FEDERAL UNIVERSITY OF UBERLÂNDIA
FACULTY OF ELECTRICAL ENGINEERING

GABRIEL AUGUSTO ROSA

Understanding and Predicting Interruptions Index of Medium Voltage Customers Using Fully Connected Networks

Uberlandia, Brazil
2022

GABRIEL AUGUSTO ROSA

Understanding and Predicting Interruptions Index of Medium Voltage Customers Using Fully Connected Networks

Dissertation presented to the Faculty of Electrical Engineering at the Federal University of Uberlandia as a partial requirement for obtaining the degree Master of Science

Concentration area: Neural Networks
Advisor: Prof. Dr. Keiji Yamanaka

Uberlandia
2022

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

R788 Rosa, Gabriel Augusto, 1993-
2022 Understanding and predicting interruptions index of medium voltage customers using fully connected networks. [recurso eletrônico] / Gabriel Augusto Rosa. - 2022.

Orientador: Keiji Yamanaka.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Elétrica.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2022.239
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia elétrica. I. Yamanaka, Keiji,1956-, (Orient.). II. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

GABRIEL AUGUSTO ROSA

Understanding and Predicting Interruptions Index of Medium Voltage Customers Using Fully Connected Networks

Dissertation presented to the Faculty of Electrical Engineering at the Federal University of Uberlandia as a partial requirement for obtaining the degree Master of Science

Concentration area: Neural Networks

Uberlandia, Brazil, 2022

Examination Board:

Prof. Dr. Keiji Yamanaka – Advisor – (UFU)

Prof. Dr. Alexandre Cardoso – (UFU)

Prof. Dr. Alan Petrônio Pinheiro – (UFU)

Prof. Dr. Elder Vicente de Paulo Sobrinho – (UFTM)

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Engenharia Elétrica
Av. João Naves de Ávila, 2121, Bloco 3N - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4707 - www.posgrad.feelt.ufu.br - copel@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Engenharia Elétrica				
Defesa de:	Dissertação de Mestrado Acadêmico, 772, PPGEELT				
Data:	Cinco de maio de dois mil e vinte e dois	Hora de início:	09:30	Hora de encerramento:	11:30
Matrícula do Discente:	11922EEL018				
Nome do Discente:	Gabriel Augusto Rosa				
Título do Trabalho:	Understanding and predicting interruptions index of medium voltage customers using fully connected networks.				
Área de concentração:	Processamento da informação				
Linha de pesquisa:	Inteligência Artificial				
Projeto de Pesquisa de vinculação:	Coordenador do projeto: Keiji Yamanaka. Título do projeto: Estudo e Aplicações de Técnicas de Inteligência Computacional. Agência financiadora:__ Número do processo na agência financiadora: ___Vigência do projeto: 2010 - atual.				

Reuniu-se por meio de videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Engenharia Elétrica, assim composta: Professores Doutores: Alexandre Cardoso - FEELT/UFU; Alan Petrônio Pinheiro - FEELT/UFU; Elder Vicente de Paulo Sobrinho - UFTM; Keiji Yamanaka - FEELT/UFU, orientador(a) do(a) candidato(a).

Iniciando os trabalhos o(a) presidente da mesa, Dr(a). Keiji Yamanaka, apresentou a Comissão Examinadora e o candidato(a), agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos(às) examinadores(as), que passaram a arguir o(a) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato(a):

Aprovado(a).

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Elder Vicente de Paulo Sobrinho**, **Usuário Externo**, em 05/05/2022, às 12:03, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Keiji Yamanaka**, **Professor(a) do Magistério Superior**, em 05/05/2022, às 14:32, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alan Petronio Pinheiro**, **Professor(a) do Magistério Superior**, em 07/05/2022, às 20:53, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Alexandre Cardoso**, **Professor(a) do Magistério Superior**, em 10/05/2022, às 14:45, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3557220** e o código CRC **D88A5C73**.

Referência: Processo nº 23117.029794/2022-03 SEI nº 3557220

This work is dedicated to my family and all electrical engineering department at Uberlandia Federal University.

ACKNOWLEDGEMENTS

To my wife Luiza, for her love, companionship, and encouragement.

To my parents, João and Terezinha, and my brother Matheus, for the love, advice, and encouragement they always offered me.

To my advisor Prof. Dr. Keiji Yamanaka, for his help, attention, and trust.

To my colleague Daniel Oliveira, for his support and companionship throughout the course.

To all the teachers and those who one day teach me and contributed to my development.

To the Postgraduate Program in Electrical Engineering, for making the work possible.

Professor Alan and my colleague Daniel will allow me to use the database and compare myself in this research.

To all my friends, always with me.

Research financed by the ANEEL R&D project Nº 05160-1805 / 2018, between CEB and UFU, and with partial support from CNPQ, process number 135168/2019-8.

"Sometimes it is the people no one imagines anything of who do the things that no one can imagine." (Alan Turing)

ABSTRACT

The losses caused by the lack of electricity typically exceed the cost of the electricity itself. Improving power quality is a way to reduce or avoid loss of production in the industry, prevent fires or explosions, and minimize damages to industrial equipment. Therefore, finding customers that probably will have interruptions in advance will generate value for both the company and customers. The purpose of this study is to analyze data from units that consume electricity using neural networks and decision trees, such as self-organizing maps, CHAID and CART, and using fully connected neural networks to predict the interruption index for the next year. The results reveal an important space for improvements such as the connection between non-compliance of established indicators over time and specific points of electrical network with problems. That way supports the concessionaries to manage their infrastructure to get a better quality of the electric power network.

Keywords: Fully Connected Networks, Power Quality, Interruptions on Medium Voltages.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

1. ABBREVIATIONS:

AUC – Area Under the Curve

CAPEX - Capital Expenditure

CART - Classification and Regression

CHAID - Chi-square automatic interaction detection

CNNs - Convolutional Neural Networks

CO1 - Commercial

CO2 - Transport Services, Except Electric Traction

CO3 - Communications and Telecommunications Services

CO9 - Other Services and Other Activities

CP - Public consultation

DT – Decision Trees

FFNNs - Feedforward Perceptron Multi-Layer Neural Networks

FN – False Negative

FNNs - Fully Connected Neural networks

FP – False Positive

IN - Industrial

NU – Non-Urban

OPEX - Operational Expenditure

PP1 - Federal Public Power

PP2 - State or District Government

RE1 - Residential

RL – Reinforming Learning

RNNs - Recurrent Neural Networks

RU1 - Rural Agriculture

RU1B - Rural Agriculture (water pumping service intended for irrigation activity)

RU5 - Agro-Industrial

RU6 - Public Service of Rural Irrigation

SOMs – Self Organizing Maps

SP1 - Electric Traction

SP2 - Water, Sewage and Sanitation

TN – True Negative

TP – True Positive

UB - Urban

2. ACRONYMS

ANEEL - Agência Nacional de Energia Elétrica

ABRADEE - Associação Brasileira de Distribuidores de Energia Elétrica

CNAE - National Classification of Economic Activities

CNPQ - Conselho Nacional de Desenvolvimento Científico e Tecnológico

BDGD - Base de Dados Geográfica da Distribuidora

IBGE - Instituto Brasileiro de Geografia e Estatística

SRD - Superintendência de Regulação dos Serviços de Distribuição

CONTENT

Chapter 1

INTRODUCTION

1.1 Context

Power quality is a topic of high importance due to the high demand for electricity in modern society and the impact it has on the production chain. Providing reliable electricity at reasonable costs avoids economic losses, increases productivity, and enables industries and agriculture to become competitive (Oseni & Pollitt, 2013; Horváth, 2014). Since power quality is an indispensable requirement for the proper functioning of high-tech equipment in industries, mitigating interruptions can prevent large economic losses to industrial consumers and improve the functioning of electronic devices, since modern equipment are much more sensitive to energy variations (Muhamad *et al*., 2017; Khalid & Dwivedi, 2011).

The term "power quality" has different concepts in electrical engineering. Some references prefer to use terms like "power supply quality", others like "voltage quality". But, despite the different concepts generated about the term, the fact is that electric power systems must generate energy and deliver it continuously with an acceptable voltage to the final consumer (Bollen, 2000; Bollen 2003).

Thus, conducting research related to the generation, transmission, and distribution of electricity, as well as implementing innovations in the electricity sector, are important actions to face challenges. These challenges include maintaining quality indexes, guaranteeing the energy supply to essential activities, and maintaining industrial productivity.

1.2 Justification

This work is justifiable by its contribution for the literature related to the use of government data of Brazilian electricity consumption with focus on predicting electricity interruptions, especially for consumer units of medium voltage. We can also consider the research relevant due to the proof of concept that previously identifies the consumer units that will have interruptions, making it possible to create action plans that can help to avoid losses, reduce the severity of damages caused by the lack of energy, improve the planning of maintenance routines, improve the production chain as a whole, avoid fines for concessionaires,

increase circuit safety, increase the value perceived by the customer, in addition to providing data to support the reduction of customer dissatisfaction.

We can also find justifications in document CP 038/19 by Technical Note No. 046/2019-SRD in which according to ABRADEE (Brazilian Association of Electric Energy Distributors), in its contribution to the regulation of energy supply, it clarifies that the quality indices of Brazilian energy have been improving every year. However, non-compliance directly affects the concessionaire, as it is necessary to compensate customers for the transgression of individual continuity indicators, as well as directly affecting billing, as in some cases the interruption can mean energy not supplied and leads to lawsuits by the harmed consumers.

More work that justifies the research was developed by Barros (2020), who obtained access to a database available by ANEEL to create a project for a tool for managing OPEX and CAPEX actions with an impact on the reduction of continuity and compensation indicators (ANEEL Research and Development Project no ¯ 116/2018) interruptions caused by broken networks or in poor conditions represented a considerable portion (56%) of the DIC indicator (Individual Interruption Duration per Consumer Unit) in area networks, in consumers medium voltage according to the same article. This reveals that investments in infrastructure are necessary and identifying consumer units can support the prioritization of investment in the replacement of assets. Since investments are necessary, finding these customers in advance also helps to promote tariff balance as it will provide more data to work out the cost-benefit of these investments.

1.3 Objective

1.3.1 General Objective

The general objective of this research is to analyze data from Brazilian electricity distribution companies, to find patterns in customer consumption profiles, using Kohonen's self-organizing map algorithm (SOM), to extract features using decision trees and to predict interruptions of medium voltage using fully connected neural networks.

1.3.2 Specific Objectives

To achieve the objective established, some specific objectives were defined:

- Achieving the best topologies and efficient training for selected models;
- ensure precision for predictions;

- create a concept proof so that it can be developed by energy utilities;
- expand the academic view of the challenges of the power quality sector;
- reveal data related to problems of electric utilities/government concerning the profile of consumer units.

1.4 Limitations, Conventions, and Scope

In order to achieve the objectives proposed in this research, it was necessary to limit its scope and establish conventions. The main limitations are mentioned below:

- The classification of municipalities was performed based on the size of consumer units of the municipalities and the sectors present in the economy.
- Other types of training could have been used, compared and brought more gains, for example separating cases into old and new consumer units.
- It was not tested whether there is a deterministic component in the data.

Regarding the conventions adopted, it can be mentioned that:

- Assuming the 9.5 interruption index for FIC is reasonable for the Brazilian electricity sector.
- The technique/method used did not matter once the resolution of the objectives provides a contribution to the electricity sector. In this way, was agreed that SOM, Decision Trees and Fully Connected Neural Networks techniques are the most suitable for solving the problem.
- It is agreed that in this work the DEC/FEC indicators are associated with energy quality, and that they are the most relevant from the utility's point of view.
- It is agreed that the initial parameters used for the construction of neural networks are adequate for the first version of the algorithms, which would later be revealed by applying the tunning on the algorithm of fully connected neural networks.
- It is agreed that the CNAE is suitable for analyzing the descriptions of economic activities.
- The training methods imbalanced, balanced by weight class and oversampling are the most appropriate techniques to deal with the evaluation of metrics in unbalanced sets.
- Convention that 80/20 is an adequate distribution ratio for training.

- It is agreed that the tuning performed by Keras Tunner provides adequate results for neural network optimization.
- Precision, recall, and Area Under the Curve (AUC) are the best metrics for analyzing training with unbalanced data.

1.5 Chapter Organization

This dissertation is divided into 6 chapters, as follows:

Chapter 1 establishes the subject of the research, presents the objectives intended to be achieved with the work, which involves predicting continuity indices of electrical power interruptions of medium voltage consumers through fully connected neural networks. It also presents the challenges, which include ensuring the accuracy of predictions and how to make the evaluation of results useful to the management of electric companies and the regulation of the sector. Therefore, it guides on how this written work was structured to facilitate the whole understanding.

Chapter 2 explains the importance of power quality for medium voltage consumers, in addition to describing how Artificial Intelligence models can help power utilities in energy quality, the works present in the literature related to the research topic, emphasize the difference with the work developed. There is also a discussion of the results obtained by the main references that guided the development of this study.

Chapter 3 specifies the algorithms used: SOM, CHAID, CART, and FNNs. It addresses its mathematical formulations, describing the functioning, advantages, and particular aspects of each machine learning model. Furthermore, the chapter lists the software tools that allowed the development of artificial intelligence models.

Chapter 4 presents the methodology used to develop the algorithms for predicting continuity indices, describing the pre-processing steps, training and testing sampling techniques, and the extraction of attributes. In addition, the statistical parameters that will be used to evaluate the results described in chapter 5 are described.

Chapter 5 explains in detail all the results achieved, dealing with the efficiency of each proposed prediction model. The data are presented in a structured way in tables and graphs, allowing the comparison from different perspectives of the particularities of each algorithm.

Chapter 6 presents the conclusion, the contributions made by this research, and proposals for possible future related works.

The appendices contain the codes of collection, treatment, and the algorithms developed for each model implemented in Python.

Chapter 2

THEORETICAL FOUNDATION

2.1 Introduction

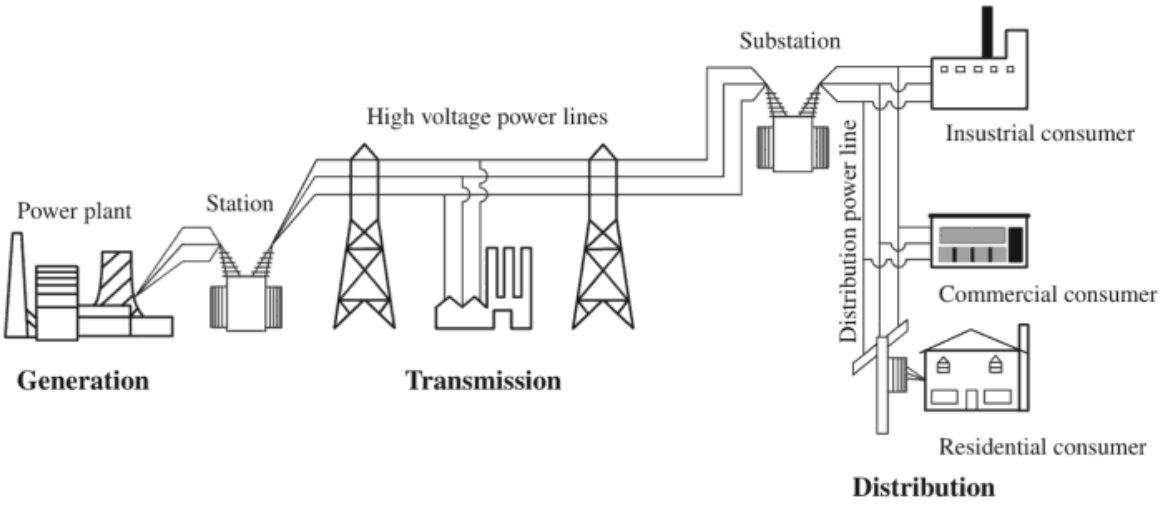
The energy supply process is a complex process, which involves different elements from delivery to the final consumer. Considering that this work is limited to solving the power quality problems of power interruptions of medium voltage customers, this chapter intends to deal with the importance of power quality for these customers, and how Artificial Intelligence models can help utility companies electrical energy in the delivery of the quality of the electrical energy supplied.

2.2 Importance of Power Quality for Medium Voltage Customers

The quality of electrical energy is directly associated with the daily life of medium voltage consumers. Several factors can impair the quality of medium voltage energy supply. Since consumer units have different needs, existing problems in distribution can cause different types of impacts on these consumers (Das *et al.,* 2018). Thus, this section will be started by defining what a medium voltage customer of an electrical company is, and in addition, will be defined what an electrical system is and its responsibilities.

An electrical power system is formed by different elements, capable of bringing energy to the final consumer. These elements are plants and stations, capable of generating electric energy, high voltage lines, capable of transmitting electric energy, substations, capable of distributing electric energy, and distribution networks, responsible for delivering electric energy to the final consumer, which is technically called a consumer unit. In turn, distribution networks can be distinguished by their consumption capacity, which is measured in volts, so they are separated by voltages: medium voltage networks. This generally provides energy to industrial and commercial consumers, and low voltage networks, which provide energy mostly to residential consumers. Figure 01 below illustrates the electrical power system and its elements:

Figure 1 – Electric power system.



Source: Blume, 2002.

While in industrial and commercial consumers, problems in distribution can affect productivity, burn equipment and even cause explosions. In residential consumers, the lack of energy can cause problems in domestic activities and cause damage to domestic equipment. Thus, for both consumers, power quality is extremely important. This consumer requirement reveals a primary responsibility of transmission networks: to deliver electrical energy through the transmission network in a continuous manner, and with an acceptable voltage to the final consumer. This responsibility is directly linked to two other factors: modernization and better administration of distribution networks. These factors ensure the high quality of energy for end consumers (Dashtdar *et al.,* 2018).

Sannino, Svensson, and Larsson (2003) divide power quality disturbances into two classes: the first, interruptions and voltage changes are caused by faults mainly in the power system. In these cases, the system causes total or partial interruption of energy delivery. In the second class, the phenomena are caused by the low quality of the current. In these cases, the current often arrives unbalanced at the end consumer, but without interruptions, thus fluctuating the quality delivered (Khalid *et al.,* 2011). In both cases, despite the difference in the location of the failure and the severity of the damage, both can cause harm to consumers.

