghong; Pan, Heng; Zhao, Liang. A Network-Based High Level Data Classification Technique, 2018, Rio de Janeiro. *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8, doi: 10.1109/IJCNN.2018.8489081.

BIBLIOGRAPHY

ABBASI, A.; HOSSAIN, L.; LEYDESDORFF, L. Betweenness centrality as a driver of preferential attachment in the evolution of research collaboration networks. **Journal of Informetrics**, Elsevier, v. 6, n. 3, p. 403–412, 2012. Citation on page 41.

ABRAHAM, I.; DELLING, D.; GOLDBERG, A. V.; WERNECK, R. F. A hub-based labeling algorithm for shortest paths in road networks. In: SPRINGER. **International Symposium on Experimental Algorithms**. 2011. p. 230–241. Citation on page 40.

ADEBIYI, A. A.; ADEWUMI, A. O.; AYO, C. K. Comparison of ARIMA and artificial neural networks models for stock price prediction. **Journal of Applied Mathematics**, 2014. Available: <https://doi.org/10.1155/2014/614342>. Citations on pages 29 and 101.

AHLGREN, P.; JARNEVING, B.; ROUSSEAU, R. Requirements for a cocitation similarity measure, with special reference to Pearson's correlation coefficient. **Journal of the American Society for Information Science and Technology**, Wiley Online Library, v. 54, n. 6, p. 550–560, 2003. Citation on page 89.

AKIKI, T. J.; ABDALLAH, C. G. Determining the hierarchical architecture of the human brain using subject-level clustering of functional networks. **Scientific Reports**, Nature Publishing Group, v. 9, n. 1, p. 1–15, 2019. Citations on pages 29 and 99.

AL-QANESS, M. A.; EWEES, A. A.; FAN, H.; AZIZ, M. A. E. Optimization method for forecasting confirmed cases of COVID-19 in China. **Journal of Clinical Medicine**, Multidisciplinary Digital Publishing Institute, v. 9, n. 3, p. 674, 2020. Citations on pages 119 and 126.

ALBERT, R.; ALBERT, I.; NAKARADO, G. L. Structural vulnerability of the north american power grid. **Physical Review**, v. 69, n. 2, p. 025103, 2004. Citations on pages 38 and 100.

ALBERT, R.; BARABÁSI, A. L. Statistical mechanics of complex networks. **Reviews of Modern Physics**, v. 74, p. 47–97, 2002. Citation on page 38.

ALBERT, R.; JEONG, H.; BARABÁSI, A.-L. Error and attack tolerance of complex networks. **Nature**, Nature Publishing Group, v. 406, n. 6794, p. 378, 2000. Citations on pages 40 and 42.

ALPAYDIN, E. **Introduction to machine learning**. : The MIT Press, 2010. Citation on page 43.

ANDERSON, J.; BOTHELL, D.; BYRNE, M.; DOUGLASS, S.; LEBIERE, C.; QIN, Y. An integrated theory of the mind. **Psychological Review**, v. 111, p. 1036–60, 11 2004. Citation on page 27.

ANDERSON, J. R. **Learning and memory: An integrated approach**. : John Wiley & Sons Inc, 2000. Citation on page 27.

ANDERSON, R. M.; ANDERSON, B.; MAY, R. M. **Infectious diseases of humans: dynamics and control**. : Oxford university press, 1992. Citation on page 119.

ANDRIS, C.; LEE, D.; HAMILTON, M. J.; MARTINO, M.; GUNNING, C. E.; SELDEN, J. A. The rise of partisanship and super-cooperators in the U.S. House of Representatives. **PLoS One**, Public Library of Science, v. 10, n. 4, p. 1–14, 2015. Available: <https://doi.org/10.1371/journal.pone.0123507>. Citations on pages 29, 32, and 82.

ANGHINONI, L.; ZHAO, L.; JI, D.; PAN, H. Time series trend detection and forecasting using complex network topology analysis. **Neural Networks**, Elsevier, v. 117, p. 295–306, 2019. Citations on pages 44 and 102.

AREF, S.; NEAL, Z. Detecting coalitions by optimally partitioning signed networks of political collaboration. **Scientific Reports**, Nature Publishing Group, v. 10, n. 1, p. 1–10, 2020. Citation on page 119.

ARMSTRONG, E. Integrity, transparency and accountability in public administration: Recent trends, regional and international developments and emerging issues. **United Nations, Department of Economic and Social Affairs**, p. 1–10, 2005. Citation on page 82.

ARORA, P.; KUMAR, H.; PANIGRAHI, B. K. Prediction and analysis of COVID-19 positive cases using deep learning models: A descriptive case study of India. **Chaos, Solitons & Fractals**, Elsevier, p. 110017, 2020. Citations on pages 119 and 126.

ATKINSON, R. C.; SHIFFRIN, R. M. Human memory: A proposed system and its control processes. In: **Psychology of Learning and Motivation**. : Elsevier, 1968. v. 2, p. 89–195. Citation on page 27.

BACHELIER, L. **Théorie de la spéculation**. : Gauthier-Villars, 1900. Citation on page 100.

