U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSc

2.º CICLO

FCUP
2020

U.PORTO

Automated monitoring of WiBACK traffic behavior: application of machine learning algorithms to a wireless system

Francisca Guimarães Frias Rodrigues

FC

U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

MSc

Automated monitoring of WiBACK traffic behavior: application of machine learning algorithms to a wireless system

Francisca Guimarães Frias Rodrigues

Dissertação de Mestrado apresentada à Faculdade de Ciências da Universidade do Porto em Engenharia Matemática

2020

MSc

Automated monitoring of WiBACK traffic behavior: application of machine learning algorithms to a wireless system

Mestrado em Engenharia Matemática
Departamento de Matemática 2020

Orientador

Rui Pedro de Magalhães Claro Prior, Professor
Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto e Instituto de Telecomunicações

Antonio Carlos de Oliveira Júnior, Investigador Sénior
Fraunhofer Portugal AICOS

Waldir Aranha Moreira Junior, Investigador Sénior
Fraunhofer Portugal AICOS

Co-orientador

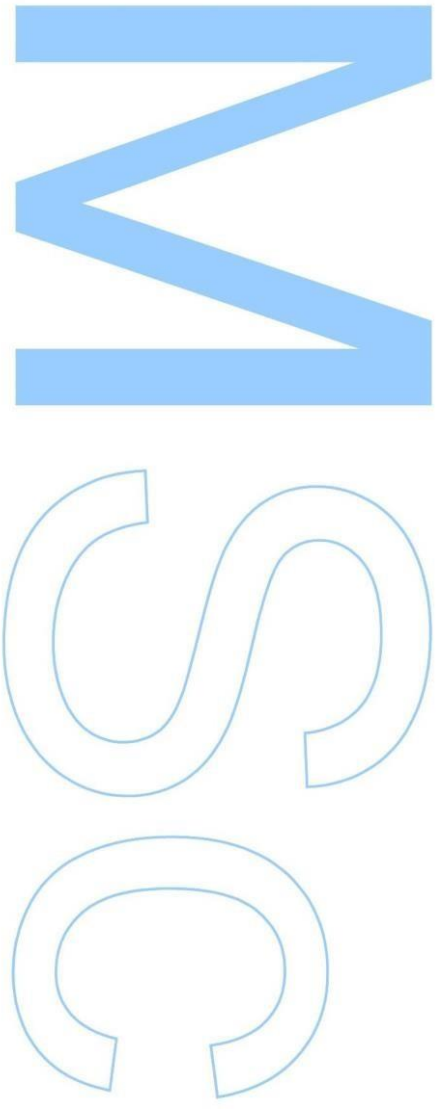
André Ribeiro da Silva de Almeida Marçal, Professor
Departamento de Matemática da Faculdade de Ciências da Universidade do Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,



Resumo

A aprendizagem de máquina, um subconjunto da inteligência artificial, tem sido aplicada para automatizar tarefas usualmente feitas pelo Homem, assim como para desenvolver outras novas, com o objetivo de melhorar a tecnologia existente ou construir mais. As vantagens de automatizar máquinas ou processos são a redução de custos, a poupança de tempo, e o acrescento de valor às comunidades.

Neste trabalho, a aprendizagem de máquina é aplicada em particular às redes sem fios. O objetivo principal é contribuir para o aumento da automatização do WiBACK, uma tecnologia que permite a criação de redes de comunicação sem fios com ligações de longa distância. As redes sem fios apresentam algumas falhas durante o seu funcionamento, o que é esperado, tendo em conta não só a capacidade limitada das ligações, mas também as condições físicas que interferem com a transmissão.

Perante isso, este trabalho foca-se na importância da relação entre erros (i.e., falhas) nas ligações, bem como as respetivas causas e soluções, procurando entender tal relação e observando se a classificação das causas pode ser feita de forma eficaz sem considerar as classes de erros, mas também se a classificação das soluções pode ser feita de forma eficaz sem considerar as classes de erros e causas. A classificação é feita apenas com variáveis numéricas como variáveis independentes e considera árvores de decisão, os k-vizinhos mais próximos e máquinas vetor de suporte.

O trabalho realizado mostra que as máquinas vetor de suporte são mais eficazes na classificação quando comparadas às árvores de decisão e aos k-vizinhos mais próximos.