For Petleshkov and Lozanov (2019), the main causes of interruption in energy supply are successful auto-reclosing, weather conditions (falling logs and trees, storms, strong winds, snow), and defects in electricity cables (Soares *et al*., 2014). Inherent in power distribution, causes of reduced power quality are commonly recorded in the exact detection of the problem through sophisticated testing equipment. Khalid and Dwivedi (2011), report some events that can be indicators of power quality problems: pieces of equipment that deregulate at the same

time of day; short circuits that occur without overloading; equipment failures during storms; stoppages in automated systems for no apparent reason; electronic systems failures or frequent operations failures; systems that work in some places and not in others (Levi, 2005).

Since different approaches have been proposed to define, classify and demonstrate the importance of the quality of energy supply, the next section aims to show how artificial intelligence models can help utilities to improve the quality of energy supplied.

2.3 How Artificial Intelligence models can help energy utilities in energy quality

Artificial Intelligence (AI) is a branch of computer science that proposes to develop devices that simulate the human ability to reason, perceive, make decisions and solve problems. AI has drawn a lot of attention due to the effective approximation it has had with human learning and reasoning. Within this field, machine learning has gained the attention of the scientific community, because of the possibility of combining with other science areas, through the characterization of databases that have been used mainly for statistical purposes, especially in the field of electrical engineering (Saninno *et al*., 2003).

To achieve the goal of electrical modernization, the electrical network must be also worked intelligently, especially using AI. One way to work intelligently is to use statistical models that provide behavior analysis, including predicting future behaviors. In this way electric companies can benefit from these systems by making predictions of future events, to avoid interruption problems in their supply and guarantee high-quality energy. These tools are key elements, constantly analyzing the need for component updates and ensuring the integrity of the electrical matrix (Hammond, 1997; Bravo-Rodriguez, 2020).

From proactive maintenance plans, electric companies took advantage of advantages over reactive plans such as: avoiding fire situations, cascading failures, and emergency costs. However, it is not straightforward to determine where limitations are located to ensure more effective repair of vulnerable components. In this sense, several studies on the implementation of smart grids have been developed and specialized. They act, for example, in the classification of electrical distribution failures. However, the advances have been particular, although characteristics between cities and electric companies are common, there are geographic particularities such as climate, network topologies, and even standards and maintenance policies regulated differently (Rudin *et al*., 2012). However, the location of failure points is difficult to identify. Recently, neural networks have gained attention in several energy

applications and fault analysis is one of the most important. There are reports in the literature of the use of artificial neural networks, to identify fault locations using different implementations of neural networks.

Thus, the use of artificial intelligence combined with data-based strategies can prevent unnecessary investments from being made and direct investments in the right way, which can significantly improve the assertiveness of established indices. Furthermore, improving power quality can increase customer satisfaction with the service, as not all customers have been rewarded for the interruption of electricity supply (Ferreira *et al.,* 2020; Ramos & Melo, 2010).

Regarding the literature related to this section, many works make use of neural networks to apply artificial intelligence to face the challenges of electric companies. The neural networks have been successfully applied to predict interruptions. Examples are the use of neural networks to identify network fault section, fault location, and reliability worth analysis of distribution systems (Dashtdar *et al*., 2018; Heidari *et al*., 2017). Both papers do not present a categorical data analysis, only numeric variables are used for prediction, which is important data for strategic planning and better translating operational problems. The papers of Farhoumandi *et al*. (2021), Kumbhar *et al*. (2021), and Volosciuc & Dragosin (2015) discuss how neural networks can be used to evaluate interruptions. In this paper, an analysis of the interruptions presented about the network infrastructure is shown through Kohonen's self-organizing map algorithm (SOM), CHAID, and CART. Furthermore, the fully connected networks are used to exploit attributes commonly present in energy utility databases to predict interruptions indexes.

2.4 Final Considerations

In this chapter, the main reasons for using artificial intelligence methods to support power quality problems in medium voltage consumers were discussed. In section 2.2, the importance of electricity quality for medium voltage consumers was discussed, focusing on the differentiation of these consumers by voltage, their respective problems and impacts. On the other hand, section 2.3 presented the main artificial intelligence techniques and algorithms currently used to help electric utilities face the challenges. This highlights the neural network techniques that are the focus of this work.

In the next chapter, the theoretical aspects of each algorithm used in the research will be addressed, including their mathematical and statistical formulations and the explanation of the software and computational language used.

Chapter 3

LITERATURE REVIEW

3.1 Kohonen's Self-Organizing Map

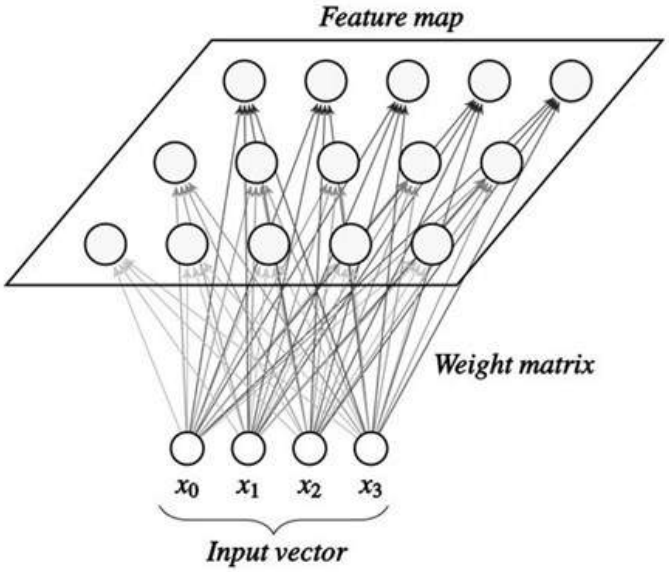
There are several techniques for grouping data, such as K-Means, DBSCAN, Hierarchical Cluster, and Grid-based grouping. For such an application, neural networks also prove to be powerful for recognizing patterns and relationships between variables (Mingoti & ima, 2006).

Kohonen's self-organizing maps are basically formed by two layers of neurons: an input layer and a unit layer, called U. The neurons of the U layer are arranged in an architecture that offers a notion of the neighborhood between the neurons (Ultsch, 1993). Considering architecture of two dimensions, the neurons are distributed in a two-dimensional grid where all input neurons are connected to all U layer neurons (Kohonen, 1990).

In this way, if an n-dimensional input of a vector of real numbers is presented to the network, all neurons in the U layer compare their weight vectors and the neuron with greater similarity receives the correspondence (Ghaseminezhad & Karami, 2011). In this way, all neurons in a given neighborhood learn about the input vector and update their weights to become more similar to the given input vector (Strecker & Uden, 2002). This process is compared to a competition among the neurons, in which the most similar neuron, the winner, and its neighborhood are adjusted to become closer to the input pattern.

The inputs are presented in a random sequence and through the repetition of this process over several periods, the range of the neighborhood is changed so that a neuron initially has many other neurons in its neighborhood, but at the end of this stage it has few or no neighbors (Kohonen, 1990). After some training periods, the neurons can represent relations in the input samples in one or two-dimensional space. Therefore, the SOM can reduce dimensionality, allowing an easier way to analyze data. It is also possible to delimitate clusters observing the distance between the neurons in the SOM, or data attributes depending on the application. Figure 2 represents the feature map in a two-dimensional grid.

Figure 2 – Dimensional grid representation.



Source: Blume (2002).

3.2 Decision Trees

Decision Trees is another special method for recognizing patterns and relationships between variables. Decision trees are supervised learning methods, used for classification and regression (Somvanshi *et al*. 2016). Based on divide-and-conquer strategies, decision trees mimic the functioning of the human brain and are capable of solving complex problems. Given a target variable, decision trees can infer rules from features. Its implementations exist in the most different forms, differentiating them by the way of applying classification rules, architecture, and complexity (Gregoriades *et al*., 2021). Strongly used for recommendation algorithms, decision trees are one of the most used classification techniques for their advantages, which are: could be an easier interpretation, which means that it is an explanatory model and in computational terms a very efficient one. However, from different perspectives, decision trees have some disadvantages such as compromised processing time when there are many variables and the fact of working with preferences of hypotheses over others because the bias is inductive (Rakhra, 2021).

For understanding the operation of decision trees, DTs divide the data sets into subsets where at each node they are established by the rules defined in the branches, these divisions use different metrics such as information gain or Gini impurity indices (Podgorelec *et al*., 2013). Unlike black-box methods, DTs provide visibility into the rules and thus explains the rules used

for prediction. Through representation and rules, it is possible to identify which predictor variables have more strength to better explain the model. In this way, supporting not only data scientists but also decision-makers (Breiman, 2017).

Other concepts involved are: Impurity functions, which are the criteria by which the model will split the data and merge potential tree nodes. The importance of variables is that the scale of the explanatory power of predictor variables. Decision trees use the importance of variables to reduce the relative error and include the variables that will best explain the model. That way all variables with zero importance are excluded. Stopping criteria are defined for when none of the nodes can be subdivided, or when a certain depth is reached. The predictive strength, which is used to avoid overfitting, which is the excessive fit of the data, is calculated by the contribution of variance for each leaf node (Rokach, & Maimon, 2005).

As mentioned in the first paragraph, there are different implementations of decision trees, which can be cited mainly such as ID3, C4.5, J4.8, C5.0, CART, Random Forest. Sections 3.2.1 and 3.2.2 dealt specifically with the CHAID and CART algorithms, which were used by this work to understand the problem.

3.2.1 CART

Classification and Regression Trees or CART is a type of Decision Tree algorithm used to solve predictive classification problems. This is one of the most classic DT algorithms, it is based on the GINI impurity measure as a separation criterion and creates a binary representation of the created rules. We can define GINI in a simplified way as a measure by which the probability of misclassification of a new instance of a random variable is calculated (Singh & Gupta, 2014).

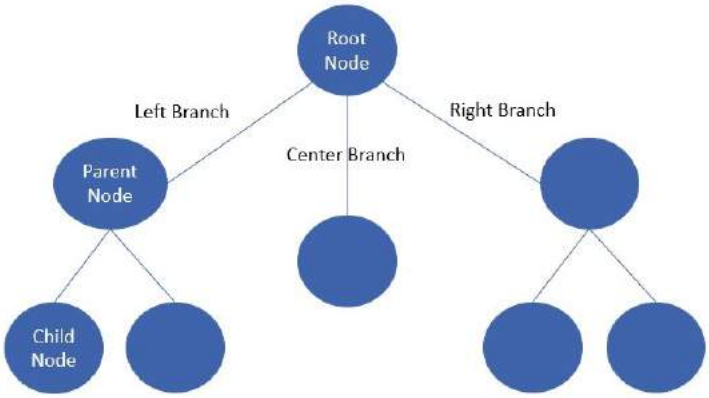
The CART algorithm works as follows: First, the best division is found for each variable, for each division the criterion is to maximize the separation measure. The result is a set that contains the best divisions. Once the best node split has been found. It is necessary to find between the divisions of the first step, the one that maximizes the separation criterion. Finally, the nodes are split using the best split made in the previous step, so the steps are repeated iteratively until the stopping criterion is satisfied (Rutkowski *et al*., 2014).

3.2.2 CHAID

CHAID, or Chi-squared Automatic Interaction Detection, is another decision tree variation that uses chi-squared statistics to define branch breaks at nodes. One of the first steps

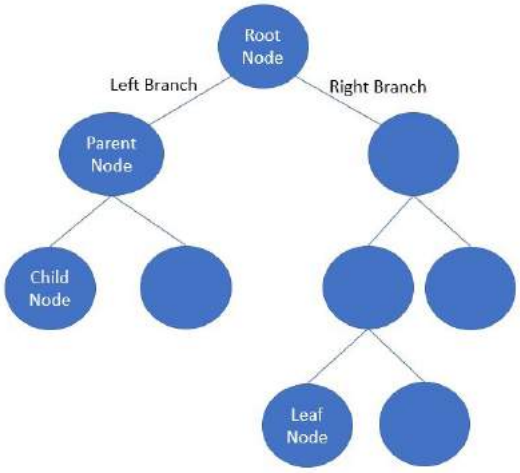
of this algorithm is to verify if the relationships between the variables are statistically significant through the chi-square test of independence. Subsequently, if an entry has more than two categories, they will be compared and if they do not show significant differences they will be grouped in the same node. This is done recursively, and the process ends when all categories have been tested. For nominal variables, any category can be joined into a single node. For ordinal variables, only continuous categories can be grouped into a single node. The model has the disadvantage of fully examining all clustering possibilities which is costly in terms of processing time. However, it is a very efficient model and because it uses the chi-square method, it provides a good extraction of variables (Milanović & Stamenković, 2016; Baizyldayeva *et al*., 2013; Hammann & Drewe, 2012; Safara *et al*., 2020). Figures 3 and 4 following show the difference between the architecture types and their branch types in the illustrations.

Figure 3 – Representation of CHAID and its multiply nodes.



Source: The author.

Figure 4 – Representation of CART and its binary nodes.



Source: The author.

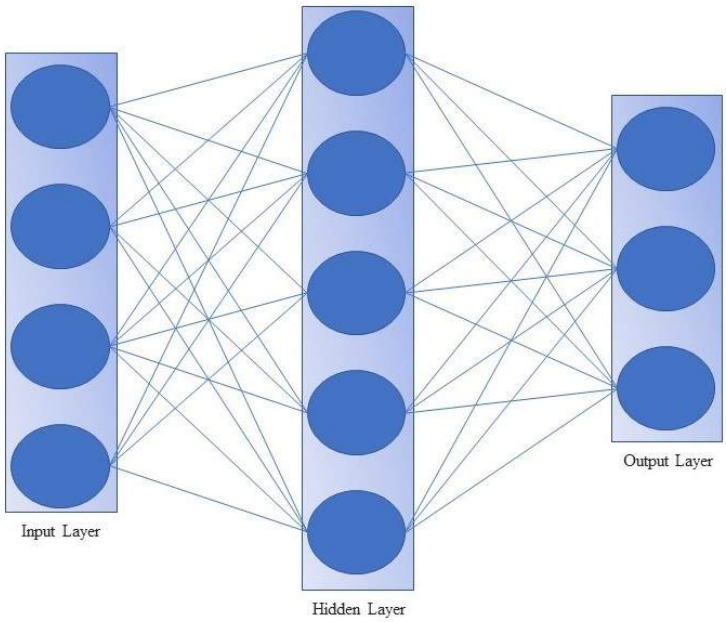
3.3 Fully Connected Neural Networks

Within the universe of artificial intelligence, neural networks research, along with its advances, has gained prominence in several fields and has brought significant contributions. Neural Networks can be understood as a method of optimizing processing time in several computational processes, as well as improving results. It is a hierarchical method and abstract analysis of layers, but even so, it can be applied in several situations of real-life as image processing, medicine, biometrics, among others (Vargas *et al*., 2017).

Artificial neural networks stand out for their ability to address complex nonlinear relationships and working with multiple layers. These characteristics allow not needing any additional training data in some cases, being able to be applied in different types of forecasts and achieving good accuracy. Although there are disadvantages in using them, such as being a black-box method, and being difficult to interpret the results, they are quite useful in electrical engineering. Its application can range from a complete power supply process, from the detection of network failures to the analysis of the operations of power systems (Kumbhar *et al*., 2021). A relevant work and correlated with this work in the literature, with the use of artificial neural networks to identify fault sites, is that we can mention the work of Adewole, Tzoneva, and Behardien (2016), who proposed a hybrid method that uses indices in pre-processing stages. The results showed an excellent performance in detecting different types of faults and origins (Adewole *et al*., 2016).

Analogously other methods, have a wide variety of types of neural networks, such as Recurrent Neural Networks (RNNs), applied in speech recognition and natural language translation; Convolutional Neural Networks (CNNs), used for image recognition; Feedforward Perceptron Multi-Layer Neural Networks (FFNNs) used for classification, and the Fully Connected Neural networks (FNNs), used for detection and prediction, which this work seeks to deepen and implement. The main difference between Fully Connected layers and other neural networks are those layers where all the inputs from one layer are connected to every activation unit of the next layer. An example of Fully Connected Neural Network can be visualized in Figure 5 (White, 1989; Bhamare & Suryawanshi, 2018; Faris *et al*., 2016).

Figure 5 – Representation of fully connected neural network.



Source: The author.

Fully connected neural networks were chosen because they are a model capable of reaching high precisions and dealing better with unbalanced data than other techniques such as logistic regression, random forest, ag bosting, etc., and also better than other neural networks. A deeper analysis of the fully connected neural networks parameters is presented in the results section, which will illustrate how to use fully connected neural networks to predict interruptions.

3.4 Tools Utilized

3.4.1 *Python*

The algorithm was implemented in the Python language. Among the benefits that justify choosing such a language, there is the availability of a wide range of libraries, ease of testing, and the parallel processing that some of its libraries offer. Python was created to organize programming patterns, to be able to manipulate objects, and can be very useful for working with data because it has standard and comprehensive libraries (Yarlagadda, 2018).

3.4.2 Pandas

The Python Pandas library was necessary to perform the data crossing and manipulations of the time series of the data of the concessionaires used in the research. Integrated per Python, this library allows an excellent integration of data preparation

functionalities for artificial intelligence, mainly due to the simple syntax that converts operations on dataframes into SQL queries. Despite being a less robust tool, the ease of implementation can be very useful at times (Hagedorn *et al*., 2021).

3.4.3 Pyspark

To support the step of data processing, considering that the database has a high number of records, parallel processing techniques were used, and proper data structure to reach the best precision when modeling the problem using neural networks to specify the libraries for parallel processing, the library used was Pyspark. The major benefit of this library is working with large volumes of data by native libraries with large community support and also providing great availability of data handling features (Stančin & Jović, 2019).

3.4.4 IBM SPSS

For initial analysis and feature extraction, was applied IBM SPSS. It was chosen because it is statistical software, that allows performing descriptive analysis of variables through a visual interface, using advanced statistical procedures it is possible to have high precision to make quality decisions. Therefore, the use of this statistical software is useful to interpret the outputs, facilitating the appropriate formulations and interesting research questions (Fávero & Belfiore, 2017).

3.4.4 Tensor Flow/Keras

For training and evaluating the results, the libraries Tensor Flow and Keras were chosen. They are a high-level neural networks API, integrated by the Python language, which allows very fluid experimentation. Supported by strong documentation, it is possible to assemble quite functional neural network algorithms, whether from small or large datasets (Arnold, 2017).

3.5 Final Considerations

This chapter was intended to address the theoretical and mathematical aspects of the algorithms used in this research. In addition, it was about the usage of computational tools for the elaboration of the proposed models.