BACKES, A. R.; BRUNO, O. M. Shape classification using complex network and multi-scale fractal dimension. **Pattern Recognition Letters**, Elsevier, v. 31, n. 1, p. 44–51, 2010. Citations on pages 27, 28, and 45.

BACKES, A. R.; CASANOVA, D.; BRUNO, O. M. Texture analysis and classification: A complex network-based approach. **Information Sciences**, v. 219, p. 168 – 180, 2013. ISSN 0020-0255. Available: <http://www.sciencedirect.com/science/article/pii/S0020025512004677>. Citations on pages 28, 45, and 46.

BARABÁSI, A.-L. **Linked: The new science of networks**. : Perseus Books Group, 2002. Citation on page 100.

BARABÁSI, A.-L.; ALBERT, R. Emergence of scaling in random networks. **Science**, American Association for the Advancement of Science, v. 286, n. 5439, p. 509–512, 1999. Citations on pages 31, 38, 39, and 99.

BARABÁSI, A.-L. *et al.* **Network Science**. : Cambridge University Press, 2016. Citations on pages 38 and 39.

BARTHÉLEMY, M.; BARRAT, A.; PASTOR-SATORRAS, R.; VESPIGNANI, A. Dynamical patterns of epidemic outbreaks in complex heterogeneous networks. **Journal of Theoretical Biology**, Elsevier, v. 235, n. 2, p. 275–288, 2005. Citations on pages 30, 42, and 119.

BERLUSCONI, G.; CALDERONI, F.; PAROLINI, N.; VERANI, M.; PICCARDI, C. Link prediction in criminal networks: A tool for criminal intelligence analysis. **PLoS One**, Public Library of Science, v. 11, n. 4, 2016. ISSN 19326203. Citations on pages 83 and 98.

BERTON, L.; LOPES, A. de A.; VEGA-OLIVEROS, D. A. A comparison of graph construction methods for semi-supervised learning. In: **2018 International Joint Conference on Neural Networks (IJCNN)**. 2018. p. 1–8. Citations on pages 47 and 68.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. : Springer, 2006. Citation on page 43.

BLISS, C. A.; KLOUMANN, I. M.; HARRIS, K. D.; DANFORTH, C. M.; DODDS, P. S. Twitter reciprocal reply networks exhibit assortativity with respect to happiness. **Journal of Computational Science**, Elsevier, v. 3, n. 5, p. 388–397, 2012. Citation on page 41.

BOCCALETTI, S.; LATORA, V.; MORENO, Y.; CHAVEZ, M.; HWANG, D.-U. Complex networks: Structure and dynamics. **Physics Reports**, Elsevier, v. 424, n. 4, p. 175–308, 2006. Citation on page 37.

BRANDES, U. A faster algorithm for betweenness centrality. **Journal of Mathematical Sociology**, Taylor & Francis, v. 25, n. 2, p. 163–177, 2001. Citations on pages 30 and 136.

BRASIL.IO. **covid19 - dataset - Brasil.IO. https://data.brasil.io/dataset/covid19.html. [Accessed on May, 27, 2020]**. 2020. Citation on page 120.

BRAVAIS, A. **Analyse Mathématique. Sur les Probabilités des Erreurs de Situation d'un Point**. : Imprimerie Royale, 1844. Citation on page 142.

BREIMAN, L. Bagging predictors. **Machine Learning**, v. 24, n. 2, p. 123–140, 1996. Citations on pages 58, 76, and 140.

____. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001. Citations on pages 58, 76, and 140.

BULLMORE, E.; SPORNS, O. Complex brain networks: graph theoretical analysis of structural and functional systems. **Nature Reviews Neuroscience**, Nature Publishing Group, v. 10, n. 3, p. 186–198, 2009. Citation on page 42.

CÂMARA. **Dados Abertos. [Accessed on June 19, 2018]**. 2018. Available: <https://dadosabertos.camara.leg.br/>. Citation on page 83.

CAO, H.; LIN, T.; LI, Y.; ZHANG, H. Stock price pattern prediction based on complex network and machine learning. **Complexity**, v. 2019, p. 1–12, 2019. Citation on page 102.

CARNEIRO, M. G.; CHENG, R.; ZHAO, L.; JIN, Y. Particle swarm optimization for network-based data classification. **Neural Networks**, Elsevier, v. 110, p. 243–255, 2019. Citation on page 47.

CARNEIRO, M. G.; ZHAO, L. Organizational data classification based on the importance concept of complex networks. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 29, n. 8, p. 3361–3373, 2017. Citations on pages 27, 28, 44, 46, and 133.

CARRINGTON, P. J.; SCOTT, J.; WASSERMAN, S. **Models and methods in social network analysis.** Cambridge: Cambridge University Press, 2006. Citation on page 38.

CARTWRIGHT, D.; HARARY, F. Structural balance: a generalization of heider's theory. **Psychological Review**, American Psychological Association, v. 63, n. 5, p. 277, 1956. Citation on page 38.

CASIRAGHI, G. Multiplex network regression: How do relations drive interactions? **arXiv preprint arXiv:1702.02048**, 2017. Citation on page 44.