Palavras chave: classificação, ligações em redes sem fios, aprendizagem de máquina, erros em ligações de redes sem fios, redes sem fios, análise de componentes principais

Abstract

Machine learning, a subfield of artificial intelligence, has been widely applied to automatize tasks usually performed by humans and to develop new tasks, to improve existent technology or build more. Advantages of automatizing machines or processes are reducing costs, saving time, and offer added value to communities.

In this work, machine learning is applied in particular to wireless networks. The main goal is to contribute to further automate WiBACK, a technology which allows the creation of wireless communication networks over high distance links. Wireless networks present faults during its function, which is expected, regarding not only the limited capacity of links but also the physical conditions that interfere with the transmission.

This work focuses on the importance of the relation between errors (i.e., fails) in links, and respective causes and solutions. The aim is to understand this relation and if the classification of causes can be accurate without considering classes of errors, but also if the classification of fixes can be accurate without considering classes of errors and causes. The classification is performed only with numerical variables as input, and considers decision trees, k-nearest neighbors, and support vector machines.

The realized work shows that support vector machines are more effective in the classification when compared to decision trees and k-nearest neighbors.

Key works: classification, wireless links, machine learning, links errors, wireless networks, principal components analysis

Acknowledgments

I want to thank Professors André Marçal and Rui Prior for the orientation in this Master Thesis and to Waldir Moreira and Antonio Oliveira from Fraunhofer Portugal AICOS, which have guided me from the beginning of my work in the best way possible. I want to thank Diana Gomes from Fraunhofer Portugal AICOS, who has always helped me. Being at Fraunhofer Portugal AICOS was a great opportunity.

I want also to thank all my colleagues and friends from AEFCUP, who have made me grown as a human being. A great thanks to my friends from the Mathematics bachelor's degree, who have been with me in good and bad moments.

This thesis is dedicated to my parents, who have also provided me with the opportunity of having a great education and academic studies, and always allowed me to do my choices with my freedom. It is also dedicated to my boyfriend and best friend, that always supported me during all my academic experience.

Contents

List of Figures

List of Tables

List of abbreviations

Abbreviation	Meaning
AICOS	*Assistive Information and Communication Solutions*
CAPEX	*Capital Expenditure*
CART	*Classification And Regression Trees*
CV	*Cross-Validation*
DT	*Decision Tree*
GS	*Grid Search*
IP	*Internet Protocol*
KNN	*K-Nearest Neighbors*
ML	*Machine Learning*
OPEX	*Operational Expenditure*
P2P	*Peer-2-Peer*
PCA	*Principal Components Analysis*
PEI	*Packet Error Indicator*
RED	*Random Early Drop*
SDN	*Software-Defined Networking*
SMOTE	*Synthetic Minority Over Sampling Technique*
SVM	*Support Vector Machines*
TCP	*Transmission Control Protocol*
WiBACK	*Wireless Backhaul Technology*
WLAN	*Wireless Local Area Network*
ZSM	*Zero-touch network and Service Management*

Chapter 1.
Introduction

Machine Learning (ML) is a subfield of artificial intelligence that has been applied to several areas to reduce costs, save time, and add value to communities' daily life. It consists of a set of mathematical models and algorithms that learn patterns existent in data, and try to generalize captured aspects to new data sets, classifying, predicting, or grouping the data. It automatizes processes and has the potential to reduce human error and solve high complexity problems. Today, these algorithms constitute an advantage to the processing, analysis, and usage of huge amounts of data produced, which is possible by the increasing computational power available. One of the areas ML has been applied to is networking [1].

However, the application of ML to networking is still relatively incipient, so it is yet possible to explore different perspectives of data analysis and to implement new and more robust tools.

The application case of this dissertation is the Wireless Backhaul Technology (WiBACK[1]), designed by the Fraunhofer Institute for Applied Information Technology (Fraunhofer FIT[2]). WiBACK is a wireless software and hardware solution with directed radio technology that provides a communication backhaul in challenged areas (i.e., rural, sparse population, without expert readily available, digital illiteracy).

It is functioning in the Zambezia province of northern Mozambique (Mocuba and Alto Molócué), Santiago Island in Cape Verde (Cidade Velha, Salineiro, and Calabaceiro), Maio Island in Cape Verde (Porto Inglês city and Barreiro, Figueira da Horta, Morro, Calheta, and Morrinho villages), Nordhorn in Germany, Brunneck in Italy, Uganda, near Branquilla in Colombia, Manizales in Colombia, and Bunda in Tanzania.