In section 3.1, the theoretical aspects of the self-organizing maps algorithm were discussed, detailing its mathematical aspects and how it works.

Section 3.2 was intended to detail the operation of decision tree algorithms, especially introducing the concepts of CHAID and CART decision tree derivations used in the work.

In section 3.3, the theoretical concepts of fully connected neural networks were discussed, informing their mathematical aspects, advantages, and disadvantages.

Section 3.4 addressed the computational tools used in this work, discussing the python language and the libraries used to implement the algorithms mentioned in the chapter.

In the next chapter, the methodology used in the research will be discussed, presenting the database used, processing steps, and methods for evaluating the algorithms.

Chapter 4

MATERIALS AND METHODS

From initial descriptive analyses, a hypothesis was formulated regarding possible problems related to energy quality faced by customers. To confirm this hypothesis, a cluster analysis was carried out, to understand the profile of these customers and the main existing challenges. The confirmation of the hypothesis by the analysis enabled the development of an artificial intelligence model that could find in advance the customers who will present continuity indices above those stipulated by Aneel.

The other steps in the implementation of the solution were: previous data analysis with descriptive statistics, data normalization or standardization, removal of outliers, application and adaptation of the SOM algorithm from Vettigli (2021), delimitation of clusters to analyze, and evaluation of the centroids and the data samples into the clusters. The steps followed for the creation of the solution were: data preparation, extract features using decision trees (CHAID, based on Chi-Square, and CART, based on feature importance), exploratory data analysis, data normalization or standardization, application, and adaptation of the neural network algorithm (fully connected with oversampling and balanced per class).

Additionally, to enable the objectives to be achieved satisfactorily, adjustments and analysis of the parameters were made, allowing the refinement of the results. Data processing was a challenging step because the database has a high number of records, which required the use of parallel processing techniques, as well as an adequate matrix data structure to compute centroids efficiently. To support the step of data processing, considering that the database has a high number of records, parallel processing techniques were used, and proper data structure to reach the best precision when modeling the problem using neural networks. For initial analysis and feature extraction was applied IBM SPSS. To build the implementation the Python language was chosen. This decision was made because such language offers a wide range of libraries, ease of testing, and can process data in parallel.

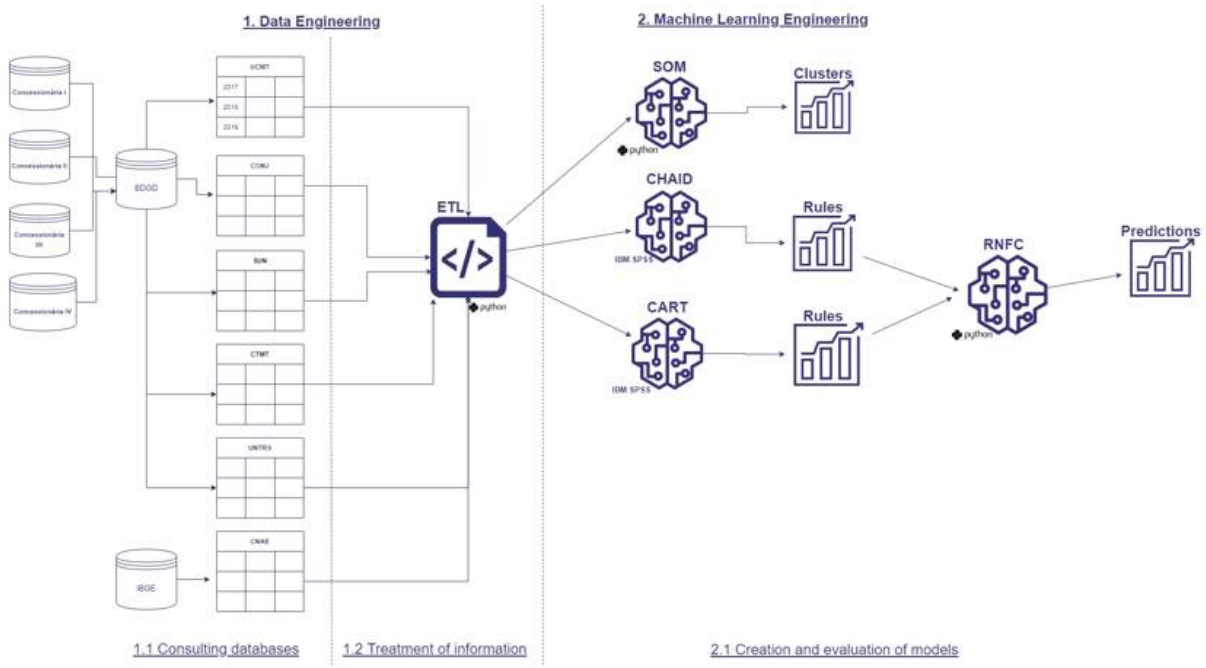
4.1 Data, Architecture of Concept Proof, and Implementation Strategies

During the implementation of the algorithm, one of the main challenges faced was the selection of variables and the proper processing of information. A prototype was created using database tables of medium voltage units to achieve the objectives. The data did not exist ready to be consumed, the data was scattered in different tables and some essential information was not available, requiring a web scraping on the ANEEL website to obtain the information. In this way, information query scripts, cleaning, treatment, data crossing, standardization of variable names, construction of variables, data standardization, and exploratory analysis were necessary to create the prototype that could provide sufficient data for the research objectives.

The architecture of the concept proof is shown on Figure 6. The data sent by concessionaries for ANEEL are stored on the database, named BDGD, which is a data representation of the Brazilian electrical system. Despite having some registration problems, it is the source used by ANEEL for the application of fines and price stipulation (ANEEL, 2021). Despite its vulnerabilities, BDGD is one of the main sources of energy supply analysis, even though it does not have all the relevant information involved in energy supply (Tronchoni *et al.*, 2010). Due to regulatory resolutions, Brazilian concessionaires are required to submit their information for audit (ANEEL, 2021). Access to the database was provided by the Research and Development consortium, however it can be requested by anyone for analysis by requesting the regulatory agency.

These data were available in different tables separated by Medium Voltage (UCMT) and Low Voltage (UCBT) and on tables that describes the description of circuits (CTMT), substations (SUN), substation transformer unit (UNTRS), conjunto (CONJ), that could be interpreted as a set of consumer units, and other common elements of network distribution. The data was enriched with CNAE-Subclasses (National Classification of Economic Activities, version 2.3) and join through by python pipelines, the series of data available is of the years of 2017, 2018 and 2019. The data provided by BDGD, have all existing distributors in Brazil. Thus, to create the concept proof, two extractions were used, the first covering a complete state of the Brazilian Southeast region, having 14000 customers and 42 variables. Used to understand the context through the Self Organizing Maps algorithm, and a second clipping that also involves a complete state of the western region with 910 medium voltage customers and 130 variables.

Figure 6 – Architecture of Concept Proof.



Source: The author.

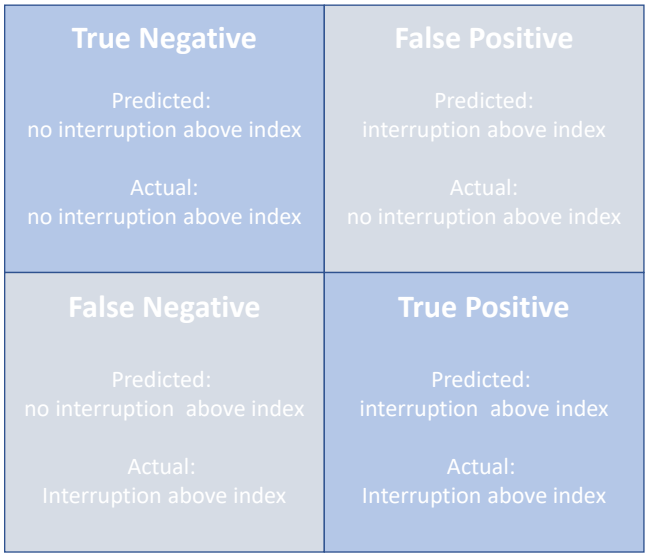
The variable target was created based on the interruption indicators, DIC and FIC, above the established by the National Electric Energy Agency of Brazil (ANEEL) for the next year. The problem may be classified as mild imbalanced, whose proportion of minority class cases positives (interruption indexes over than limits) is 23.41%.

Many implementations of neural networks were tested, as well as own authorship as adapted from available implementations. Fully connected neural networks were chosen because they are a model capable of reaching high precisions and dealing better with unbalanced data. The first implementation was built as a baseline, just using a single layer and the final implementation has two layers fully connected. To understand the best distribution for the dataset sample, both implementations were trained with imbalanced class, oversampling, and class weights. The ratio split of training and testing sets was 80/20.

Moreover, parameters such as number of neurons, number of epochs, and learning rate, were carefully adjusted based on the results of baseline implementation. The optimal parameters to tune the neural network was found using Keras Tuner Library.

To evaluate the results, a confusion matrix was obtained considering the number of correct and incorrect predictions, according to the following definitions: False Positive (FP) as the prediction of an interruption above the index when actually there was not an interruption above the reference; False Negative (FN), prediction of no interruption above the index when actually there was an interruption above the reference; True Positive (TP), prediction of an interruption above the index when actually there was an interruption above the reference; True Negative (TN), prediction of no interruption above the index when actually, in fact, there was not an interruption above the reference. The confusion matrix used in this section of results is represented by Figure 7:

Figure 7 – Representation of confusion matrix.



Source: The author.

Considering that the target variable is an imbalanced class, the metrics used should be focused on reducing the false positives and false negatives, therefore, precision, recall, and Area Under the Curve (AUC) were obtained from the confusion matrix.

For understanding, if the training was working, the plots of the model's precision, loss, recall, and AUC on the training and validation set were analyzed to verify if the results were stabilizing and the model learning correctly.

4.5 Final Considerations

This chapter describes the methods adopted for analyzing data from medium voltage consumers and for formulating the artificial intelligence model, capable of predicting the interruption rates for each consumer unit.

In section 4.1, were presented the database, the techniques used to generate the classes, the form of splitting the sets of training and test, extraction of attributes, which will be used as inputs for the artificial intelligence models and the explanation of the methods used for evaluating realized forecasts.

In the next chapter, the results obtained and the comparison of the efficiency of the proposed algorithms will be shown.

Chapter 5

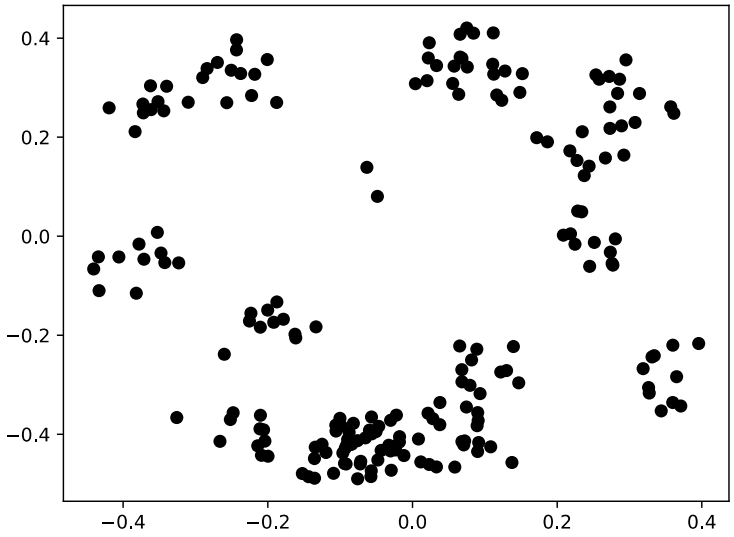
RESULTS

5.1 Clusters Formed by Self Organizing Maps

Among the possible analyzes considering the available databases, which have tables of various elements of the electrical network, the SOM was used to group data from medium voltage consumer units. As presented on the section 4.1, scenario 1 was used to evaluate the clusters. With 14000 samples, the variables utilized were the average consumption in one year (ENE_M), installed load (CAR_INST), and duration of individual interruption per consumer unit (DIC). These variables were chosen because the algorithm works only with numerical variables, which are the variables that generated the best results interpretations.

To understand the effect of each SOM parameter, synthetic data with two numerical attributes were used in such a way that they could be visually observed, as shown in Figure 8.

Figure 8 – Synthetic data.



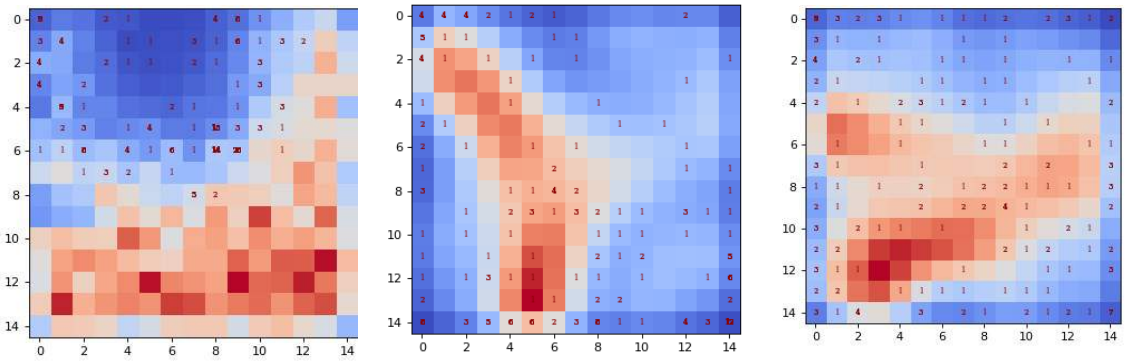
Source: The author.

It is expected that a tuned SOM will be able to separate such data since the sets can be visually separated. Among the parameters analyzed, the following stand out: the number of training seasons, the size of the neighborhood, and the dimensions of SOM.

As shown in Figure 9, where blue indicates clusters with a smaller average distance to their neighbors, and red indicates distant clusters, the number of epochs should not be too small,

as the data is condensed only in a part of SOM. On the other hand, it cannot be too big, as the data spread in such a way that the clusters mix visually, in addition to increasing the computational cost.

Figure 9 – Average distance map for different numbers of epochs.



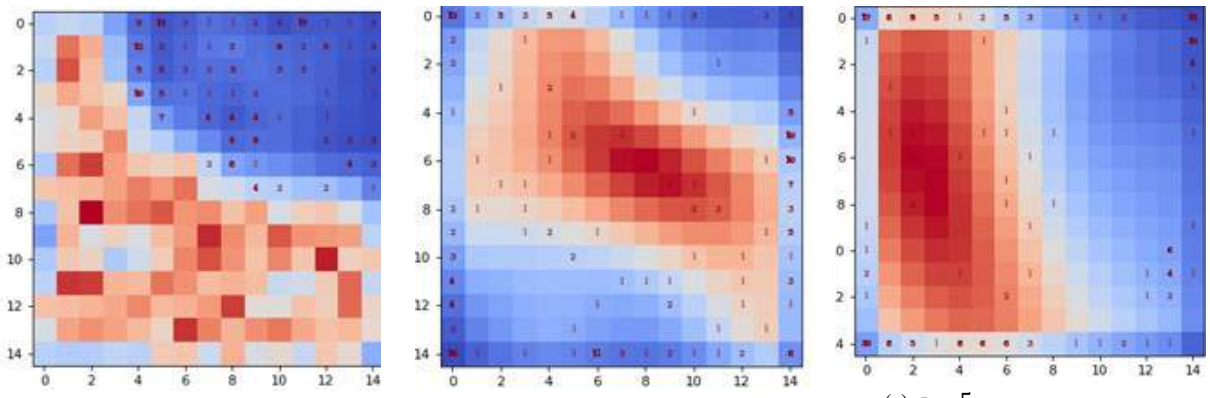
(a) 10 epochs (b) 150 epochs (c) 10000 epochs

Source: The author.

Considering a Gaussian neighborhood function, the effect of a neuron on the neighborhood can be adjusted with the standard deviation parameter σ. As shown in Figure 10, σ must not be too small, since this way the data is condensed only in one part of the SOM. On the other hand, it cannot be too big, as the data spread in such a way that the clusters are visually confused with a single large cluster.

Similarly, regarding the number of neurons, there is also an ideal range, as shown in Figure 11. Also, the increase in SOM means an increase in computational cost and execution time.

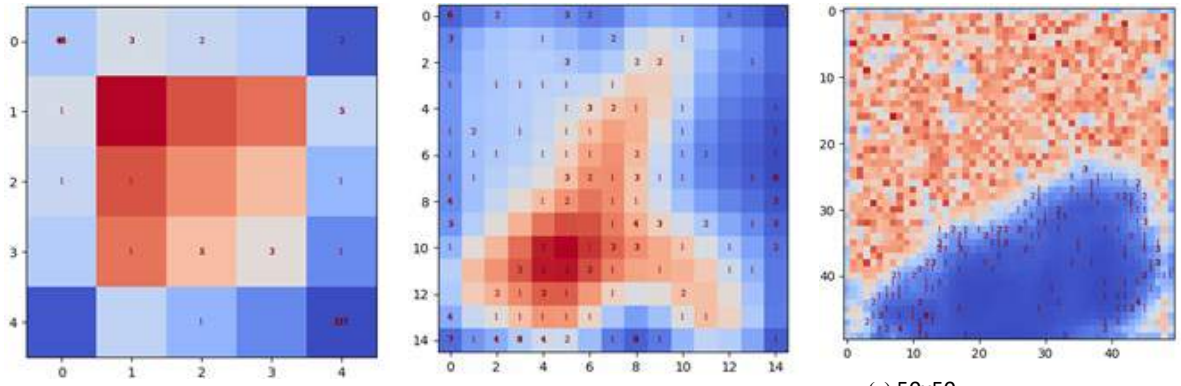
Figure 10 – Average distance map for different neighborhood sizes.



(a) $\sigma = 1$ (b) $\sigma = 3$ (c) $\sigma = 5$

Source: The author.

Figure 11 – Average distance map for different SOM dimensions.