CAUCHY, A. L. Sur les polygones et polyedres, second mémoire. **J. Ecole Polytechnique**, v. 9, p. 87–98, 1813. Citation on page 37.

CHABRIER, A. Vehicle routing problem with elementary shortest path based column generation. **Computers & Operations Research**, Elsevier, v. 33, n. 10, p. 2972–2990, 2006. Citation on page 40.

CHANG, H.; SU, B.-B.; ZHOU, Y.-P.; HE, D.-R. Assortativity and act degree distribution of some collaboration networks. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 383, n. 2, p. 687–702, 2007. Citation on page 41.

CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. **Semi-supervised learning.** : The MIT Press, 2006. Citations on pages 43 and 44.

CHEN, D.; LÜ, L.; SHANG, M.-S.; ZHANG, Y.-C.; ZHOU, T. Identifying influential nodes in complex networks. **Physica A: Statistical Mechanics and its Applications**, Elsevier, v. 391, n. 4, p. 1777–1787, 2012. Citation on page 136.

CHIANG, W.-C.; ENKE, D.; WU, T.; WANG, R. An adaptive stock index trading decision support system. **Expert Systems with Applications**, Elsevier, v. 59, p. 195–207, 2016. Citation on page 103.

CLAUSET, A.; NEWMAN, M. E.; MOORE, C. Finding community structure in very large networks. **Physical Review E**, APS, v. 70, n. 6, p. 066111, 2004. Citation on page 73.

COLLIRI, T.; JI, D.; PAN, H.; ZHAO, L. A network-based high level data classification technique. In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. 2018. p. 1–8. Citations on pages 102 and 133.

COLLIRI, T.; ZHAO, L. Analyzing the bills-voting dynamics and predicting corruption-convictions among Brazilian congressmen through temporal networks. **Scientific Reports**, Nature Publishing Group, v. 9, n. 1, p. 1–11, 2019. Citation on page 119.

COSTA, L. d. F.; JR, O. N. O.; TRAVIESO, G.; RODRIGUES, F. A.; BOAS, P. R. V.; ANTIQUEIRA, L.; VIANA, M. P.; ROCHA, L. E. C. Analyzing and modeling real-world phenomena with complex networks: a survey of applications. **Advances in Physics**, Taylor & Francis, v. 60, n. 3, p. 329–412, 2011. Citations on pages 41 and 119.

COSTA, L. d. F.; RODRIGUES, F. A.; TRAVIESO, G.; BOAS, P. R. V. Characterization of complex networks: A survey of measurements. **Advances in Physics**, Taylor & Francis, v. 56, n. 1, p. 167–242, 2007. Citation on page 40.

DEZSŐ, Z.; BARABÁSI, A.-L. Halting viruses in scale-free networks. **Physical Review E**, APS, v. 65, n. 5, p. 055103, 2002. Citation on page 119.

DOMENICO, M. D.; SOLÉ-RIBALTA, A.; COZZO, E.; KIVELÄ, M.; MORENO, Y.; PORTER, M. A.; GÓMEZ, S.; ARENAS, A. Mathematical formulation of multilayer networks. **Physical Review X**, APS, v. 3, n. 4, p. 041022, 2013. Citation on page 119.

DONG, E.; DU, H.; GARDNER, L. An interactive web-based dashboard to track COVID-19 in real time. **The Lancet Infectious Diseases**, Elsevier, 2020. Citation on page 120.

DOROGOVTSEV, S. N.; MENDES, J. F. **Evolution of networks: From biological nets to the Internet and WWW**. : OUP Oxford, 2013. Citation on page 99.

EGGHE, L.; LEYDESDORFF, L. The relation between pearson's correlation coefficient r and salton's cosine measure. **Journal of the American Society for Information Science and Technology**, Wiley Online Library, v. 60, n. 5, p. 1027–1036, 2009. Citation on page 89.

ERDÖS, P.; RÉNYI, A. On the evolution of random graphs. **Publ. Math. Inst. Hung. Acad. Sci**, v. 5, n. 1, p. 17–60, 1960. Citation on page 38.

ESQUIVEL, A. V.; ROSVALL, M. Compression of flow can reveal overlapping-module organization in networks. **Physical Review X**, APS, v. 1, n. 2, p. 021025, 2011. Citation on page 89.

FALOUTSOS, M.; FALOUTSOS, P.; FALOUTSOS, C. On power-law relationships of the internet topology. **ACM SIGCOMM Computer Communication Review**, v. 29, n. 4, 1999. Citations on pages 38 and 99.

FANG, K.; SIVAKUMAR, B.; WOLDEMESKEL, F. M. Complex networks, community structure, and catchment classification in a large-scale river basin. **Journal of Hydrology**, v. 545, p. 478 – 493, 2017. Available: <http://www.sciencedirect.com/science/article/pii/S0022169416307715>. Citation on page 42.

FAPESP. **Research data metasearch. http://repositorio.uspdigital.usp.br/handle/item/243. [Accessed on August, 21, 2020]**. 2020. Citations on pages 132 and 141.