It was designed to be self-configurable and self-managed, taking care of its setup and operation

[1] WiBACK. 2020. URL: https://www.wiback.org/

[2] Fraunhofer FIT. 2020. URL: https://www.fit.fraunhofer.de/

of the network, to increase reliability and performance. However, as it is very often installed in places where there are no people with needed abilities to perform technical interventions, the goal is to further automate its management. As WiBACK is installed in places where there are no experts to fix fails, it would be advantageous for the respective communities if the software could identify problems, their causes and fixes, and fix them automatically.

1.1 Problem characterization and motivation

Several works in the literature are focused on predicting errors of wireless and wired networks, but, in general, respective causes are not even identified, and the importance of the relationship between errors, causes, and fixes is not presented. However, Alzamly's [2, 3] work, developed in Fraunhofer FIT, shows the importance of this relation. One of the goals of this thesis is to understand such relation. Other is to conclude not only if the classification of causes is accurate without considering classes of errors, but also if the classification of fixes is accurate without considering classes of errors and causes. This idea of the classification using only numerical variables, and the chosen ML algorithms, are the main differences between this thesis and the work done in Fraunhofer FIT, regarding the same software and the same problem.

To contribute to the further automatization of WiBACK, the challenge of identifying each occurrence of a link error, its cause, and a solution arises, such as to classify it with ML algorithms. To this end, first, it was necessary to study the behavior of WiBACK's networks, understand it, and create a set of errors, causes, and solutions. Then, the labeled data set had to be analyzed, and the ML algorithms created.

This dissertation is based on the WiBACK technology produced in Fraunhofer FIT, and considers as a starting point Alzamly's work (understanding of WiBACK networks links performance and the sets of errors, causes, and fixes [2, 3]). The work was developed using the open-source scikit-learn [4] library of Python for ML and data processing.

1.2 Aims and objectives

The main goal of this dissertation is to evaluate the performance and viability of ML methods to further automatize the WiBACK system. To achieve that, the following specific goals were defined:

- To learn which are the factors that affect the performance of the operation of wireless links, in particular for WiBACK networks;
- To understand numerical variables that constitute data sets, and its relation to wireless, in particular with WiBACK network's performance;
- To understand the behavior of ML algorithms with data sets provided, evaluating their performance with different evaluation metrics (accuracy, balanced accuracy, and weighed F1-score);
- To perform the classification of errors, causes, and solutions independently, regarding that a part of each data set have an exact correspondence between errors, causes, and solutions, and compare results;
- To update, improve, and compare the work already done in the automation of WiBACK, identifying which are the best approaches;
- To understand the requirements needed to perform the classification of samples online using WiBACK's software, instead of relying on offline approaches.

1.3 Contributions

Scientific contributions of this dissertation are important for the study and improvement of wireless networks, and for testing and evaluating known ML classification algorithms. Main contributions are:

- Calculation of the Euclidean distance between points of each class in each principal

component, to conclude which are the principal components that best separate the data by its classes of errors, causes, and fixes.

- Training different ML classification algorithms with different sets of numerical variables. This was done because it was not clear which was the best set of numerical variables to consider as input, regarding not just the relation between them, but also the wireless context, in which some of values are less accurate, because they are estimated.

- Analysis of the existent relation between errors, causes, and fixes. Most of the work regarding classification found in the literature focuses on binary problems. Besides, it does not present the relation between these three categorical variables. One of the objectives of this analysis was to conclude whether by classifying errors, causes, and fixes separately, the relation between them was maintained in misclassified instances. The classification of errors, causes, and solutions was performed separately (in three independent procedures) because of the existent relation between them. This relation could introduce redundancy in the classification of causes (because they depend on errors) and fixes (because they depend on errors and causes) if it was not done separately. This is the main difference between this work and Alzamly's [3].

This work also has originated two papers. The first one shows and analyzes results of the classification with the first data set provided (*Evaluation of Machine Learning Algorithms for Automated Management of Wireless Links*). The second one compares the classification of errors, causes, and fixes done separately with Alzamly's methodology [3], with the last data set collected in Fraunhofer FIT (*Applying Machine Learning Methods to Classify Wireless Link Errors, their Causes and Solutions*). The first one was accepted in the 21st International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2020. The second one was accepted in IEEE International Symposium on Personal, Indoor and Mobile Radio Communications.