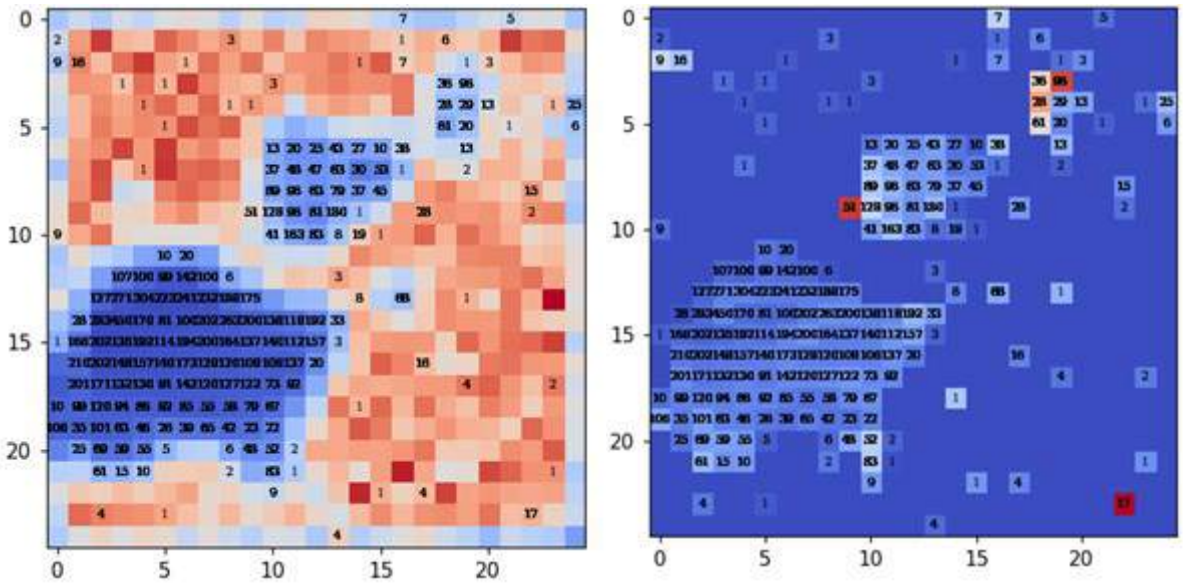
(a) 5x5 (b) 15x15 (c) 50x50

Source: The author.

5.2 Clusters based on energy consumption, installed load, and duration of the individual interruption

With around 14000 samples, and with the attributes of average consumption (ENE_M), installed load (CAR_INST) and DIC, the clusters that are shown in Figure 12 were obtained. Superclusters are seen in blue on the map of average distances, indicating neurons close together.

Figure 12 – (a) Average distance map. (b) The variance between samples classified in each neuron.

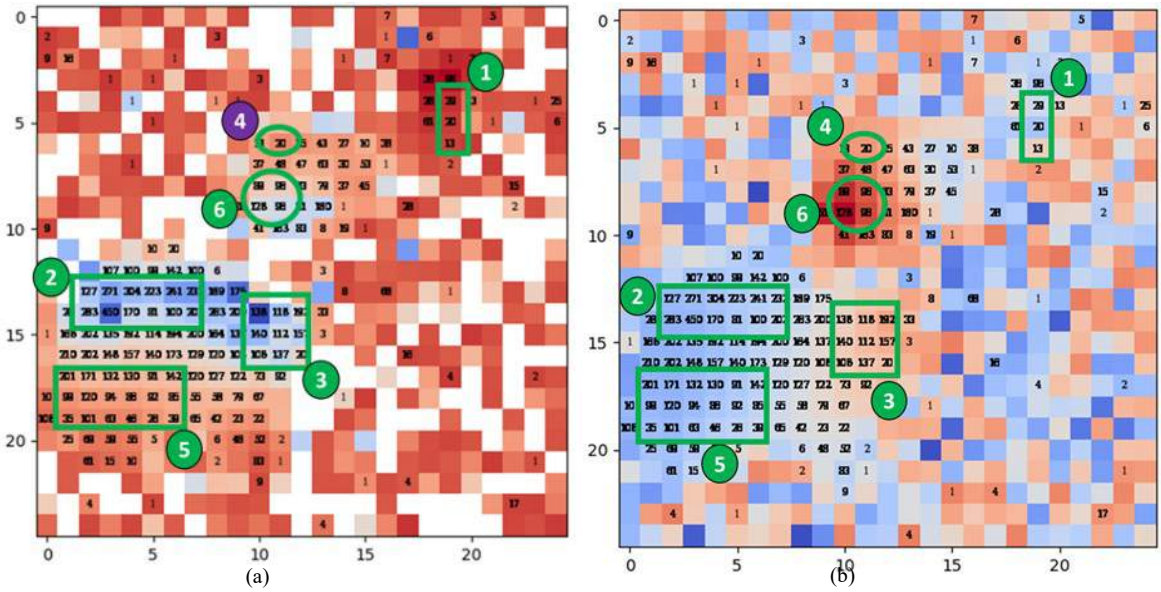


(a) (b)

Source: The author.

For a choice of superclusters, that is, groups of neurons for analysis, it is interesting to use maps of the components used by SOM to group the data, aiming to understand what led to the formation of each cluster and its characteristics. Figure 13 shows the maps of the average monthly consumption components ENE_M and DIC, which reveal another aspect: the formation of smaller clusters with common component values within the larger clusters.

Figure 13 – (a) ENE_M and (b) DIC components map.



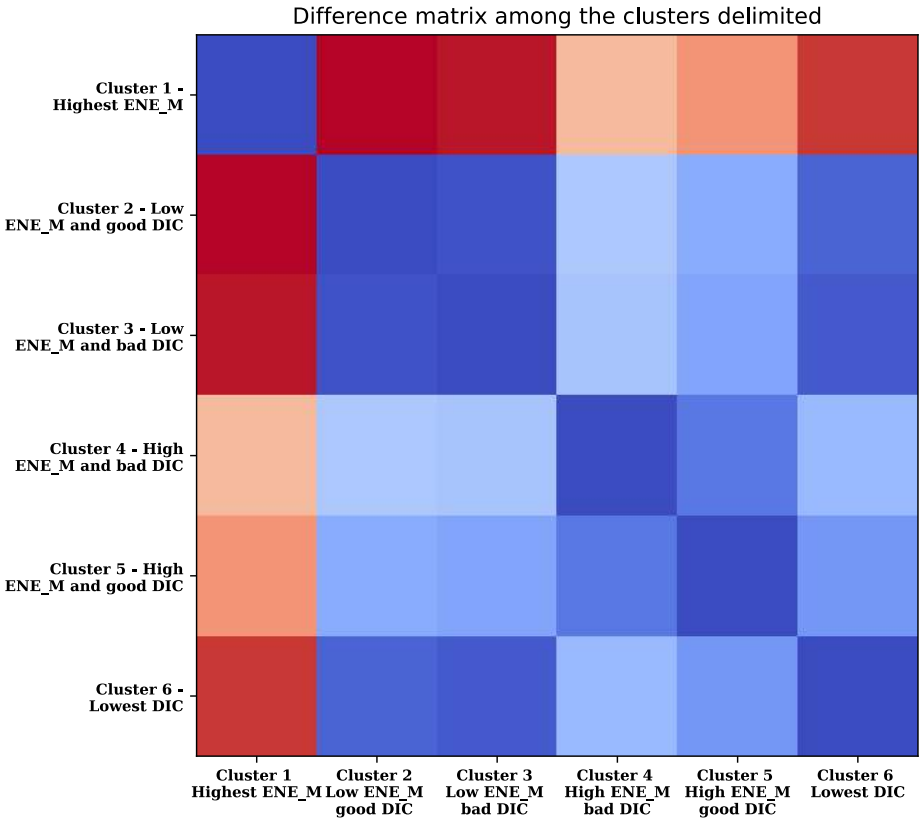
(a) (b)

Source: The author.

To verify the difference between the chosen superclusters, a matrix, shown in Figure 14, was used, each coordinate shows how different the two groups are. Group 1 is the most different, as the first row and the first column of the matrix have more red tones, indicating a greater difference for the other superclusters. Cluster 4 is less distant from cluster 1, as it has a lighter shade of red. This confirms what is seen in SOM's topology.

Comparing clusters 2 and 3, we can see from the map of the DIC component, Figure 13, that despite being close in the SOM topology, these two groups have different DICs: Group 2 has a good index, while group 3 has a higher index, that is worse, with more power outages.

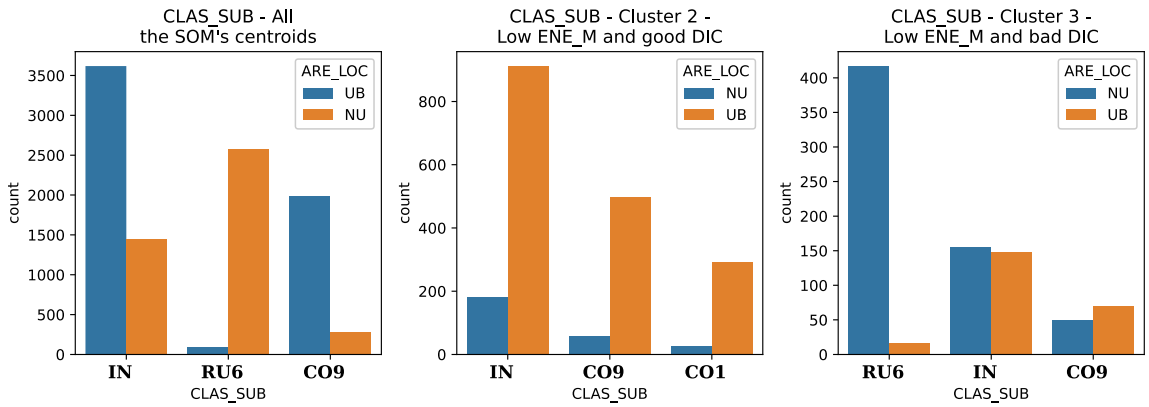
Figure 14 – Difference matrix among the delimited clusters.



Source: The author.

Regarding the classes of consumers present in each supercluster, group 2 presents a predominance of industrial and commercial units, while group 3 presents a predominance of rural and industrial units installed in non-urban areas, as shown in Figure 15. This figure also shows the most frequent categories in the entire database, to enable the understanding of the patterns of the groups analyzed in relation to the whole.

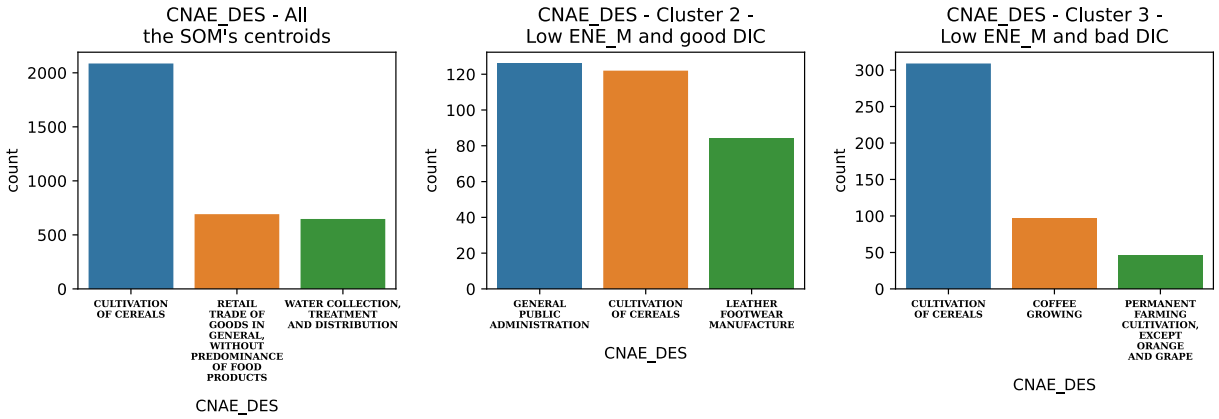
Figure 15 – Types of customers, urban UB and non-urban NU, with higher frequencies in the whole database and clusters 2 and 3.



Source: The author

Figure 16 shows the most frequent activities performed by these groups obtained using CNAE data from IBGE - Instituto Brasileiro de Geografia e Estatística.

Figure 16 – Most frequent activities in the whole database and superclusters 2 and 3.



Source: The author.

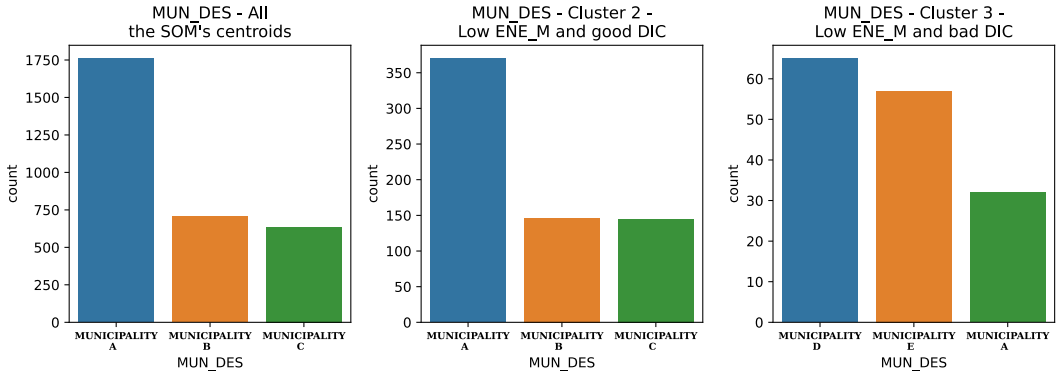
To analyze the clusters considering the cities with the largest number of consumer units in each cluster, a description of each municipality is useful. Such descriptions are presented below

- Municipality A is an extremely urbanized municipality with a mild climate, economic and political center.
- Municipality B is an extremely urbanized municipality.
- Municipality C is an urbanized municipality.
- Municipality D is a medium-sized and hot climate municipality with an economy based on mining and agriculture.
- Municipality E is a small rural municipality, with great economic importance.
- Municipality F is an urbanized municipality.
- Municipality G is a medium-sized municipality with a mild climate and an economy based on agriculture.
- Municipality H is an urbanized municipality.
- Municipality I is a small rural municipality.
- Municipality J is a medium-sized municipality with an economy based on mining, industry, and agriculture.
- Municipality K is a medium-sized municipality with a hot climate and an economy based on agriculture.
- Municipality L is a small rural municipality.

- Municipality M is a small rural municipality.
- Municipality N is a small rural municipality.
- Municipality O is a medium-sized municipality with a hot climate and an economy based on agriculture and mining.

As shown in Figure 17, cluster 2 follows the trend of the base with large urban municipalities, A, B, and C, being more frequent. Cluster 3, on the other hand, presents small towns more frequently, D and E, whose economy is based on agriculture or mining. The municipality E, the second most frequent in group 3, is a big exporter of citrus products. This leads us to a situation in need of improvement since agricultural activities need good infrastructure regarding electricity. For example, to avoid losses in production due to lack of refrigeration or irrigation.

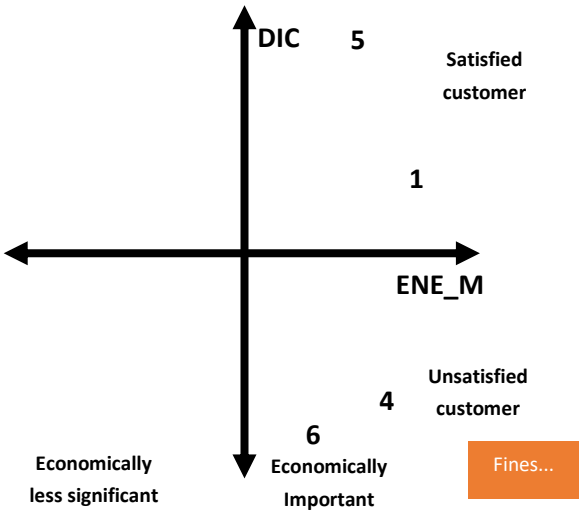
Figure 17 – Masked municipalities with higher frequencies in the database and clusters 2 and 3.



Source: The author.

To analyze subgroups 1, 4, 5, and 6, it is interesting to check the relationship between DIC and the average consumption (ENE_M) variables in these clusters. A high DIC means more time without power, which is a situation of operational difficulty for the concessionaire and poor service for the consumer unit. If the DIC indicator is below the levels established by law, the concessionaires pay fines. Higher consumption is related to the greater economic importance of the consumer unit for the concessionaire. These relationships are illustrated in Figure 18.

Figure 18 – Relationship between DIC e ENE_M.

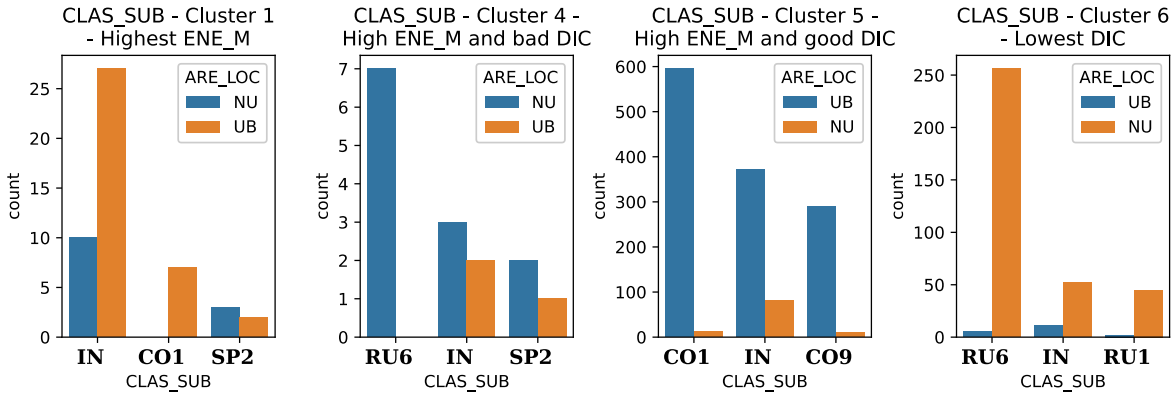


Source: The author.

Thus, groups 4 and 6 have a worse DIC indicator, which shows scope for improvement. On the other hand, 1 and 4 represent economically important consumers.

Figure 19 shows the most frequent groups of consumer units in each cluster. Again, the clusters with the worst indicators have a rural predominance, shown in the graphs on the right. Group 5 has a commercial predominance and presents the best DIC indicators.