FERREIRA, L. N.; ZHAO, L. Detecting time series periodicity using complex networks. In: IEEE. **2014 Brazilian Conference on Intelligent Systems**. 2014. p. 402–407. Citations on pages 33, 34, 49, and 105.

FORTUNATO, S. Community detection in graphs. **Physics Reports**, Elsevier, v. 486, n. 3-5, p. 75–174, 2010. Citation on page 44.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In: SPRINGER. **European Conference on Computational Learning Theory**. 1995. p. 23–37. Citations on pages 58, 76, and 140.

GAMMERMAN, A.; VOVK, V.; VAPNIK, V. Learning by transduction. **arXiv preprint arXiv:1301.7375**, 1998. Citation on page 44.

GAO, X.; AN, H.; FANG, W.; HUANG, X.; LI, H.; ZHONG, W.; DING, Y. Transmission of linear regression patterns between time series: From relationship in time series to complex networks. **Physical Review E**, APS, v. 90, n. 1, p. 012818, 2014. Citation on page 44.

GELMAN, A.; HILL, J. **Data analysis using regression and multilevelhierarchical models**. : Cambridge University Press New York, NY, USA, 2007. Citations on pages 58, 76, and 140.

GILBERT, E. N. Random graphs. **Ann. Math. Statist.**, The Institute of Mathematical Statistics, v. 30, n. 4, p. 1141–1144, 12 1959. Available: <https://doi.org/10.1214/aoms/1177706098>. Citation on page 38.

GLEISER, P. M.; SPOORMAKER, V. I. Modelling hierarchical structure in functional brain networks. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society Publishing, v. 368, n. 1933, p. 5633–5644, 2010. Citations on pages 29 and 99.

GOH, K.-I.; OH, E.; KAHNG, B.; KIM, D. Betweenness centrality correlation in social networks. **Physical Review E**, APS, v. 67, n. 1, p. 017101, 2003. Citation on page 41.

GOZES, O.; FRID-ADAR, M.; GREENSPAN, H.; BROWNING, P. D.; ZHANG, H.; JI, W.; BERNHEIM, A.; SIEGEL, E. Rapid AI development cycle for the coronavirus (COVID-19) pandemic: Initial results for automated detection & patient monitoring using deep learning CT image analysis. **arXiv preprint arXiv:2003.05037**, 2020. Citation on page 132.

GUIMERA, R.; AMARAL, L. A. N. Functional cartography of complex metabolic networks. **Nature**, Nature Publishing Group, v. 433, n. 7028, p. 895, 2005. Citations on pages 29, 34, 42, 87, 90, and 100.

GUNS, R. Link prediction. In: **Measuring scholarly impact**. : Springer, 2014. p. 35–55. Citation on page 95.

GURESEN, E.; KAYAKUTLU, G.; DAIM, T. U. Using artificial neural network models in stock market index prediction. **Expert Systems with Applications**, v. 38, n. 8, p. 10389 – 10397, 2011. ISSN 0957-4174. Citations on pages 33 and 102.

HAGMANN, P.; CAMMOUN, L.; GIGANDET, X.; MEULI, R.; HONEY, C. J.; WEDEEN, V. J.; SPORNS, O. Mapping the structural core of human cerebral cortex. **PLoS Biol**, Public Library of Science, v. 6, n. 7, p. e159, 2008. Citations on pages 29 and 99.

HARARI, Y. N. **Sapiens: A brief history of humankind**. : Random House, 2014. Citation on page 117.

HAYEK, F. A. The use of knowledge in society. **The American Economic Review**, JSTOR, p. 519–530, 1945. Citation on page 100.

HINTON, G. E. Connectionist learning procedures. **Artificial Intelligence**, Elsevier, v. 40, n. 1-3, p. 185–234, 1989. Citations on pages 58, 76, 102, and 140.

HOERL, A. E.; KENNARD, R. W. Ridge regression: Biased estimation for nonorthogonal problems. **Technometrics**, Taylor & Francis Group, v. 12, n. 1, p. 55–67, 1970. Citation on page 74.

HOLME, P.; SARAMÄKI, J. Temporal networks. **Physics Reports**, Elsevier, v. 519, n. 3, p. 97–125, 2012. Citation on page 86.

HUANG, W.; NAKAMORI, Y.; WANG, S. Y. Forecasting stock market movement direction with support vector machine. **Computers & Operations Research**, v. 32, n. 10, p. 2513–2522, 2005. Citations on pages 29 and 101.

HUILLIER, S. Mémoire sur la polyèdrométrie. **Annales de Mathématiques**, n. 3, p. 169–189, 1861. Citation on page 37.

HULOVATYY, Y.; CHEN, H.; MILENKOVIĆ, T. Exploring the structure and function of temporal networks with dynamic graphlets. **Bioinformatics**, Oxford University Press, v. 31, n. 12, p. i171–i180, 2015. Citation on page 86.

JAEGER, P. T.; BERTOT, J. C. Transparency and technological change: Ensuring equal and sustained public access to government information. **Government Information Quarterly**, Elsevier, v. 27, n. 4, p. 371–376, 2010. Citation on page 82.