1.4 Fraunhofer Portugal AICOS

This thesis was developed at Fraunhofer Portugal AICOS[3], which belongs to Associação Fraunhofer Portugal Research (Fraunhofer Portugal), a non-profit private association founded by Fraunhofer Gesellschaft, the largest organization for applied research in Europe. The thesis was a collaboration between Fraunhofer AICOS and Fraunhofer FIT, within the scope of researching artificial intelligence and, particularly, ML techniques to make a wireless network autonomous.

Fraunhofer Portugal aims on the creation of scientific knowledge capable of generating added value to its clients and partners, exploring technology innovations oriented towards economic growth, the social well-being, and the improvement of the quality of life of its end-users. It promotes and coordinates the cooperation between its research centers, other research institutions and Industry partners, with the objective of undertaking applied research of direct utility to private and public enterprises, and of wide benefit to society. This constitutes a wide benefit to society and it is currently materialized through the Research Center for Assistive Information and Communication Solutions – AICOS (Fraunhofer AICOS). AICOS' mission is to create *Remarkable Technology, Easy to Use*. As a leading partner for industry, AICOS creates applied research solutions capable of contributing to the market success of its clients' products and services by focusing on the value for their customers.

User analysis in different environments, computer vision, cognitive and decision support systems, and the internet of things are fields of study in this research center. It steers its activities towards applied research and its clients' success, with whom it establishes close cooperation for the development of innovative, intuitive, accessible and ubiquitous technological solutions. The main innovation themes are cognitive connected solutions, digital farming, accountable artificial intelligence, decentralized health technology, and living and ageing with data.

Fraunhofer Portugal is motivated to pursuit a dynamic equilibrium between application-oriented fundamental research, enabling to work ahead on solutions to problems that will not become acutely relevant to industry and society until five or ten years from now. Simultaneously,

[3]Fraunhofer Portugal. 2020. URL: https://www.aicos.fraunhofer.pt/en/home.html

Fraunhofer Portugal is developing innovative research and development projects, solving industry's pressing problems. In other words, the research work is oriented toward concrete applications and results.

1.5 Document structure

Chapter 2 presents considerations about errors, causes, and fixes made in some works, a ML overview, the application of ML classification algorithms to networks context and previous work with ML and WiBACK. In chapter 3, various data sets that were used for classification tasks, the selection and description of numerical variables, categorical variables (errors, causes, and fixes), and the technique applied to balance classes of data sets are presented. Chapter 4 describes PCA, and ML classification algorithms used in this work, namely Decision Trees (DTs), K-nearest Neighbors (KNN), and Support Vector Machines (SVM). Chapter 5 presents and discusses the experimental results. The main conclusions are drawn in chapter 6, where topics for future work are also proposed.

Chapter 2.
Machine learning for network assistance

The main goal of this chapter is to present ML techniques, mostly within the context of network ML classification applications, highlighting differences of our work, in which such techniques were applied in the scope of the WiBACK technology. The goal is also to describe challenges of ML in the practical context of improving network functioning.

First, it is presented a section regarding errors, causes, and fixes. Then, a ML overview, applications for networking, and the previous application of ML for WiBACK.

2.1 Networks: errors, causes, and fixes

The main causing factor of IEEE 802.11 wireless networks performance decreasing is detected interference [5, 6]. It includes interference of the transmitter node with its receiver in a different channel, external Wireless Local Area Network (WLAN) interference, radar, among others [2, 3, 5, 6, 7]. The solution set presented in this thesis to reduce the interference effects includes to change the channel and to increase the channel power. In the case of radio interference, the communication range information is not enough for the user to conclude if the link performance is accurate, because the interference range information could be higher. Besides, the transmission quality from a node A to a node B is not necessarily equal to the quality from B to A.

The effect of a jammer in the transmitter power level and the number of received packets [7] has been studied. A possible solution to deal with the interference is to use non-overlapping channels, which implies a minimal distance of 25MHz [7]. However, in multi-hop mesh networks, which typically involve the transmitter and receptor at the same node in different channels, the interference still affects the communication [8].