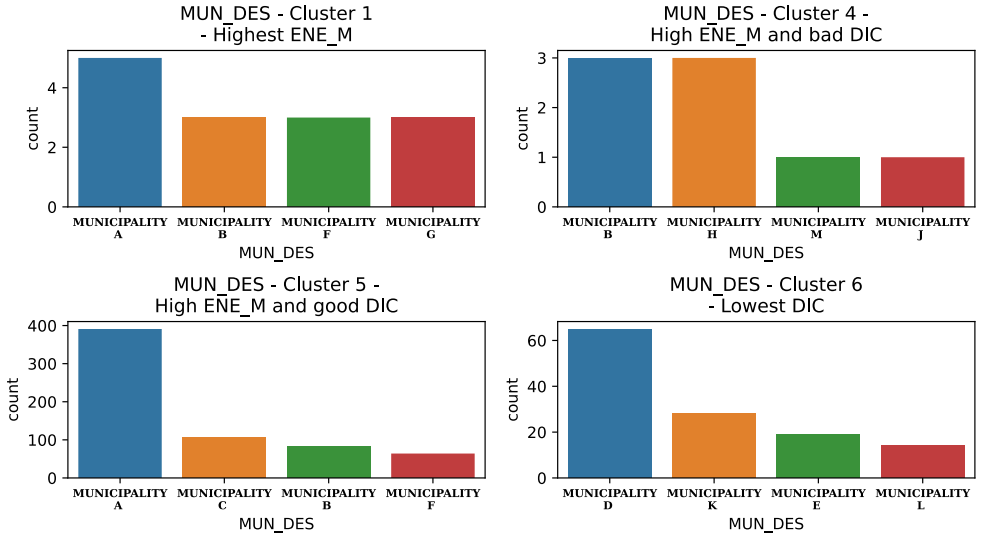
Figure 19 – Types of customers, urban UB and non-urban NU, with higher frequencies in superclusters 1, 4, 5, and 6.



Source: The author.

Regarding the municipalities present in each group, smaller municipalities are presented in the two clusters with the worst DIC indicators, clusters 4 and 6, as shown in Figure 18. However, there are also large municipalities with consumer units with bad DIC, such as the case of municipalities B and H, the most frequent municipalities in cluster 4, the second cluster in Figure 20.

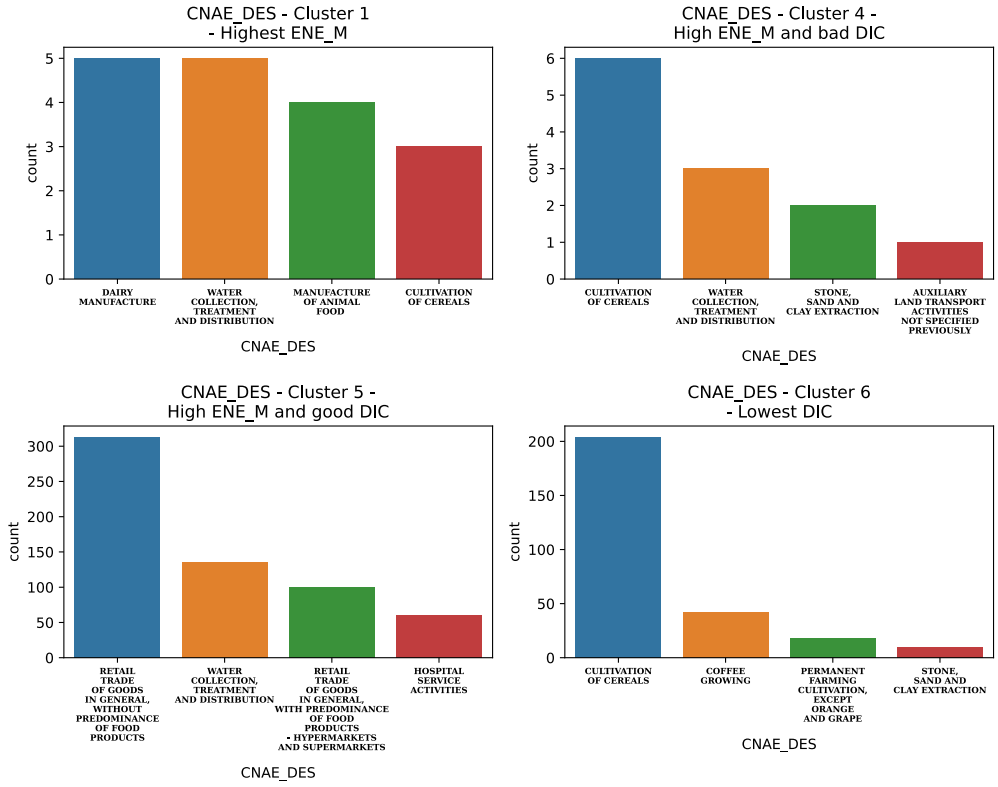
Figure 20 – Masked municipalities with higher frequencies in superclusters 1, 4, 5, and 6.



Source: The author.

As regards the activities carried out by the clusters' units, agriculture activities predominate in cluster 6, as shown in Figure 21. Also, there are water treatment companies in clusters 1, 4, and 5. It is interesting to highlight that this type of service is essential and that these units are in urban areas, indicating another scope for improvement.

Figure 21 – Most frequent activities in superclusters 1, 4, 5, and 6.



Source: The author.

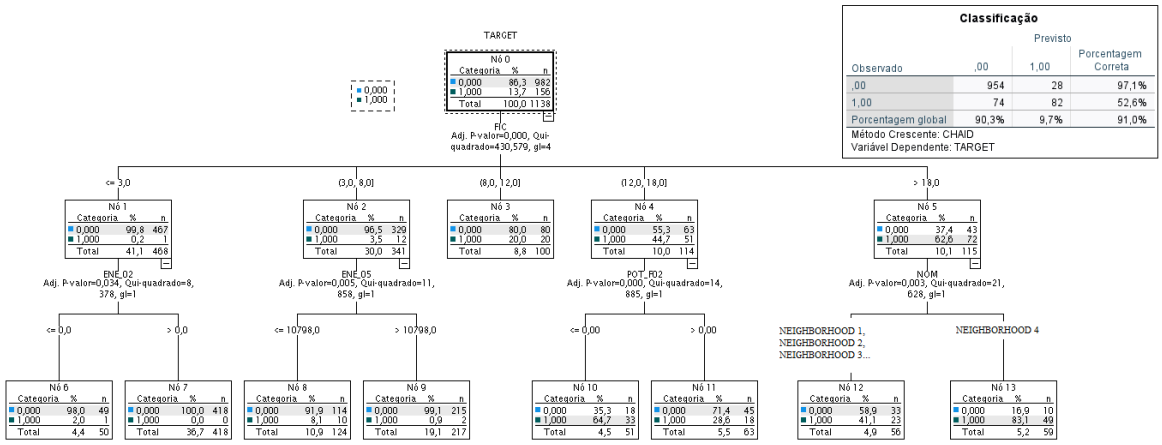
The presented SOM was shown to be capable of grouping the electrical consumers, reflecting the relations between the variables, such as DIC and consumption category (rural, industrial, commercial). Patterns were found according to the objectives of the work, such as, the understanding of the energy consumption throughout the year of some clusters.

In addition, the hypothesis raised in the methodology that there were power interruption problems was confirmed in the initial analysis. Figures 19, 20, and 21 shows the formation of clusters of BDGD customers with low and very low DIC (Electricity Continuity Index – Duration of interruptions). Through these results obtained by the cluster analysis of the SOM neural network, it is possible to see that the power quality indicators. More precisely related to interruptions, had a marked impact on cluster formation and revealed an existing problem in the power supply process. As mentioned in section 4.1, about data sources considerations, the interruption indexes in over than limits is 23.41%. This data combined with the results of clusters formed by SOM, supports the creation of the target variable described in the implementation strategies section, validate the reason why it is possible to understand the patterns of this problem, and allows us to conclude that the main objective of this research is feasible to be solved through artificial intelligence models. Therefore, in the following sections, these models will be presented and discussed.

5.3 Feature extraction and selection – CART and CHAID

The feature extraction and selection were implemented by the analysis of scenario 2, as presented on section 4.1, whose initial analysis starts with 130 brute variables and 852 transformed. Using variables selection of two models of decisions tree (CHAID and CART), their numbers were reduced to 8 brute variables and 236 transformed. The results of the models are shown in Figure 22 and Figure 23.

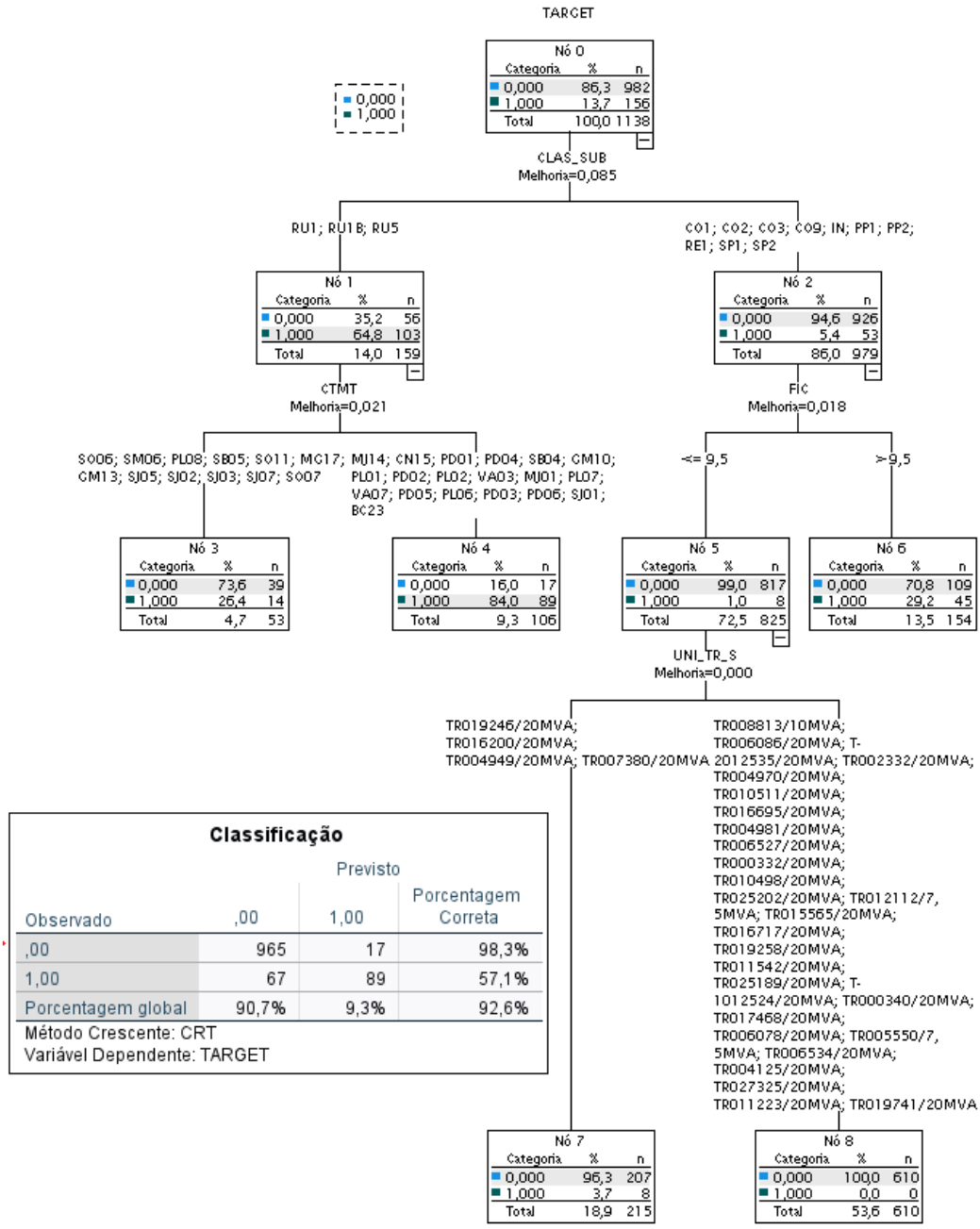
Figure 22 – Rules of Classification - Decision Tree CHAID.



Source: The author.

A decision tree CHAID based on chi-square, reveals the five best nodes that visually explain the relationship between the variables and the interruptions. To understand this, the tree shows that the greater the frequency of interruptions, the greater the chance of interruptions above the limit stipulated by the regulatory agency in the next year, especially in specific neighborhoods. In contrast, for certain customers, the lower the number of interruptions in the current period and the higher the energy consumption in the fifth month, the lower the chance of interruptions outside the norm in the subsequent year.

Figure 23 – Rules of Classification - Decision Tree CART.



Source: The author.

A decision tree CART builds its rules based on feature importance, generating binary nodes which have shown that rural networks have more chance of interruptions than urban networks, especially in specific circuits. However, when urban networks have more than 9.5 interruptions, they probably will have in the next year interruption indexes outside the norm in the subsequent year.

The features selected for the training and validation of neural networks were:

- ENE_02: measured active energy of the 2nd period (kWh)
- ENE_05: measured active energy of the 5th period (kWh)
- DIC: the annual duration values (in hours) of the unit's individual interruptions in the year prior to the target
- FIC: the annual frequency values of the unit's individual interruptions in the year prior to the target
- POT_F02: apparent rated power with forced ventilation 02 (MVA)
- NOM: substation name
- CLAS_SUB: description of class and subclass (Commercial, Rural, Public Service, Commerce)
- CTMT: Medium Voltage Circuit Description
- UNI_TR_S: Substation Transformer Unit Description

The matrix of correlations of the variables is shown in Figure 24:

Figure 24 – Correlation Matrix of numeric features.

Source: The author.

Analyzing the matrix of correlations, it is possible to deduce that the frequency and duration are much correlated. A strong correlation between ENE_02 and ENE_05 was expected because of the consumption that doesn't have significant variance over the months in the same year. None of the variables have a direct correlation with the target variable.

5.4 The baseline neural network implementation

The simple implementation, called baseline, was constructed with one input layer that had 16 neurons and activation ReLU activation. A dropout layer has a rate of 0.5., an output layer with 1 neuron and sigmoid activation. The optimizer was implemented with Adam algorithm, using a learning rate of 0.001 and the loss function used was binary crossentropy, as shown in Figure 25.

Figure 25 – Baseline algorithm parameterization.

```
model = keras.Sequential([
    keras.layers.Dense(
        16, activation='relu',
        input_shape=(train_features.shape[-1],)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1, activation='sigmoid',
                       bias_initializer=output_bias),
])
```

```
model.compile(
    optimizer=keras.optimizers.Adam(lr=1e-3),
    loss=keras.losses.BinaryCrossentropy(),
    metrics=metrics)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                3744
_________________________________________________________________
dropout (Dropout)            (None, 16)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 17
=================================================================
Total params: 3,761
Trainable params: 3,761
Non-trainable params: 0
```

Source: The author.

5.5 The fully connected neural network implementation

The implementation developed from the baseline was created with two layers fully connected. The first input layer had 256 neurons and ReLU activation. A dropout layer with a rate of 0.5 was applied, and a second input layer was parameterized with 75 neurons and ReLU activation, suggested by the process of tuning. After this was configured an output layer with 1 neuron and sigmoid activation. The optimizer was implemented with Adam algorithm with a learning rate of 0.001 and the loss function used was binary crossentropy, as shown in Figure 26.

Figure 26 – Fully connected neural network algorithm parameterization.

```
model = keras.Sequential([
    keras.layers.Dense(
        256, activation='relu',
        input_shape=(train_features.shape[-1],)),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(
        75, activation='relu',
        input_shape=(train_features.shape[-1],)),
    keras.layers.Dense(1, activation='sigmoid',
                       bias_initializer=output_bias),
```

```
model.compile(
    optimizer=keras.optimizers.Adam(lr=1e-3),
    loss=keras.losses.BinaryCrossentropy(),
    metrics=metrics)
```