JÄKEL, F.; SCHÖLKOPF, B.; WICHMANN, F. A. Generalization and similarity in exemplar models of categorization: Insights from machine learning. **Psychonomic Bulletin & Review**, Springer, v. 15, n. 2, p. 256–271, 2008. Citation on page 27.

JEBARA, T.; WANG, J.; CHANG, S.-F. Graph construction and b-matching for semi-supervised learning. In: **Proceedings of the 26th Annual International Conference on Machine Learning**. 2009. p. 441–448. Citation on page 47.

JI, L.-J.; ZHANG, Z.; GUO, T. To buy or to sell: Cultural differences in stock market decisions based on price trends. **Journal of Behavioral Decision Making**, Wiley Online Library, v. 21, n. 4, p. 399–413, 2008. Citation on page 102.

KARYPIS, G.; HAN, E.-H.; KUMAR, V. Chameleon: hierarchical clustering using dynamic modeling. **Computer**, v. 32, n. 8, p. 68–75, 1999. Citation on page 44.

KIRKLAND, J. H.; GROSS, J. H. Measurement and theory in legislative networks: The evolving topology of Congressional collaboration. **Social Networks**, v. 36, n. 1, p. 97–109, 2014. ISSN 03788733. Citation on page 82.

KIVELÄ, M.; ARENAS, A.; BARTHELEMY, M.; GLEESON, J. P.; MORENO, Y.; PORTER, M. A. Multilayer networks. **Journal of Complex Networks**, Oxford University Press, v. 2, n. 3, p. 203–271, 2014. Citation on page 119.

KOSTOPOULOS, G.; KARLOS, S.; KOTSIANTIS, S.; RAGOS, O. Semi-supervised regression: A recent review. **Journal of Intelligent & Fuzzy Systems**, IOS Press, v. 35, n. 2, p. 1483–1500, 2018. Citation on page 120.

KROT, A.; PROKHORENKOVA, L. O. Local clustering coefficient in generalized preferential attachment models. In: SPRINGER. **International Workshop on Algorithms and Models for the Web-Graph**. 2015. p. 15–28. Citation on page 41.

KRUSCHKE, J. K. Alcove: an exemplar-based connectionist model of category learning. **Psychological Review**, American Psychological Association, v. 99, n. 1, p. 22, 1992. Citation on page 27.

KULLBACK, S.; LEIBLER, R. A. On information and sufficiency. **The Annals of Mathematical Statistics**, JSTOR, v. 22, n. 1, p. 79–86, 1951. Citations on pages 30 and 139.

LAKE, B. M.; LAWRENCE, N. D.; TENENBAUM, J. B. The emergence of organizing structure in conceptual representation. **Cognitive Science**, v. 42, n. S3, p. 809–832, 2018. Citation on page 27.

LAKE, B. M.; ULLMAN, T. D.; TENENBAUM, J. B.; GERSHMAN, S. J. Building machines that learn and think like people. **Behavioral and Brain Sciences**, Cambridge University Press, v. 40, p. e253, 2017. Citation on page 27.

LEE, K.; JO, G. Expert system for predicting stock market timing using a candlestick chart. **Expert Systems with Applications**, v. 16, n. 4, p. 357 – 364, 1999. ISSN 0957-4174. Citations on pages 29 and 101.

LIBEN-NOWELL, D.; KLEINBERG, J. The link-prediction problem for social networks. **Journal of the American Society for Information Science and Technology**, Wiley Online Library, v. 58, n. 7, p. 1019–1031, 2007. Citations on pages 89 and 95.

LICHMAN, M. **UCI Machine Learning Repository**. 2013. Available: <http://archive.ics.uci.edu/ml>. Citations on pages 61, 76, and 141.

LIU, W.; LÜ, L. Link prediction based on local random walk. **EPL (Europhysics Letters)**, IOP Publishing, v. 89, n. 5, p. 58007, 2010. Citation on page 41.

LIU, W.; SUZUMURA, T.; JI, H.; HU, G. Finding overlapping communities in multilayer networks. **PLoS One**, Public Library of Science, v. 13, n. 4, p. e0188747, 2018. Citation on page 44.

LOGLISCI, C.; MALERBA, D. Leveraging temporal autocorrelation of historical data for improving accuracy in network regression. **Statistical Analysis and Data Mining: The ASA Data Science Journal**, Wiley Online Library, v. 10, n. 1, p. 40–53, 2017. Citation on page 44.

LOHMANN, G.; MARGULIES, D. S.; HORSTMANN, A.; PLEGER, B.; LEPSIEN, J.; GOLDHAHN, D.; SCHLOEGL, H.; STUMVOLL, M.; VILLRINGER, A.; TURNER, R. Eigenvector centrality mapping for analyzing connectivity patterns in fMRI data of the human brain. **PLoS One**, Public Library of Science, v. 5, n. 4, p. e10232, 2010. Citation on page 41.

LUNA-PLA, I.; NICOLÁS-CARLOCK, J. R. Corruption and complexity: a scientific framework for the analysis of corruption networks. **Applied Network Science**, Springer, v. 5, n. 1, p. 1–18, 2020. Citations on pages 83 and 119.