WiBACK follows a centralized Software-Defined Networking (SDN)-like approach regarding spectrum allocation and physical topology forming while minimizing internal and external interference, but they still exist. So, consequences of detected interference had also been studied in WiBACK networks, showing that there are packet losses even when the link is established and calibrated [2]. The interference in WiBACK was classified as WLAN or non-WLAN. The former could be due to another antenna, two WLAN devices interfering with each other, or two different links using the same channel and overlapping in the Fresnel zone [9]. The latter, detected on the transmitter or the receiver side, can be explained by obstacles causing signal diffraction or reflection, weather conditions (such as rain, fog, or snow), a high electricity voltage near the antenna, among others. As it is presented in this thesis, WLAN and non-WLAN interference cause between 25% and 46.9% of WiBACK network studied errors.

Anomaly detection can be performed with a classifier for each node of the network, and anomalies can be classified based on the percentage of the data that is abnormal [10]. So, it is possible to have different orders of anomalies [10]: i) partial data measurements at a node are anomalous; ii) all data measurements at a node are anomalous; and iii) data from a set of nodes is anomalous. Classifying faults in data such as that is different from the data sets labeling procedure that is presented in this thesis because labels correspond to problems such as an occupied channel or a weak link, that are not based on the number of abnormal data measurements. Besides, anomalies can be local or global, taking into account that they can be detected in one or several nodes [11], but this was also not the vision that has originated labels of data sets analyzed in this thesis.

There is a proposed approach with SVM to detect both types of anomalies without having higher energy costs, contrarily to the usual, where it is too expensive to detect global anomalies [11].

SVM were applied for fault detection, where one of the classes was named as a *random fault*, which corresponded to an instant error, a data perturbation for an instant [12]. It is similar to the *unknown* class in data sets provided for this dissertation, in the sense that it is not a specific known error. On the other hand, faults can be classified based on two aspects: the period since the fault begins until it is solved or the locality [13]. Authors [13] have classified faults into

persistent, which occurred until a fault recovery, and transient, which were temporary faults due to network congestion, weather conditions, among others. Some of the variables considered for the classification performed in this thesis are related to the airtime, but there is not a label based specifically on that neither in the locality.

Another problem with wireless networks is packet loss. One of the Transmission Control Protocol (TCP) functions is congestion control [14]. TCP was initially used in wired networks, so it identifies losses as being caused by the buffer overflow [1]. However, with the appearance of wireless networks, the cause for packet loss is not unique anymore [1]. Also regarding TCP, it could be useful to give insight about the system performance.

The performance of wireless networks is highly affected by external factors and interference, which causes errors links, and, consequently, packet loss [1]. The fact that TCP does not distinguish the cause of the error results in a reduction of links throughput [15]. Lost and corrupt packets lead to packet retransmission, which can cause packet losses in the transmitter interface [2], link throughput decreasing due to a transmission rate higher than the optimal [15], and network delay [16]. To maximize the throughput of the link, one possible solution is to dynamically adjust the transmission rate of each channel according to its conditions [15].

ML techniques were proposed to classify causes, and their implementation resulted in higher throughput, and reduction of the energy consumption, the packet loss and the end-to-end delay [14, 17]. This is an example that shows the importance of identifying not only problems of networks and their solutions, but also their causes.

2.2 Machine learning overview

ML, a subfield of artificial intelligence, could be seen as a set of algorithms that learn from examples of a training data set [18].

Learning methods can be divided into three groups: supervised, unsupervised, and semi-supervised [19]. In the first case, the data is labeled, and the machine assigns one label to each instance. In the second, the data is unlabeled and is grouped into clusters based on variables values and samples correlation. In a semi-supervised approach, the input is a mix between labeled

and unlabeled data, and the output will be grouped according to existing categories and considering identified structures on unlabeled instances. Supervised tasks split into regression and classification, if the target variable is numerical continuous or categorical, respectively [20].

The data sets used in this thesis were labeled. Labels correspond to errors of WiBACK, their causes, and possible solutions. Therefore, supervised learning algorithms were considered for data modeling purposes. However, the unsupervised and semi-supervised ML and regression models have also been applied to networking.

One of the reasons that have been motivated the scientific community and companies to apply ML instead of traditional methods is the need to analyze large amounts of complex data, extracting relevant information, suggesting, and predicting [21].