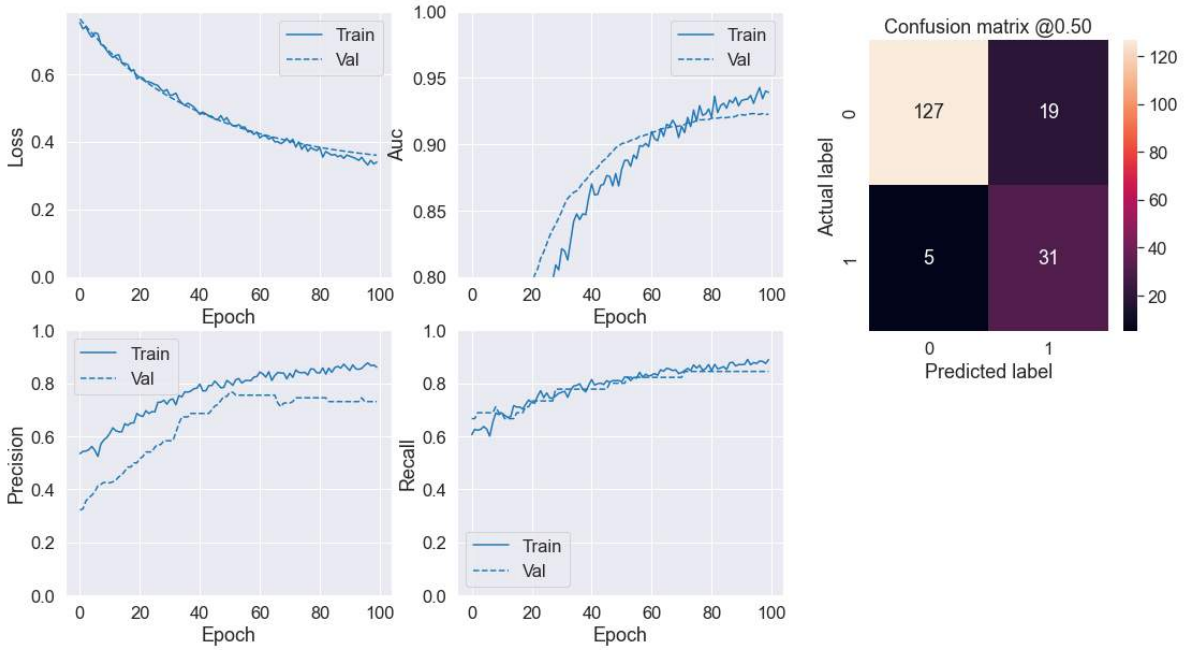
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 256)               58880
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 75)                19275
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 76
=================================================================
Total params: 78,231
Trainable params: 78,231
Non-trainable params: 0
```

Source: The author.

5.6 Imbalanced, balanced per weight class, and oversampling training

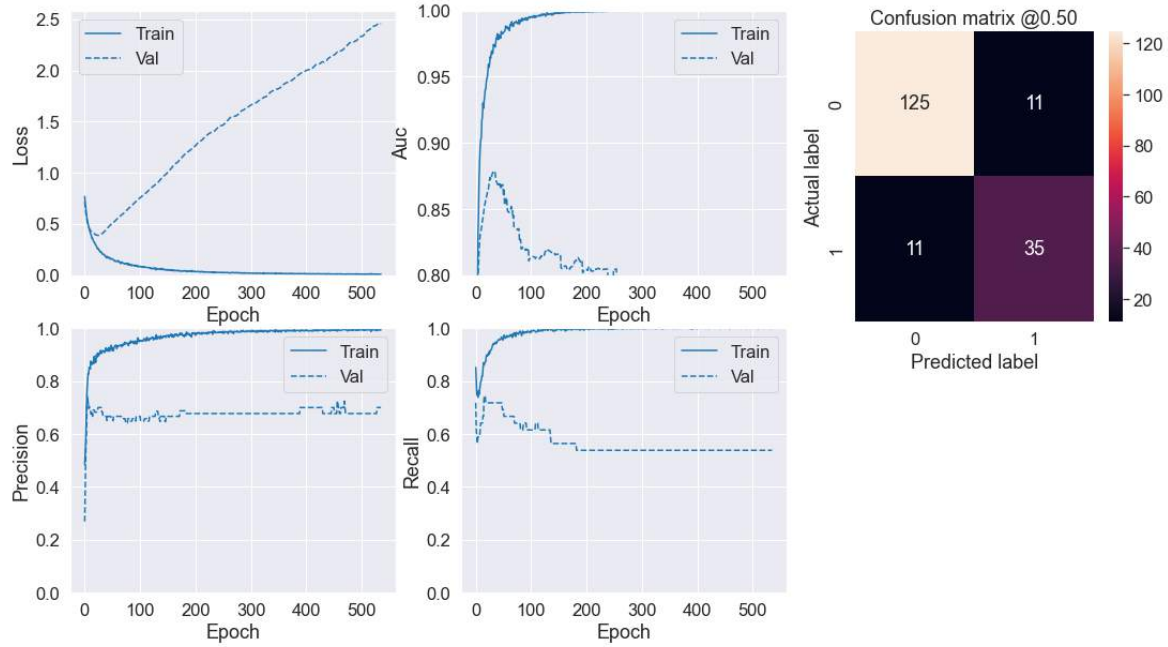
The baseline implementation was trained with different samples (imbalanced, balanced per weight class, and oversampling) after 100 epochs, the best results were generated by the samples of oversampling with 0.62 of precision, 0.86 of recall, 0.72 of f1-score, and 0.91 of AUC. These results revealed that the problem of interruptions can be predicted, and precision metrics improved. To understand if it was possible to achieve better precision and recall, more layers and neurons were added. Although, after 1000 epochs the precision increased to 0.76 with the fully connected neural network, the recall decreased to 0.76, the f1-score increased to 0.76 and the AUC got very close at 0.90 with a sample generated by oversampling. The result with the fully connected neural network can be considered better due to the f1 score, which is a harmonic mean of precision and recall, that had better results compared to the baseline with the same type of sample. The plots and metrics of training can be visualized in Figures 27, 28, and Tables 1, 2.

Figure 27 – Confusion matrix and plots of precision, loss, recall, and AUC on the training and validation of baseline algorithm.



Source: The author.

Figure 28 – Confusion matrix and plots of precision, loss, recall, and AUC on the training and validation of fully connected neural network algorithm.



Source: The author.

e

Table 1: Results of baseline algorithm using different trainings (imbalanced, balanced per weight class, and oversampling training).

Imbalanced	Balanced per Weight Class	Oversampling
precision: 0.45	precision: 0.45	precision: 0.62
recall: 0.75	recall: 0.75	recall: 0.86
f1 score: 0.56	f1 score: 0.56	f1 score: 0.72
auc: 0.85	auc: 0.91	auc: 0.90

Source: The author.

Table 2: Results of fully connected neural network algorithm using different trainings (imbalanced, balanced per weight class and oversampling training)

Imbalanced	Balanced per Weight Class	Oversampling
precision: 0.83	precision: 0.79	precision: 0.76
recall: 0.56	recall: 0.73	recall: 0.76
f1 score: 0.66	f1 score: 0.75	f1 score: 0,76
auc: 0.89	auc: 0.89	auc: 0.90

Source: The author.

Chapter 6

CONCLUSION

6.1 Introduce

Techniques of data analyzing were used in a real database. The detailed examples with synthetic data contribute to a better understanding of the SOM parameters tuning. The presented neural network was shown to be capable of predicting exceeded interruption indexes of medium voltage electrical consumers. Based on the available variables, the result of the proof of concept is interesting. Even with the uncertainties that are inherent to the problem of power outages, the model manages to explain the interruptions, that is, the continuity indices, for the following year. In addition to showing the factors that best explain the target variable studied, it provides a better prediction than basic (random) algorithms.

6.2 Main contributions

The main contributions of this work are related to the success in demonstrating how to create a proof of concept capable of profiling interruptions and explaining interruptions simply. The data and facts obtained, support the achievement of the main purpose objective of this paper, such as accuracy of up to 76% to predict the interruptions, which can be considered a good result for the metric.

In addition, the data allowed mining the Geographic Databases of Distributors and creating a proof of concept capable of profiling outages and explaining outages, even with the uncertainties that are inherent to the problem of power outages, the model manages to explain the interruptions. That is, the continuity indices, for the following year. As well as subsidizing academics for future work and correlated research, reveals existing problems in consumer units so that concessionaires and regulatory bodies to better fulfill their responsibilities.

6.3 Future Work

As for future work, it can be mentioned the implementation of improved SOMs with the treatment of categorical variables. It is possible to estimate loads installed in residential consumer units, using a Brazilian possession and consumption habits research base, based on

electrical equipment data from consumer units. Lastly is possible to extend the implementation to low voltage consumers, since the same variables exist for this type of consumer unit, it is only necessary to replicate the model and analyze the results.

PUBLICATIONS RELATED

The publications related to this dissertation were:

Rosa, G. A., Oliveira Ferreira, D. D., Pinheiro, A. P., & Yamanaka, K. (2021, September). Analysis of Electricity Customer Clusters Using Self-organizing Maps. In: **Proceedings of SAI Intelligent Systems Conference** (pp. 312-325). Springer, Cham.

Rosa, G. A., Oliveira Ferreira, D. D., Pinheiro, A. P., & Yamanaka, K. (2022, September). Predicting Interruptions of Medium Voltage Customers Using Fully Connected Networks. In: **Proceedings of SAI Intelligent Systems Conference** (In process of publication). Springer, Cham.

BIBLIOGRAPHIC REFERENCES

ABELLÁN, J.; LÓPEZ, G.; DE OÑA, J. Analysis of traffic accident severity using decision rules via decision trees. **Expert Systems with Applications**, v. 40, n. 15, p. 6047-6054, 2013. https://doi.org/10.1016/j.eswa.2013.05.027

ABRADEE - ASSOCIAÇÃO BRASILEIRA DE DISTRIBUIDORES DE ENERGIA ELÉTRICA. **CP 038/19 by Technical Note No. 046/2019-SRD**. Available at: https://www.aneel.gov.br/consultas-publicas-abradee. Accessed in: out. 2021.

ADEWOLE, A. C.; TZONEVA, R.; BEHARDIEN, S. Distribution network fault section identification and fault location using wavelet entropy and neural networks. **Applied soft computing**, v. 46, p. 296-306, 2016. https://doi.org/10.1016/j.asoc.2016.05.013

ANEEL. Agência Nacional de Energia Elétrica. **Resolução Normativa nº 937, de 15 de junho de 2021**. Aprova a Revisão 15 do Módulo 6 e 3 do Módulo 10, ambos dos Procedimentos de Distribuição de Energia Elétrica no Sistema Elétrico Nacional - Prodist. Diário Oficial da União, Brasília, 15 de junho de 2021. Available at: https://www.in.gov.br/en/web/dou/-/resolucao-normativa-aneel-n-937-de-15-de-junho-de-2021-327362547. Last access: out 2021.

ARNOLD, T. B. kerasR: R Interface to the Keras Deep Learning Library. **J. Open Source Softw.**, v. 2, n. 14, p. 296, 2017. https://doi.org/10.21105/joss.00296

BAIZYLDAYEVA, U. B.; USKENBAYEVA, R. K.; AMANZHOLOVA, S. T. Decision making procedure: applications of IBM SPSS cluster analysis and decision tree. **World Applied Sciences Journal**, v. 21, n. 8, p. 1207-1212, 2013.

BHAMARE, D.; SURYAWANSHI, P. Review on reliable pattern recognition with machine learning techniques. **Fuzzy Information and Engineering**, v. 10, n. 3, p. 362-377, 2018. https://doi.org/10.1080/16168658.2019.1611030

BLUME, Steven W. **Electric power system basics for the nonelectrical professional**. 2nd ed., John Wiley & Sons, 2016. https://doi.org/10.1002/9781119180227

BOLLEN, Math H. **Understanding power quality problems. In Voltage sags and Interruptions**. Piscataway, New Jersey: Wiley-IEEE press, 2000.

BOLLEN, Math H. What is power quality? **Electric Power Systems Research**, v. 66, p. 5-14, 2003. https://doi.org/10.1016/S0378-7796(03)00067-1

BRAVO-RODRÍGUEZ, J. C.; TORRES, F. J.; BORRÁS, M. D. Hybrid machine learning models for classifying power quality disturbances: A comparative study. **Energies**, v. 13, n. 11, p. 2761, 2020. https://doi.org/10.3390/en13112761

BREIMAN, Leo. **Classification And Regression Trees (The Wadsworth statistics / problability series)**. Chapman & Hall, 1984.

DAS, C. K.; BASS, O.; KOTHAPALLI, G., MAHMOUD, T. S., & HABIBI, D. Overview of energy storage systems in distribution networks: Placement, sizing, operation, and power quality. **Renewable and Sustainable Energy Reviews**, v. 91, p. 1205-1230, 2018. https://doi.org/10.1016/j.rser.2018.03.068

DASHTDAR, M., DASHTI, R., & SHAKER, H. R. Distribution network fault section identification and fault location using artificial neural network. In: **International Conference on Electrical and Electronic Engineering (ICEEE**), 5, 2018. p. 273-278. https://doi.org/10.1109/ICEEE2.2018.8391345

DE BARROS, André Felipe Antunes. **Análise das principais causas de descontinuidade no fornecimento de energia elétrica e de seus impactos nos indicadores de qualidade.** 2020. 51p. Projeto de Graduação (Graduação em Engenharia Elétrica). Escola Politécnica, Universidade Federal do Rio de Janeiro, 2020.

FARHOUMANDI, M.; ZHOU, Q.; SHAHIDEHPOUR, M. A review of machine learning applications in IoT-integrated modern power systems. **The Electricity Journal**, v. 34, n. 1, p. 106879, 2021. https://doi.org/10.1016/j.tej.2020.106879

FARIS, H.; ALJARAH, I.; MIRJALILI, S. Training feedforward neural networks using multi-verse optimizer for binary classification problems. **Applied Intelligence**, v. 45, n. 2, p. 322-332, 2016. https://doi.org/10.1007/s10489-016-0767-1

FÁVERO, Luiz Paulo; BELFIORE, Patrícia. **Manual de análise de dados**: estatística e modelagem multivariada com Excel®, SPSS® e Stata®. 1st ed., GEN LTC, 2017.

FERREIRA, V. H.; OLIVEIRA, L. B.; PINHO, A. C.; HENRIQUES, H. O.; FORTES, M. Z.; NUNES, F. A.; POSE, A. C. A.; DE OLIVEIRA, R. B. Análise do Impacto das Ações de Manutenção nos Indicadores de Continuidade em Redes de Distribuição utilizando Machine Learning e Regressão com Dados em Painel. **Anais do Simpósio Brasileiro de Sistemas Elétricos**. Santo André, v. 1, n. 1, p. 1-6, 13 ago. 2020. https://doi.org/10.48011/sbse.v1i1.2160

GHASEMINEZHAD, M. H., & KARAMI, A. A novel self-organizing map (SOM) neural network for discrete groups of data clustering. **Applied Soft Computing**, v. 11, n. 4, p. 3771-3778, 2011. https://doi.org/10.1016/j.asoc.2011.02.009

GREGORIADES, A.; PAMPAKA, M.; HERODOTOU, H.; CHRISTODOULOU, E. Supporting digital content marketing and messaging through topic modelling and decision trees. **Expert systems with applications**, v. 184, p. 115546, 2021. https://doi.org/10.1016/j.eswa.2021.115546

HAGEDORN, S.; KLÄBE, S.; SATTLER, K. U. **Putting Pandas in a Box**. In *CIDR*, Chaminade CA, 10-13 jan. 2021. p.1-6.

HAMMANN, F.; DREWE, J. Decision tree models for data mining in hit discovery. **Expert Opinion on Drug Discovery**, v. 7, n. 4, p. 341-352, 2012. https://doi.org/10.1517/17460441.2012.668182

HAMMOND, P. W. A new approach to enhance power quality for medium voltage AC drives. **IEEE transactions on industry applications**, v. 33, n. 1, p. 202-208, 1997. https://doi.org/10.1109/28.567113

HEIDARI, A.; AGELIDIS, V. G.; POU, J.; AGHAEI, J.; GHIAS, A. M. Reliability worth analysis of distribution systems using cascade correlation neural networks. **IEEE transactions on power systems**, v. 33, n. 1, p. 412-420, 2017. https://doi.org/10.1109/TPWRS.2017.2705185

KÁDÁR-HORVÁTH, A. The Effect of Energy Prices on Competitiveness of Energy-Intensive Industries in the EU. In: **International Entrepreneurship and Corporate Growth in Visegrad Countries**, p. 129-146, 2014.

KERAS. Keras Tuner. Available at: https://keras.io/keras_tuner/. Last access: 2022, jan 15.

KHALID, S.; DWIVEDI, B. Power quality issues, problems, standards & their effects in industry with corrective means. **International Journal of Advances in Engineering & Technology**, v. 1, n. 2, p. 1, 2011.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, v. 78, v. 9, p. 1464-1480, 1990. https://doi.org/10.1109/5.58325

KUMBHAR, A.; DHAWALE, P. G.; KUMBHAR, S.; PATIL, U.; MAGDUM, P. A Comprehensive review: Machine learning and its application in integrated power system. **Energy Reports**, v. 7, p. 5467-5474, 2021. https://doi.org/10.1016/j.egyr.2021.08.133

LEVI, V.; STRBAC, G.; ALLAN, R. Assessment of performance-driven investment strategies of distribution systems using reference networks. **IEEE Proceedings-Generation, Transmission and Distribution**, v. 152, n. 1, p. 1-10, 2015. https://doi.org/10.1049/ip-gtd:20041109

MILANOVIĆ, M.; STAMENKOVIĆ, M. CHAID decision tree: Methodological frame and application. **Economic Themes**, v. 54, n. 4, p. 563-586, 2016. https://doi.org/10.1515/ethemes-2016-0029

MINGOTI, S. A.; LIMA, J. O. Comparing SOM neural network with Fuzzy c-means, K-means and traditional hierarchical clustering algorithms. **European journal of operational research**, v. 174, n. 3, p. 1742-1759, 2006. https://doi.org/10.1016/j.ejor.2005.03.039

MUHAMAD, M. I.; MARIUN, N.; RADZI, M. A. M. The effects of power quality to the industries. In: **Student Conference on Research and Development**, 5, 2007, pp. 1-4. https://doi.org/10.1109/SCORED.2007.4451410

KHALID, S., & DWIVEDI, B. Power quality issues, problems, standards & their effects in industry with corrective means. **International Journal of Advances in Engineering & Technology**, v. 1, n. 2, p. 1, 2011.

OSENI, M. O.; POLLITT, M. G. **The economic costs of unsupplied electricity:** Evidence from backup generation among African firms. 2013. 43p. Working Paper - Faculty of Economics, University of Cambridge, 2013.

PETLESHKOV, A.; PETLESHKOV, A.; LOZANOV, Y. (2019, June). Analysis of the interruptions in a section of power distribution network medium voltage 20 kV. In: **Conference on Electrical Machines, Drives and Power Systems (ELMA)**, 16, 2019, p. 1-5. https://doi.org/10.1109/ELMA.2019.8771522

PODGORELEC, V.; ŠPROGAR, M.; POHOREC, S. Evolutionary design of decision trees. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, v. 3, n. 2, p. 63-82, 2013. https://doi.org/10.1002/widm.1079

RAKHRA, M.; SONIYA, P.; TANWAR, D.; SINGH, P.; BORDOLOI, D.; AGARWAL, P.; TAKKAR, S.; JAIRATH, K.; VERMA, N. (2021). Crop Price Prediction Using Random Forest and Decision Tree Regression: A Review. **Materials Today: Proceedings**, 2021. In press. https://doi.org/10.1016/j.matpr.2021.03.261

RAMOS, M. O. D. S.; MELLO, S. S. M. M. D. Reflections of ANEEL criteria for quality in the supply of electrical energy, 2010.

ROKACH, L.; MAIMON, O. Decision trees. In: **Data mining and knowledge discovery handbook** Boston: Springer, 2005. p. 165-192. https://doi.org/10.1007/0-387-25465-X_9

RUDIN, C.; WALTZ, D.; ANDERSON, R. N.; BOULANGER, A.; SALLEB-AOUISSI, A.; CHOW, M., DUTTA, H.; GROSS, P. N.; HUANG, B. IEROME, S. ISAAC, F.; KRESSNER, A. PASSONNEAU, R. J. RADEVA, A. WU, L. Machine learning for the New

York City power grid. **IEEE transactions on pattern analysis and machine intelligence**, V. 34, n. 2, p. 328-345, 2012. https://doi.org/10.1109/TPAMI.2011.108

RUTKOWSKI, L.; JAWORSKI, M.; PIETRUCZUK, L.; DUDA, P. The CART decision tree for mining data streams. **Information Sciences**, v. 266, p. 1-15, 2014. https://doi.org/10.1016/j.ins.2013.12.060

SAFARA, F.; SOURI, A.; SERRIZADEH, M. Improved intrusion detection method for communication networks using association rule mining and artificial neural networks. **IET Communications**, v. 14, n. 7, p. 1192-1197, 2020. https://doi.org/10.1049/iet-com.2019.0502

SANNINO, A.; SVENSSON, J.; LARSSON, T. Power-electronic solutions to power quality problems. **Electric Power Systems Research**, v. 66, n. 1, p. 71-82, 2013. https://doi.org/10.1016/S0378-7796(03)00073-7

SINGH, S.; GUPTA, P. Comparative study ID3, cart and C4. 5 decision tree algorithm: a survey. **International Journal of Advanced Information Science and Technology (IJAIST)**, v. 27, n. 27, p. 97-103, 2014.

SOARES, B. N.; DA ROSA ABAIDE, A.; BERNARDON, D. Methodology for prioritizing investments in distribution networks electricity focusing on operational efficiency and regulatory aspects. In: **2014 49th International Universities Power Engineering Conference (UPEC)**, 49, 2014, p. 1-6. https://doi.org/10.1109/UPEC.2014.6934727

SOMVANSHI, M.; CHAVAN, P; TAMBADE, S.; SHINDE, S. V. A review of machine learning techniques using decision tree and support vector machine**.** In: **2016 International Conference on Computing Communication Control and Automation (ICCUBEA)**. IEEE, 2016. p. 1-7. https://doi.org/10.1109/ICCUBEA.2016.7860040

STANČIN, I.; JOVIĆ, A. An overview and comparison of free Python libraries for data mining and big data analysis. In: **International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)**, v. 42, 2019, pp. 977-982. https://doi.org/10.23919/MIPRO.2019.8757088

STRECKER, U.; UDEN, R. Data mining of 3D poststack seismic attribute volumes using Kohonen self-organizing maps. **The Leading Edge**, v. 21, n. 10, p. 1032-1037, 2002. https://doi.org/10.1190/1.1518442

TRONCHONI, A. B.; PRETTO, C. O.; DA ROSA, M. A.; LEMOS, F. A. B. Descoberta de Conhecimento em Base de Dados de Eventos de Desligamentos de Empresas de Distribuição. ***Revista de Controle & Automação*****.** v. 21, n. 2, p. 185-200, 2010. https://doi.org/10.1590/S0103-17592010000200007

ULTSCH, A. Self-organizing neural networks for visualisation and classification. In: OPITZ, O.; LAUSEN, B.; KLAR, R. **Information and classification. Studies in Classification, Data Analysis and Knowledge Organization.** Berlin, Heidelberg: Springer, 1993. p. 307-313. ISBN: 978-3-642-50974-2. https://doi.org/10.1007/978-3-642-50974-2_31

VARGAS, R.; MOSAVI, A.; RUIZ, R. Deep learning: a review. **Advances in Intelligent Systems and Computing**, v. 5, n. 2, 2017. https://doi.org/10.20944/preprints201810.0218.v1

VETTIGLI, G. **MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map**, GitHub, 2021. Available at: https://github.com/JustGlowing/minisom

VOLOSCIUC, S. D.; DRAGOSIN, M. Neural Networks Used in the Evaluation of Power Quality. **Acta Universitatis Cibiniensis**, v. 66, p. 202-207, 2015. https://doi.org/10.1515/aucts-2015-0054

WHITE, H. Learning in artificial neural networks: A statistical perspective. **Neural computation**, v. 1, n. 4, p. 425-464, 1989. https://doi.org/10.1162/neco.1989.1.4.425

YARLAGADDA, R. T. Python Engineering Automation to Advance Artificial Intelligence and Machine Learning Systems. **International Journal of Innovations in Engineering Research and Technology [Ijiert]**, v. 5, n. 6, p. 87-97, 2018.

APPENDICES A – Source Codes

A.1 Collection and Treatment of Data