MALKIEL, B. G.; FAMA, E. F. Efficient capital markets: A review of theory and empirical work. **The Journal of Finance**, Wiley Online Library, v. 25, n. 2, p. 383–417, 1970. Citation on page 100.

MASO, C. D.; POMPA, G.; PULIGA, M.; RIOTTA, G.; CHESSA, A. Voting behavior, coalitions and government strength through a complex network analysis. **PLoS One**, Public Library of Science, v. 9, n. 12, 2014. ISSN 19326203. Citations on pages 29, 32, and 82.

MEO, P. D.; FERRARA, E.; FIUMARA, G.; PROVETTI, A. Generalized Louvain method for community detection in large networks. In: IEEE. **2011 11th International Conference on Intelligent Systems Design and Applications**. 2011. p. 88–93. Citation on page 87.

METSKY, H. C.; FREIJE, C. A.; KOSOKO-THORODDSEN, T.-S. F.; SABETI, P. C.; MYHRVOLD, C. CRISPR-based surveillance for COVID-19 using genomically-comprehensive machine learning design. **BioRxiv**, Cold Spring Harbor Laboratory, 2020. Citation on page 132.

MITCHELL, M. Complex systems: Network thinking. **Artificial Intelligence**, Elsevier, v. 170, n. 18, p. 1194–1212, 2006. Citation on page 100.

MITCHELL, T. M. **Machine Learning.** USA: New York, NY: McGraw-Hill, Inc, 1997. Citation on page 43.

MONTOYA, J. M.; SOLé, R. V. Small world patterns in food webs. **Journal of Theoretical Biology**, v. 214, n. 3, p. 405–412, 2002. Citations on pages 38 and 99.

MOODY, J.; MUCHA, P. J. Portrait of political party polarization. **Network Science**, Cambridge University Press, v. 1, n. 1, p. 119–121, 2013. Citations on pages 29, 32, and 82.

MORENO, Y.; NEKOVEE, M.; PACHECO, A. F. Dynamics of rumor spreading in complex networks. **Physical Review E**, APS, v. 69, n. 6, p. 066130, 2004. Citation on page 42.

MOTTER, A. E.; LAI, Y.-C. Cascade-based attacks on complex networks. **Physical Review E**, APS, v. 66, n. 6, p. 065102, 2002. Citation on page 100.

NEAL, Z. P. A sign of the times? Weak and strong polarization in the US Congress, 1973–2016. **Social Networks**, Elsevier, 2018. Citation on page 82.

NETO, F. A.; ZHAO, L. High level data classification based on network entropy. **Neural Networks (IJCNN), The 2013 International Joint Conference on. IEEE**, 2013. Citation on page 44.

NEWMAN, M. E. Mixing patterns in networks. **Physical Review E**, APS, v. 67, n. 2, p. 026126, 2003. Citations on pages 41 and 57.

____. Fast algorithm for detecting community structure in networks. **Physical Review E**, APS, v. 69, n. 6, p. 066133, 2004. Citations on pages 110 and 121.

____. Modularity and community structure in networks. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 103, n. 23, p. 8577–8582, 2006. Citations on pages 28, 41, and 67.

NI, B.; YAN, S.; KASSIM, A. Learning a propagable graph for semisupervised learning: Classification and regression. **Knowledge and Data Engineering, IEEE Transactions on**, v. 24, n. 1, p. 114–126, 2012. Citation on page 44.

OSLER, C. L. Support for resistance: technical analysis and intraday exchange rates. **Economic Policy Review**, v. 6, n. 2, 2000. Citation on page 103.

PAGE, L.; BRIN, S.; MOTWANI, R.; WINOGRAD, T. **The PageRank citation ranking: Bringing order to the web.** Stanford InfoLab, 1999. Citations on pages 41, 46, and 89.

PALLA, G.; DERÉNYI, I.; FARKAS, I.; VICSEK, T. Uncovering the overlapping community structure of complex networks in nature and society. **Nature**, Nature Publishing Group, v. 435, n. 7043, p. 814–818, 2005. Citations on pages 44 and 99.

PASTOR-SATORRAS, R.; CASTELLANO, C.; MIEGHEM, P. V.; VESPIGNANI, A. Epidemic processes in complex networks. **Reviews of Modern Physics**, APS, v. 87, n. 3, p. 925, 2015. Citation on page 30.

PASTOR-SATORRAS, R.; VESPIGNANI, A. Epidemic spreading in scale-free networks. **Physical Review Letters**, APS, v. 86, n. 14, p. 3200, 2001. Citations on pages 30, 100, and 119.

____. Immunization of complex networks. **Physical Review E**, APS, v. 65, n. 3, p. 036104, 2002. Citations on pages 100 and 119.

PEARSON, K. Note on regression and inheritance in the case of two parents. **Proceedings of the Royal Society of London**, The Royal Society London, v. 58, n. 347-352, p. 240–242, 1895. Citation on page 142.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A.; COURNAPEAU, D.; BRUCHER, M.; PERROT, M.; DUCHESNAY, E. Scikit-learn: Machine learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011. Citations on pages 58, 59, 61, 76, and 140.