Before the application of classification algorithms, there is a need to perform variable selection, since some of them could introduce redundant information. The correlation between input variables is one of the criteria to select variables considered in the classification algorithms [22], particularly the Spearman's correlation [23], which is distribution-free, meaning that there are no assumptions related to variables distribution [24].

Overfitting is related to the use of more features than necessary (and models or parameters more complex than necessary) [25]. While training the data, there is a risk of the classifier collecting several details instead of finding the best general predictive rule, which might lead to a wrong choice of the objective function to minimize the prediction error on training [25].

Grid Search (GS) [26] runs the algorithms with all possible combinations of hyperparameters specified and gives the combination of optimal parameters that lead to the lowest error. The importance of GS to chose hyperparameters of SVM was shown in [27] with 10-fold Cross-Validation (CV).

With k-CV, the algorithm is trained and tested k times, by splitting the entire data set k times in training and test sets. The hold-out technique is more computational costly [28]. However, it is preferred when the goal is the comparison of algorithms, because the latter does not regard the variance of the training set, and CV results in an unbiased estimate of the expected value of the training set prediction error [28]. The importance of the CV when the goal is to compare supervised classification algorithms is related to the statistical power of the cross-validated t

test, and tests were performed with 10 cross-validated t test [29].

Combining hyperparameter tuning strategies (e.g., GS) with appropriate validation approaches (e.g., CV) may be adequate to reduce bias and avoid overfitting [30].

Learning systems provide an approach for labeling the data set [31]. In these systems, actions taken to solve the network's problems are observed and saved [31]. Then, the data set for training was labeled with the saved solutions [31]. This is thoroughly different from the labeling procedure performed to create data sets provided for this thesis, done based in the numerical variables and on the observation from network behavior [3].

Thinking of an online approach where the algorithm acts by itself, even if the error percentage is too low, one error could cause high costs in the network (e.g, if the network changes the channel without being needed [1]). Another concern is that fixes could not be worth it if they take too long to be implemented (e.g, to change the channel). On the other hand, the algorithm must be capable to adapt to different data distribution, avoiding overfitting. There is also the need to retrain the model every time network characteristics change [32]. There is a concern of collecting data in different periods, and different locals to train algorithms with more data [33, 34]. Data sets provided for this thesis were collected in different periods and different networks were studied [2].

2.3 Applications for networking

There is a survey [1] that initially has motivated this thesis, where the application of ML to several fields of networking, and the existent gap between these two scientific areas were shown. Traffic classification, packet loss classification, queue management, predicting fault, detecting fault, practicality, and applicability of ML, were topics explored related to this thesis. Regarding the challenges of applying ML to networking [1], there is the unfamiliarity with the possible actions to take in the software, the uncertainty of how much time fixes will take in practice, the entire knowledge of the network that the user might have to apply algorithms in the most contextualized way possible, and memory costs.

Networks and ML have other limitations that difficult the development of ML solutions [32],

such as i) data and network conditions, that can change over time and space, which may imply a non-efficient retraining of algorithms; ii) data collection, that can be costly; iii) learning errors, that can imply high costs, as aforementioned; iv) when the networks are created, their components are included by human fixed knowledge at that time, not an engineering model; v) it is not easy to design a protocol or an architecture autonomously; vi) improvements in the congestion control protocol are needed to create an algorithm that fit different network states; vii) reliable, scalable, and fast simulators are needed, as feasible real-time ML techniques, which is not always easy to develop because of possible delays; and viii) some ML algorithms (like the case of reinforcement learning) might introduce complexity, some assume that the data distribution does not change, which most of the times is not verified, and others are not interpretable.

Another issue is the increase of dynamic traffic as a consequence of the Internet of things devices, tactile Internet, virtual and augmented reality, among others [1]. Some operators fail to afford Capital Expenditure (CAPEX) costs related to the growth of technology and data [1]. WiBACK was designed with features that reduce the CAPEX and Operational Expenditure (OPEX) costs.

SDN, that facilitates the usability and programming of the software, and allows to overcome difficulties of the traditional Internet Protocol (IP), also makes easier the application of ML in this context [35]. For example, a SDN controller can probe the existence of an anomaly at a faster rate [35].