```
##Defining Session Spark
import pyspark
from pyspark import SparkContext
from pyspark.sql import SparkSession
spark = SparkSession \
 .builder \
 .appName("eletricity_app") \
 .config("spark.executor.memory", "3g") \
 .getOrCreate()
from pyspark.sql import SQLContext
from pyspark.sql.functions import concat,col
from pyspark.sql.functions import lit
import pandas as pd
sqlContext = SQLContext(spark)
sqlContext = SparkSession.builder.getOrCreate()
##Defining the parameters
ANO = '2018'
##Read the variables of consuming and loads
dados = sqlContext.read.format("csv").option("header", "true").option("delimiter",
";").load(path")
dados = dados.withColumn('CHAVE', concat(lit(ANO), col('BRR'), col('ARE_LOC')))
dados = dados.withColumnRenamed("COD_ID","COD_ID_PRINCIPAL")
dados.show(n=5, truncate=False)
##Read the variable CONJ
conj = pd.read_excel(r"path", sheet_name='CONJ-2018', engine='openpyxl')
conj=spark.createDataFrame(conj)
conj=conj.drop('Shape_Area', 'Shape_Leng', 'DESCR', 'OBJECTID')
```

```
conj.printSchema()
conj.show()
```

##Read the variable CTMT

```
import pandas as pd
ctmt = pd.read_excel(r"path", engine='openpyxl')
ctmt=sqlContext.createDataFrame(ctmt)
ctmt.printSchema()
ctmt.show()
ctmt =
ctmt.drop('ENE_01','ENE_02','ENE_03','ENE_04','ENE_05','ENE_06','ENE_07','ENE_08','E
NE_09','ENE_10','ENE_11','ENE_12','PERD_A3a','PERD_A4','PERD_B','PERD_MED','PE
RD_A3a_B','PERD_A4_B','PERD_B_A3a','PERD_B_A4','PNTMT_01','PNTMT_02','PNTM
T_03','PNTMT_04','PNTMT_05','PNTMT_06','PNTMT_07','PNTMT_08','PNTMT_09','PNT
MT_10','PNTMT_11','PNTMT_12','PNTBT_01','PNTBT_02','PNTBT_03','PNTBT_04','PNT
BT_05','PNTBT_06','PNTBT_07','PNTBT_08','PNTBT_09','PNTBT_10','PNTBT_11','PNTB
T_12','DESCR','PERD_A3aA4','PERD_A4A3a', 'UNI_TR_S', 'PAC', 'SUB', 'NOM',
'OBJECTID')
ctmt.show()
```

##Read the variable SUN

```
sun = pd.read_excel(r"path")
sun=sqlContext.createDataFrame(sun)
sun=sun.drop('POS', 'Shape_Area', 'DESCR', 'Shape_Length', 'OBJECTID')
sun=sun.withColumnRenamed('NOM', 'NOM_SUB')
sun.printSchema()
sun.show()
```

##Read the variable UNTRS

```
untrs = pd.read_excel(r"path")
untrs[['BARR_3', 'PAC_3']] = untrs[['BARR_3', 'PAC_3']].astype(str)
untrs=sqlContext.createDataFrame(untrs)
untrs = untrs.drop('ARE_LOC', 'SIT_ATIV', 'CONJ', 'DAT_CON', 'MUN', 'SUB', 'DESCR',
'OBJECTID')
```

```
untrs.printSchema()
untrs.show()
##Read the variable TEN_FORN
ten_forn = pd.read_excel(r"path", sheet_name='TEN_FORN')
ten_forn[['LIMITE_DIC']] = ten_forn[['LIMITE_DIC']].astype(str)
ten_forn=sqlContext.createDataFrame(ten_forn)
ten_forn.printSchema()
ten_forn.show()
##Read the variable TARE
tare = pd.read_excel(r"path")
tare[['COD_ID']] = tare[['COD_ID']].astype(str)
tare=sqlContext.createDataFrame(tare)
tare= tare.withColumnRenamed("DESCR", "DESCR_TARE")
tare.printSchema()
tare.show()
##Read the variable TARIFF GROUP
grupo_tarifario = pd.read_excel(r"path", sheet_name='GRUPO_TARIFARIO')
grupo_tarifario=sqlContext.createDataFrame(grupo_tarifario)
grupo_tarifario=
grupo_tarifario.withColumnRenamed("DESCR","DESCR_GRUPO_TARIFARIO")
grupo_tarifario.printSchema()
grupo_tarifario.show()
##Read the variable PHASES
fases = pd.read_excel(r"path", sheet_name='FASES')
fases[['COD_ID']] = fases[['COD_ID']].astype(str)
fases=sqlContext.createDataFrame(fases)
fases= fases.withColumnRenamed("DESCR","DESCR_FASES")
fases.printSchema()
fases.show()
```

```
89  ##Read the variable CLASS
90  classes = pd.read_excel(r"path", sheet_name='CLASSE')
91  classes[['COD_ID']] = classes[['COD_ID']].astype(str)
92  classes=sqlContext.createDataFrame(classes)
93  classes= classes.withColumnRenamed("DESCR","DESCR_CLASSE")
94  classes.printSchema()
95  classes.show()
96  ##Read the index DIC/FIC yearly
97  limite_dic_fic_anual = pd.read_excel(r"path")
98  limite_dic_fic_anual=sqlContext.createDataFrame(limite_dic_fic_anual)
99  limite_dic_fic_anual.printSchema()
100 limite_dic_fic_anual.show()
101 ##Read the index DIC/FIC following year
102 dic_fic_ano_anterior = sqlContext.read.format("csv").option("header",
103 "true").option("delimiter", ";").load(r"path")
104 dic_fic_ano_anterior = dic_fic_ano_anterior.withColumn('CHAVE', concat(lit(ANO),
105 col('BRR'), col('ARE_LOC')))
106 dic_fic_ano_anterior = dic_fic_ano_anterior.select("COD_ID", "DIC",
107 "FIC").withColumnRenamed("DIC","DIC_SEG").withColumnRenamed("FIC","FIC_SEG")
108 ##Join all variaveis
109 dataset_final = dados.join(conj,
110 on=(dados.CONJ==conj.COD_ID)&(dados.DIST==conj.DIST),
111 how='left').drop(conj.COD_ID).drop(conj.DIST)\
112  .join(ctmt, on=(dados.CTMT==ctmt.COD_ID)&(dados.DIST==ctmt.DIST),
113 how='left').drop(ctmt.COD_ID).drop(ctmt.DIST)\
114  .join(sun, on=(dados.SUB==sun.COD_ID)&(dados.DIST==sun.DIST),
115 how='left').drop(sun.COD_ID).drop(sun.DIST )\
116  .join(untrs, on=(dados.UNI_TR_S==untrs.COD_ID)&(dados.DIST==untrs.DIST),
117 how='left').drop(untrs.COD_ID).drop(untrs.DIST)\
118  .join(ten_forn, on='TEN_FORN', how='left')\
119  .join(tare, on=dados.ARE_LOC==tare.COD_ID, how='left').drop(tare.COD_ID,)\
```

```
 .join(grupo_tarifario, on=dados.GRU_TAR==grupo_tarifario.COD_ID,
how='left').drop(grupo_tarifario.COD_ID)\
 .join(fases, on=dados.FAS_CON==fases.COD_ID, how='left').drop(fases.COD_ID)\
 .join(classes, on=dados.CLAS_SUB==classes.COD_ID, how='left').drop(fases.COD_ID)\
 .join(limite_dic_fic_anual, on=dados.CHAVE==limite_dic_fic_anual.CHAVE,
how='left').drop(limite_dic_fic_anual.CHAVE)\
 .join(dic_fic_ano_anterior,
on=dados.COD_ID_PRINCIPAL==dic_fic_ano_anterior.COD_ID, how='left')
dataset_final.show()
dataset_final.count()
dataset_final = dataset_final.filter(col('MAX_DIC').isNotNull())
dataset_final.count()
dataset_final = dataset_final.filter(col('MAX_FIC').isNotNull())
dataset_final.count()
dataset_final = dataset_final.filter(col('DIC_SEG').isNotNull())
dataset_final.count()
dataset_final = dataset_final.filter(col('FIC_SEG').isNotNull())
dataset_final.count()
dataset_final.printSchema()
pandasDF2=dataset_final.toPandas()
pandasDF2.to_csv('ETL_VARIAVEIS_ELETRICITY_11-12.csv', sep='|', header=True,
index = False)
```

A.2 Training The Baseline Neural Network Algorithm

Importing the libraries

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.optimizers import Adam
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext
from pyspark.ml.feature import OneHotEncoder, StringIndexer, StandardScaler,
VectorAssembler
```

```
from pyspark.ml import Pipeline
from pyspark.sql.functions import rand
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.sql.functions import col
from pyspark.sql.types import StringType,BooleanType,DateType, IntegerType
import numpy as np
from pyspark.sql.functions import desc, row_number, monotonically_increasing_id
from pyspark.sql.window import Window
from pyspark.sql.functions import rand
import pyspark.sql.functions as func
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Load the data

```
df = pd.read_csv('ETL_VARIAVEIS_ELETRICITY_11-12.csv', delimiter ='|')
df.loc[((df['DIC_SEG'] > df.MAX_DIC)
 |(df['FIC_SEG'] > df.MAX_FIC)), 'NORMALIDADE_ANO_SEGUINTE'] = 1
df['NORMALIDADE_ANO_SEGUINTE'] =
df['NORMALIDADE_ANO_SEGUINTE'].fillna(0)
df = df[['OBJECTID', 'ENE_02','ENE_05','DIC', 'FIC', 'POT_F02', 'NOM', 'CLAS_SUB',
'CTMT', 'UNI_TR_S', 'NORMALIDADE_ANO_SEGUINTE']]
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.2)
print("Training Data")
print(train.shape)
print(train.head())
```

```
print("Test Data")
print(test.shape)
print(test.head())
```

EDA

```
mpl.rcParams['figure.figsize'] = (12, 10)
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
train.NORMALIDADE_ANO_SEGUINTE.value_counts()
#check if it is a imbalance dataset
no, yes = np.bincount(train['NORMALIDADE_ANO_SEGUINTE'])
total = no + yes
print('Exemplos:\n Total: {}\n Positivos: {} ({:.2f}% of total)\n'.format(
 total, yes, 100 * yes / total))
train_t = train.copy()
target = train_t.pop('NORMALIDADE_ANO_SEGUINTE')
ax = sns.countplot(x = target ,palette="Set1")
sns.set(font_scale=1.5)
ax.set_xlabel(' ')
ax.set_ylabel(' ')
fig = plt.gcf()
fig.set_size_inches(10,5)
ax.set_ylim(top=800)
for p in ax.patches:
 ax.annotate('{:.2f}%'.format(100*p.get_height()/len(target)), (p.get_x()+ 0.3,
p.get_height()+800))
plt.title('Distribution of Target')
plt.show()
#Check for null values
print('Missing data in Train')
print(train.isna().any())
print('Missing data in Test')
print(test.isna().any())
#check for data types
```

```
train.dtypes
#Find the category and numeric - Convert the numeric to category variables.
cat=[]
for c in train.columns:
 if train[c].dtypes == 'object':
 cat.append(c)
print(cat)
num=[]
for c in train.columns:
 if c not in cat:
 num.append(c)
print(num)
#Change the type from numeric
cat_features = ['NOM', 'CLAS_SUB', 'CTMT', 'UNI_TR_S']
for col in cat_features:
 train[col]=train[col].astype('object')
train.dtypes
```

Feature Engineering

```
#Change the type from numeric
for col in cat_features:
 test[col]=test[col].astype('object')
test.dtypes
train = pd.get_dummies(train,drop_first=True)
test = pd.get_dummies(test,drop_first=True)
train.dtypes
col=[]
for c in train.columns:
 col.append(c)
print(col)
tcol=[]
for c in test.columns:
 tcol.append(c)
```

```
print(col)
list(set(col) - set(tcol))
```

Create train, validation, and test sets

```
train_copy = train.copy()
test_copy = test.copy()
train.shape , test.shape
train.head(5)
test.head(5)
train_df, test_df = train_test_split(train, test_size = 0.2)
train_df, val_df = train_test_split(train_df, test_size=0.2)
train_labels = np.array(train_df.pop('NORMALIDADE_ANO_SEGUINTE'))
bool_train_labels = train_labels != 0
val_labels = np.array(val_df.pop('NORMALIDADE_ANO_SEGUINTE'))
test_labels = np.array(test_df.pop('NORMALIDADE_ANO_SEGUINTE'))
#test_labels = data
train_features = np.array(train_df)
val_features = np.array(val_df)
test_features = np.array(test_df)
scaler = StandardScaler()
train_features = scaler.fit_transform(train_features)
val_features = scaler.transform(val_features)
test_features = scaler.transform(test_features)
train_features = np.clip(train_features, -5, 5)
val_features = np.clip(val_features, -5, 5)
test_features = np.clip(test_features, -5, 5)
print('Training labels shape:', train_labels.shape)
print('Validation labels shape:', val_labels.shape)
print('Test labels shape:', test_labels.shape)
print('Training features shape:', train_features.shape)
print('Validation features shape:', val_features.shape)
print('Test features shape:', test_features.shape)
pos_df = pd.DataFrame(train_features[ bool_train_labels], columns=train_df.columns)
neg_df = pd.DataFrame(train_features[~bool_train_labels], columns=train_df.columns)
```

```
import tensorflow as tf
from tensorflow import keras
```

Baseline

```
METRICS = [
 keras.metrics.TruePositives(name='tp'),
 keras.metrics.FalsePositives(name='fp'),
 keras.metrics.TrueNegatives(name='tn'),
 keras.metrics.FalseNegatives(name='fn'),
 keras.metrics.BinaryAccuracy(name='accuracy'),
 keras.metrics.Precision(name='precision'),
 keras.metrics.Recall(name='recall'),
 keras.metrics.AUC(name='auc'),
]
def make_model(metrics=METRICS, output_bias=None):
 if output_bias is not None:
 output_bias = tf.keras.initializers.Constant(output_bias)
 model = keras.Sequential([
 keras.layers.Dense(
 16, activation='relu',
 input_shape=(train_features.shape[-1],)),
 keras.layers.Dropout(0.5),
 keras.layers.Dense(1, activation='sigmoid',
 bias_initializer=output_bias),
 ])
 model.compile(
 optimizer=keras.optimizers.Adam(lr=1e-3),
 loss=keras.losses.BinaryCrossentropy(),
 metrics=metrics)
 return model
EPOCHS = 100
BATCH_SIZE = 2048
early_stopping = tf.keras.callbacks.EarlyStopping(
 monitor='val_auc',
```