PONS, P.; LATAPY, M. Computing communities in large networks using random walks. In: SPRINGER. **International Symposium on Computer and Information Sciences**. 2005. p. 284–293. Citation on page 121.

POOLE, D. L.; MACKWORTH, A. K.; GOEBEL, R. **Computational intelligence: a logical approach**. : Oxford University Press New York, 1998. Citation on page 37.

REGNAULT, J. **Calcul des chances et philosophie de la bourse**. : Mallet-Bachelier, 1863. Citation on page 100.

RIBEIRO, H. V.; ALVES, L. G.; MARTINS, A. F.; LENZI, E. K.; PERC, M. The dynamical structure of political corruption networks. **Journal of Complex Networks**, Oxford University Press, v. 6, n. 6, p. 989–1003, 2018. Citations on pages 42, 83, 92, 93, 97, and 98.

RISH, I. An empirical study of the naive bayes classifier. **IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence**, v. 3, n. 22, 2001. IBM New York. Citations on pages 58, 76, 102, and 140.

ROBERTS, H. V. Stock-market "patterns" and financial analysis: methodological suggestions. **The Journal of Finance**, JSTOR, v. 14, n. 1, p. 1–10, 1959. Citation on page 100.

ROHBAN, M. H.; RABIEE, H. R. Supervised neighborhood graph construction for semi-supervised classification. **Pattern Recognition**, Elsevier, v. 45, n. 4, p. 1363–1372, 2012. Citation on page 47.

ROSSI, R. G.; LOPES, A. de A.; REZENDE, S. O. Optimization and label propagation in bipartite heterogeneous networks to improve transductive classification of texts. **Information Processing & Management**, Elsevier, v. 52, n. 2, p. 217–257, 2016. Citation on page 44.

RUSSELL, S. J.; NORVIG, P. **Artificial intelligence: a modern approach**. : Malaysia; Pearson Education Limited,, 2016. Citation on page 37.

SAFAVIN, S. R.; LANDGREBE, D. A survey of decision tree classifier methodology. **IEEE Trans. Syst., Man, Cybern.**, v. 21, n. 3, p. 660–674, 1991. Citations on pages 58, 76, 102, and 140.

SALTON, G.; MCGILL, M. J. **Introduction to modern information retrieval**. : McGraw-Hill, Inc., 1986. Citation on page 89.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of Research and Development**, IBM, v. 3, n. 3, p. 210–229, 1959. Citation on page 43.

SCHAEFFER, S. E. Graph clustering. **Computer Science Review**, v. 1, n. 1, p. 27–64, 2007. Citation on page 44.

SEITZ, A. R.; KIM, D.; WATANABE, T. Rewards evoke learning of unconsciously processed visual stimuli in adult humans. **Neuron**, Elsevier, v. 61, n. 5, p. 700–707, 2009. Citation on page 27.

SHAN, F.; GAO, Y.; WANG, J.; SHI, W.; SHI, N.; HAN, M.; XUE, Z.; SHI, Y. Lung infection quantification of COVID-19 in CT images with deep learning. **arXiv preprint arXiv:2003.04655**, 2020. Citation on page 132.

SHANNON, C. E. A mathematical theory of communication. **The Bell System Technical Journal**, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948. Citation on page 139.

SHARPE, W. F. Mutual fund performance. **The Journal of Business**, JSTOR, v. 39, n. 1, p. 119–138, 1966. Citation on page 115.

SILVA, T. C.; ZHAO, L. Network-based high level data classification. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 23, n. 6, p. 954–970, 2012. Citations on pages 28, 32, 33, 34, 44, 45, 46, 47, 51, 52, 53, 58, 102, 133, and 136.

____. Network-based stochastic semisupervised learning. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 23, n. 3, p. 451–466., 2012. Citation on page 44.

____. Stochastic competitive learning in complex networks. **Neural Networks and Learning Systems, IEEE Transactions on**, v. 23, n. 3, p. 385–398, 2012. Citation on page 44.

____. High-level pattern-based classification via tourist walks in networks. **Information Sciences**, v. 294, p. 109–126, 2015. Citations on pages 27, 28, 34, 46, 52, and 53.

____. **Machine Learning in Complex Networks.** : Heidelberg: Springer, 2016. Citations on pages 33, 44, and 45.

SILVA, T. C.; ZHAO, L.; CUPERTINO, T. H. Handwritten data clustering using agents competition in networks. **J. Math. Imaging Vis.**, v. 45, n. 3, p. 264–276, 2013. Citation on page 44.

SOUSA, C. A. R. de; REZENDE, S. O.; BATISTA, G. E. Influence of graph construction on semi-supervised learning. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. 2013. p. 160–175. Citation on page 47.

SPICER, J.; SANBORN, A. N. What does the mind learn? a comparison of human and machine learning representations. **Current Opinion in Neurobiology**, v. 55, p. 97 – 102, 2019. ISSN 0959-4388. Machine Learning, Big Data, and Neuroscience. Citation on page 27.

SPORNS, O. Network analysis, complexity, and brain function. **Complexity**, v. 8, n. 1, p. 56–60, 2002. Citations on pages 38 and 99.