```
 verbose=1,
 patience=10,
 mode='max',
 restore_best_weights=True)
model = make_model()
model.summary()
model.predict(train_features[:10])
results = model.evaluate(train_features, train_labels, batch_size=BATCH_SIZE, verbose=0)
print("Loss: {:0.4f}".format(results[0]))
initial_bias = np.log([yes/no])
initial_bias
model = make_model(output_bias=initial_bias)
model.predict(train_features[:10])
results = model.evaluate(train_features, train_labels, batch_size=BATCH_SIZE, verbose=0)
print("Loss: {:0.4f}".format(results[0]))
import os
import tempfile
initial_weights = os.path.join(tempfile.mkdtemp(), 'initial_weights')
model.save_weights(initial_weights)
model = make_model()
model.load_weights(initial_weights)
model.layers[-1].bias.assign([0.0])
zero_bias_history = model.fit(
 train_features,
 train_labels,
 batch_size=BATCH_SIZE,
 epochs=20,
 validation_data=(val_features, val_labels),
 verbose=0)
model = make_model()
model.load_weights(initial_weights)
careful_bias_history = model.fit(
 train_features,
 train_labels,
```

```
200     batch_size=BATCH_SIZE,
201     epochs=20,
202     validation_data=(val_features, val_labels),
203     verbose=0)
204   def plot_loss(history, label, n):
205     plt.semilogy(history.epoch, history.history['loss'],
206     color=colors[n], label='Train ' + label)
207     plt.semilogy(history.epoch, history.history['val_loss'],
208     color=colors[n], label='Val ' + label,
209     linestyle="--")
210     plt.xlabel('Epoch')
211     plt.ylabel('Loss')
```

A.3 Training The Fully Connected Neural Network Algorithm

Importing the libraries

```
1   from tensorflow.keras.models import Sequential
2   from tensorflow.keras.layers import Dense, Dropout, Activation
3   from tensorflow.keras import optimizers, regularizers
4   from tensorflow.keras.optimizers import Adam
5   from pyspark import SparkContext, SparkConf
6   from pyspark.sql import SQLContext
7   from pyspark.ml.feature import OneHotEncoder, StringIndexer, StandardScaler,
8   VectorAssembler
9   from pyspark.ml import Pipeline
10  from pyspark.sql.functions import rand
11  from pyspark.mllib.evaluation import MulticlassMetrics
12  from pyspark.sql.functions import col
13  from pyspark.sql.types import StringType,BooleanType,DateType, IntegerType
14  import numpy as np
15  from pyspark.sql.functions import desc, row_number, monotonically_increasing_id
16  from pyspark.sql.window import Window
17  from pyspark.sql.functions import rand
18  import pyspark.sql.functions as func
```

```
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Load the data

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.optimizers import Adam
from pyspark import SparkContext, SparkConf
from pyspark.sql import SQLContext
from pyspark.ml.feature import OneHotEncoder, StringIndexer, StandardScaler,
VectorAssembler
from pyspark.ml import Pipeline
from pyspark.sql.functions import rand
from pyspark.mllib.evaluation import MulticlassMetrics
from pyspark.sql.functions import col
from pyspark.sql.types import StringType,BooleanType,DateType, IntegerType
import numpy as np
from pyspark.sql.functions import desc, row_number, monotonically_increasing_id
from pyspark.sql.window import Window
from pyspark.sql.functions import rand
import pyspark.sql.functions as func
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

```
import sklearn
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('ETL_VARIAVEIS_ELETRICITY_11-12.csv', delimiter ='|')
df.loc[((df['DIC_SEG'] > df.MAX_DIC)
 |(df['FIC_SEG'] > df.MAX_FIC)), 'NORMALIDADE_ANO_SEGUINTE'] = 1
df['NORMALIDADE_ANO_SEGUINTE'] =
df['NORMALIDADE_ANO_SEGUINTE'].fillna(0)
df = df[['OBJECTID', 'ENE_02','ENE_05','DIC', 'FIC', 'POT_F02', 'NOM', 'CLAS_SUB',
'CTMT', 'UNI_TR_S', 'NORMALIDADE_ANO_SEGUINTE']]
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.2)
print("Training Data")
print(train.shape)
print(train.head())
print("Test Data")
print(test.shape)
print(test.head())
```

EDA

```
mpl.rcParams['figure.figsize'] = (12, 10)
colors = plt.rcParams['axes.prop_cycle'].by_key()['color']
train.NORMALIDADE_ANO_SEGUINTE.value_counts()
#check if it is a imbalance dataset
no, yes = np.bincount(train['NORMALIDADE_ANO_SEGUINTE'])
total = no + yes
print('Exemplos:\n Total: {}\n Positivos: {} ({:.2f}% of total)\n'.format(
 total, yes, 100 * yes / total))
 train_t = train.copy()
target = train_t.pop('NORMALIDADE_ANO_SEGUINTE')
ax = sns.countplot(x = target ,palette="Set1")
```

```
sns.set(font_scale=1.5)
ax.set_xlabel(' ')
ax.set_ylabel(' ')
fig = plt.gcf()
fig.set_size_inches(10,5)
ax.set_ylim(top=800)
for p in ax.patches:
 ax.annotate('{:.2f}%'.format(100*p.get_height()/len(target)), (p.get_x()+ 0.3,
p.get_height()+800))
plt.title('Distribution of Target')
plt.show()
plt.figure(figsize=(15, 12))
sns.heatmap(pd.concat([train_t, target], axis=1).corr(),annot=True , cmap='YlGnBu')
#Visualization
sns.pairplot(train,hue='NORMALIDADE_ANO_SEGUINTE')
#Check for null values
print('Missing data in Train')
print(train.isna().any())
print('Missing data in Test')
print(test.isna().any())
#check for data types
train.dtypes
#Find the category and numeric - Convert the numeric to category variables.
cat=[]
for c in train.columns:
 if train[c].dtypes == 'object':
 cat.append(c)
print(cat)
num=[]
for c in train.columns:
 if c not in cat:
 num.append(c)
print(num)
#Change the type from numeric
```

```
116 cat_features = ['NOM', 'CLAS_SUB', 'CTMT', 'UNI_TR_S']
117 for col in cat_features:
118  train[col]=train[col].astype('object')
119 train.dtypes
```

Feature Engineering

```
120 #Change the type from numeric
121 for col in cat_features:
122  test[col]=test[col].astype('object')
123 test.dtypes
124 train = pd.get_dummies(train,drop_first=True)
125 test = pd.get_dummies(test,drop_first=True)
126 train.dtypes
127 col=[]
128 for c in train.columns:
129  col.append(c)
130 print(col)
131 tcol=[]
132 for c in test.columns:
133  tcol.append(c)
134 print(col)
135 list(set(col) - set(tcol))
```

Create train, validation, and test sets

```
136 train_copy = train.copy()
137 test_copy = test.copy()
138 train.shape , test.shape
139 train.head(5)
140 test.head(5)
141 train_df, test_df = train_test_split(train, test_size = 0.2)
142 train_df, val_df = train_test_split(train_df, test_size=0.2)
143 train_labels = np.array(train_df.pop('NORMALIDADE_ANO_SEGUINTE'))
```

```
bool_train_labels = train_labels != 0
val_labels = np.array(val_df.pop('NORMALIDADE_ANO_SEGUINTE'))
test_labels = np.array(test_df.pop('NORMALIDADE_ANO_SEGUINTE'))
#test_labels = data
train_features = np.array(train_df)
val_features = np.array(val_df)
test_features = np.array(test_df)
scaler = StandardScaler()
train_features = scaler.fit_transform(train_features)
val_features = scaler.transform(val_features)
test_features = scaler.transform(test_features)
train_features = np.clip(train_features, -5, 5)
val_features = np.clip(val_features, -5, 5)
test_features = np.clip(test_features, -5, 5)
print('Training labels shape:', train_labels.shape)
print('Validation labels shape:', val_labels.shape)
print('Test labels shape:', test_labels.shape)
print('Training features shape:', train_features.shape)
print('Validation features shape:', val_features.shape)
print('Test features shape:', test_features.shape)
pos_df = pd.DataFrame(train_features[ bool_train_labels], columns=train_df.columns)
neg_df = pd.DataFrame(train_features[~bool_train_labels], columns=train_df.columns)
```

Two layer fully connected

```
import tensorflow as tf
from tensorflow import keras
METRICS = [
 keras.metrics.TruePositives(name='tp'),
 keras.metrics.FalsePositives(name='fp'),
 keras.metrics.TrueNegatives(name='tn'),
 keras.metrics.FalseNegatives(name='fn'),
 keras.metrics.BinaryAccuracy(name='accuracy'),
 keras.metrics.Precision(name='precision'),
```

```
 keras.metrics.Recall(name='recall'),
 keras.metrics.AUC(name='auc'),
]
def make_model(metrics=METRICS, output_bias=None):
 if output_bias is not None:
 output_bias = tf.keras.initializers.Constant(output_bias)
 model = keras.Sequential([
 keras.layers.Dense(
 256, activation='relu',
 input_shape=(train_features.shape[-1],)),
 keras.layers.Dropout(0.5),
 keras.layers.Dense(
 75, activation='relu',
 input_shape=(train_features.shape[-1],)),
 keras.layers.Dense(1, activation='sigmoid',
 bias_initializer=output_bias),
 ])
 model.compile(
 optimizer=keras.optimizers.Adam(lr=1e-3),
 loss=keras.losses.BinaryCrossentropy(),
 metrics=metrics)
 return model
 EPOCHS = 1000
BATCH_SIZE = 2048
early_stopping = tf.keras.callbacks.EarlyStopping(
 monitor='val_auc',
 verbose=1,
 patience=500,
 mode='max',
 restore_best_weights=True)
model = make_model()
model.summary()
model.predict(train_features[:10])
```

```
208 results = model.evaluate(train_features, train_labels, batch_size=BATCH_SIZE, verbose=0)
209 print("Loss: {:0.4f}".format(results[0]))
210 initial_bias = np.log([yes/no])
211 initial_bias
212 model = make_model(output_bias=initial_bias)
213 model.predict(train_features[:10])
214 results = model.evaluate(train_features, train_labels, batch_size=BATCH_SIZE, verbose=0)
215 print("Loss: {:0.4f}".format(results[0]))
216 import os
217 import tempfile
218 initial_weights = os.path.join(tempfile.mkdtemp(), 'initial_weights')
219 model.save_weights(initial_weights)
220 model = make_model()
221 model.load_weights(initial_weights)
222 model.layers[-1].bias.assign([0.0])
223 zero_bias_history = model.fit(
224  train_features,
225  train_labels,
226  batch_size=BATCH_SIZE,
227  epochs=20,
228  validation_data=(val_features, val_labels),
229  verbose=0)
230 model = make_model()
231 model.load_weights(initial_weights)
232 careful_bias_history = model.fit(
233  train_features,
234  train_labels,
235  batch_size=BATCH_SIZE,
236  epochs=20,
237  validation_data=(val_features, val_labels),
238  verbose=0)
239 def plot_loss(history, label, n):
240  plt.semilogy(history.epoch, history.history['loss'],
241  color=colors[n], label='Train ' + label)
```

```
242  plt.semilogy(history.epoch, history.history['val_loss'],
243  color=colors[n], label='Val ' + label,
244  linestyle="--")
245  plt.xlabel('Epoch')
246  plt.ylabel('Loss')
247 plot_loss(zero_bias_history, "Zero Bias", 0)
248 plot_loss(careful_bias_history, "Careful Bias", 1)
249 model = make_model()
250 #model.load_weights(initial_weights)
251 two_layer_history = model.fit(
252  train_features,
253  train_labels,
254  batch_size=BATCH_SIZE,
255  epochs=EPOCHS,
256  callbacks=[early_stopping],
257  validation_data=(val_features, val_labels))
258 def plot_metrics(history):
259  metrics = ['loss', 'auc', 'precision', 'recall']
260  for n, metric in enumerate(metrics):
261  name = metric.replace("","" ").capitalize()
262  plt.subplot(2,2,n+1)
263  plt.plot(history.epoch, history.history[metric], color=colors[0], label='Train')
264  plt.plot(history.epoch, history.history['val'+metric],
265  color=colors[0], linestyle="--", label='Val')
266  plt.xlabel('Epoch')
267  plt.ylabel(name)
268  if metric == 'loss':
269  plt.ylim([0, plt.ylim()[1]])
270  elif metric == 'auc':
271  plt.ylim([0.8,1])
272  else:
273  plt.ylim([0,1])
274 plt.legend()
```

```
plot_metrics(two_layer_history)
train_predictions_two_layer = model.predict(train_features, batch_size=BATCH_SIZE)
test_predictions_two_layer = model.predict(test_features, batch_size=BATCH_SIZE)
def plot_cm(labels, predictions, p=0.5):
 cm = confusion_matrix(labels, predictions > p)
 plt.figure(figsize=(5,5))
 sns.heatmap(cm, annot=True, fmt="d")
 plt.title('Confusion matrix @{:.2f}'.format(p))
 plt.ylabel('Actual label')
 plt.xlabel('Predicted label')
```

Two layers (Imbalanced)

```
two_layer_results = model.evaluate(test_features, test_labels,
 batch_size=BATCH_SIZE, verbose=0)
for name, value in zip(model.metrics_names, two_layer_results):
 print(name, ': ', value)
print()
plot_cm(test_labels, test_predictions_two_layer)
def plot_roc(name, labels, predictions, **kwargs):
 fp, tp, _ = sklearn.metrics.roc_curve(labels, predictions)
 plt.plot(100fp, 100tp, label=name, linewidth=2, **kwargs)
 plt.xlabel('False positives [%]')
 plt.ylabel('True positives [%]')
 plt.xlim([-0.5,20])
 plt.ylim([80,100.5])
 plt.grid(True)
 ax = plt.gca()
 ax.set_aspect('equal')
plot_roc("Train two_layer", train_labels, train_predictions_two_layer, color=colors[0])
plot_roc("Test two_layer", test_labels, test_predictions_two_layer, color=colors[0],
linestyle='--')
plt.legend(loc='lower right')
```

Model with Class Weights

```
weight_for_0 = (1 / no)(total)/2.0
weight_for_1 = (1 / yes)(total)/2.0
class_weight = {0: weight_for_0, 1: weight_for_1}
print('Weight for class 0: {:.2f}'.format(weight_for_0))
print('Weight for class 1: {:.2f}'.format(weight_for_1))
weighted_model = make_model()
weighted_model.load_weights(initial_weights)
weighted_history = weighted_model.fit(
 train_features,
 train_labels,
 batch_size=BATCH_SIZE,
 epochs=EPOCHS,
 callbacks=[early_stopping],
 validation_data=(val_features, val_labels),
 # The class weights go here
 class_weight=class_weight)
plot_metrics(weighted_history)
train_predictions_weighted = weighted_model.predict(train_features,
batch_size=BATCH_SIZE)
test_predictions_weighted = weighted_model.predict(test_features,
batch_size=BATCH_SIZE)
weighted_results = weighted_model.evaluate(test_features, test_labels,
 batch_size=BATCH_SIZE, verbose=0)
for name, value in zip(weighted_model.metrics_names, weighted_results):
 print(name, ': ', value)
print()
plot_cm(test_labels, test_predictions_weighted)
plot_roc("Train two_layer", train_labels, train_predictions_two_layer, color=colors[0])
plot_roc("Test two_layer", test_labels, test_predictions_two_layer, color=colors[0],
linestyle='--')
plot_roc("Train Weighted", train_labels, train_predictions_weighted, color=colors[1])
plot_roc("Test Weighted", test_labels, test_predictions_weighted, color=colors[1], linestyle='-
```

```
-')
plt.legend(loc='lower right')
```

Oversampling: Oversample The Minority class

```
pos_features = train_features[bool_train_labels]
neg_features = train_features[~bool_train_labels]
pos_labels = train_labels[bool_train_labels]
neg_labels = train_labels[~bool_train_labels]
BUFFER_SIZE = 100000
def make_ds(features, labels):
 ds = tf.data.Dataset.from_tensor_slices((features, labels))#.cache()
 ds = ds.shuffle(BUFFER_SIZE).repeat()
 return ds
pos_ds = make_ds(pos_features, pos_labels)
neg_ds = make_ds(neg_features, neg_labels)
for features, label in pos_ds.take(1):
 print("Features:\n", features.numpy())
 print()
 print("Label: ", label.numpy())
resampled_ds = tf.data.experimental.sample_from_datasets([pos_ds, neg_ds], weights=[0.5,
0.5])
resampled_ds = resampled_ds.batch(BATCH_SIZE).prefetch(2)
for features, label in resampled_ds.take(1):
 print(label.numpy().mean())
resampled_steps_per_epoch = np.ceil(2.0*no/BATCH_SIZE)
resampled_steps_per_epoch
resampled_model = make_model()
resampled_model.load_weights(initial_weights)
output_layer = resampled_model.layers[-1]
output_layer.bias.assign([0])
val_ds = tf.data.Dataset.from_tensor_slices((val_features, val_labels)).cache()
val_ds = val_ds.batch(BATCH_SIZE).prefetch(2)
resampled_history = resampled_model.fit(
```

```
 resampled_ds,
 epochs=EPOCHS,
 steps_per_epoch=resampled_steps_per_epoch,
 callbacks=[early_stopping],
 validation_data=val_ds)
plot_metrics(resampled_history)
train_predictions_resampled = resampled_model.predict(train_features,
batch_size=BATCH_SIZE)
test_predictions_resampled = resampled_model.predict(test_features,
batch_size=BATCH_SIZE)
resampled_results = resampled_model.evaluate(test_features, test_labels,
 batch_size=BATCH_SIZE, verbose=0)
for name, value in zip(resampled_model.metrics_names, resampled_results):
 print(name, ': ', value)
print()
plot_cm(test_labels, test_predictions_resampled)
plot_roc("Train two_layer", train_labels, train_predictions_two_layer, color=colors[0])
plot_roc("Test two_layer", test_labels, test_predictions_two_layer, color=colors[0],
linestyle='--')
plot_roc("Train Weighted", train_labels, train_predictions_weighted, color=colors[1])
plot_roc("Test Weighted", test_labels, test_predictions_weighted, color=colors[1], linestyle='-
-')
plot_roc("Train Resampled", train_labels, train_predictions_resampled, color=colors[2])
plot_roc("Test Resampled", test_labels, test_predictions_resampled, color=colors[2],
linestyle='--')
plt.legend(loc='lower right')
```