SPORNS, O.; TONONI, G.; KÖTTER, R. The human connectome: a structural description of the human brain. **PLoS Comput Biol**, Public Library of Science, v. 1, n. 4, p. e42, 2005. Citation on page 42.

STF. **Processos. https://portal.stf.jus.br/. [Accessed on October, 22, 2019]**. 2019. Citation on page 84.

TALUKDAR, P. P.; CRAMMER, K. New regularized algorithms for transductive learning. In: SPRINGER. **Joint European Conference on Machine Learning and Knowledge Discovery in Databases**. 2009. p. 442–457. Citation on page 44.

THOMPSON, W. H.; BRANTEFORS, P.; FRANSSON, P. From static to temporal network theory: Applications to functional brain connectivity. **Network Neuroscience**, v. 1, n. 2, p. 69–99, 2017. Citations on pages 86 and 119.

VALEJO, A.; FERREIRA, V.; FABBRI, R.; OLIVEIRA, M. C. F. d.; LOPES, A. d. A. A critical survey of the multilevel method in complex networks. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 53, n. 2, p. 1–35, 2020. Citation on page 136.

VANDERMONDE, A.-T. Remarques sur les problèmes de situation. **Mémoires de l'Académie Royale des Sciences (Paris)**, v. 2, p. 566–574, 1771. Citation on page 37.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**. : New York: Springer, 2000. Citations on pages 58, 76, 101, and 140.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. **arXiv preprint arXiv:1706.03762**, 2017. Citations on pages 32, 68, and 74.

VEGA-OLIVEROS, D. A.; BERTON, L.; EBERLE, A. M.; LOPES, A. de A.; ZHAO, L. Regular graph construction for semi-supervised learning. In: IOP PUBLISHING. **Journal of Physics: Conference series**. 2014. v. 490, n. 1, p. 012022. Citation on page 44.

VERRI, F. A. N.; URIO, P. R.; ZHAO, L. Network unfolding map by vertex-edge dynamics modeling. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 29, n. 2, p. 405–418, 2016. Citation on page 44.

VICTOR, J. N.; MONTGOMERY, A. H.; LUBELL, M. **The Oxford Handbook of Political Networks**. : Oxford University Press, 2017. Citation on page 82.

WACHS, J.; YASSERI, T.; LENGYEL, B.; KERTÉSZ, J. Social capital predicts corruption risk in towns. **Royal Society Open Science**, The Royal Society, v. 6, n. 4, p. 182103, 2019. Citation on page 82.

WANG, J.-L.; CHAN, S.-H. Stock market trading rule discovery using pattern recognition and technical analysis. **Expert Systems with Applications**, Elsevier, v. 33, n. 2, p. 304–315, 2007. Citation on page 100.

WANG, X. F.; CHEN, G. Complex networks: small-world, scale-free and beyond. **IEEE Circuits and Systems Magazine**, v. 3, n. 1, p. 6–20, 2003. Citation on page 100.

WANG, Y.; HU, M.; LI, Q.; ZHANG, X.-P.; ZHAI, G.; YAO, N. Abnormal respiratory patterns classifier may contribute to large-scale screening of people infected with COVID-19 in an accurate and unobtrusive manner. **arXiv preprint arXiv:2002.05534**, 2020. Citation on page 132.

WATTS, D. J. **Six degrees: The science of a connected age**. : WW Norton & Company, 2004. Citation on page 100.

WATTS, D. J.; STROGATZ, S. H. Collective dynamics of "small-world" networks. **Nature**, Nature Publishing Group, v. 393, n. 6684, p. 440, 1998. Citations on pages 31, 38, 39, and 99.

WAUGH, A. S.; PEI, L.; FOWLER, J. H.; MUCHA, P. J.; PORTER, M. A. Party polarization in congress: A network science approach. **arXiv preprint arXiv:0907.3509**, 2009. Citations on pages 29, 32, and 82.

WEST, G. B.; BROWN, J. H.; ENQUIST, B. J. A general model for the structure, and allometry of plant vascular systems. **Nature**, v. 400, p. 125–126, 2009. Citations on pages 38 and 99.

WORLDOMETER. **Coronavirus Update. https://www.worldometers.info/coronavirus/. [Accessed on February, 1, 2021]**. 2021. Citation on page 33.

XUBO, G.; QIUSHENG, Z.; VEGA-OLIVEROS, D. A.; LEANDRO, A.; ZHAO, L. Temporal network pattern identification by community modelling. **Scientific Reports**, Nature Publishing Group, v. 10, n. 1, 2020. Citation on page 119.

YAN, L.; ZHANG, H.-T.; GONCALVES, J.; XIAO, Y.; WANG, M.; GUO, Y.; SUN, C.; TANG, X.; JING, L.; ZHANG, M. *et al.* An interpretable mortality prediction model for COVID-19 patients. **Nature Machine Intelligence**, Nature Publishing Group, p. 1–6, 2020. Citation on page 132.

ZHU, X. J. **Semi-supervised learning literature survey**. University of Wisconsin-Madison Department of Computer Sciences, Madison, Wisconsin, 2005. Citation on page 44.


ICMC
USP
SÃO CARLOS