Traffic classification includes several applications, such as security and intrusion detection, service or application differentiation, performance monitoring, among others [1]. Classification of data into non-Peer-2-Peer (P2P) or P2P (among several types of P2P) is an example of traffic classification [36]. WiBACK networks form larger multi-hop topologies based on P2P and Point-to-Multipoint technologies [1].

Traffic classification with SDN seems also to be promising [37]. SDN flexibility and programmability allow the classification to be performed at the controller, making the classifier configuration easy [37].

[1]The WiBACKTM Technology in a Nutshell. Fraunhofer FIT. 2019. URL: https://www.wiback.org/content/dam/wiback/en/documents/WiBACK-in-a-nutshell.pdf

However, there is still a need to make networks receptive to these new ML-based approaches, not just by solving protocol incompatibilities, but also because networks have several devices and they are closed to new approaches. Besides, hardware switch classifiers might have capability limitations, and software switch classifiers might be slow. On the other hand, some statistical models might have delays, affecting the performance of the ML network application. If there is a lack of information when a packet is received, a delay will also occur.

The flow classification into applications could be based on the IP address and a TCP port [38], statistical application signatures [39] (e.g, mean packet size or mean flow duration), or statistical flow features [33, 40] (e.g, the number of actual packets in one direction). In the work developed in this thesis, bidirectional numerical variables like the number of packets received or the link capacity were considered (but the goal was not to perform classification into applications, it was to classify according to errors, causes, and fixes).

According to a survey [41] performed in the scope of self-organizing cellular networks, ML algorithms were evaluated based on several factors, such as the training time, the complexity, the scalability, the accuracy, among others. Particularly, KNN have been classified as low training time, complexity and scalability, and fair accuracy. SVM have been classified as fair training time and scalability, and high complexity and accuracy. DTs have been classified as low training time, complexity and accuracy, and high scalability. WiBACK has self-management, a self-organized control plane, self-healing, self-configuration, and self-optimization [2].

Another possible approach is to apply adaptive ML, which dynamically finds the necessary features to detect anomaly behavior, regarding known and unknown attacks [42]. A feature selection was performed to reduce the amount of data, and attributes considered were related to signal, channel, and packet information (as numerical variables considered in this thesis) [42]. After the clarification of which is the normal and abnormal behavior, the classifier based on distance metrics achieved a false positive rate of 0.1209%, and more than 99% detection rate [42].

As aforementioned, the network data might change over time and local, but it can also change with the transmission direction or the chosen channel (as it was also verified in WiBACK [2]).

[2]WiBACK-System. Fraunhofer FIT. 2019. URL: https://www.wiback.org/content/dam/wiback/en/documents/WiBACK-System$_e n_P B_0$8 − 04 − 2019.*pdf*

So, after traffic monitoring to clarify which was the normal behavior, a classifier for frame loss can be trained in one channel, and tested at another one, achieving a false alarm rate smaller than 0.1% and a detection rate higher than 99% [43].

Malicious and benign packet drops may be confused [44]. Besides, for example, interference and signal loss, benign drops are induced by the network due to nodes mobility, traffic density or type, channel, and fading conditions [44]. To distinguish malicious from benign drops, two ML techniques were applied [44]. As SVM require high computational resources to be implemented in real-time, the Fisher Discriminant Analysis was performed to decide if and when algorithms need to be retrained [44]. A possible approach is to train the algorithm offline, and retrain it in real-time with incremental SVM [44], including or deleting patterns instead of performing a complete training process.

Incremental SVM were also applied in online learning for autonomous intrusion detection to speed up the training phase [45]. DTs and KNN were applied to improve the congestion control on wireless networks, with the area under the curve between 0.9424 and 0.9541 [46].

The cell outage issue, common in self-organized networks, is usually done in a non-automated fashion during hours or days to solve, which has brought the necessity to apply ML techniques [41]. KNN as a global detection approach achieved accuracy up to 94% when compared to a local-outlier-factor based technique as a local approach [47]. SVM with GS and CV achieved an area under the curve between 0.7 and 0.86 [48], and a DT was also capable of detecting most of the outage events [49].

To have a system with an intellectual conscience, able to think, learn, and remember, the use of cognitive networks with artificial intelligence has been studied [50, 51]. It would be a scenario where the machine discovers automatically something wrong, and fixes it also automatically, or perceives why it can not be fixed [50, 51]. The cognitive system would be compatible with incomplete, inconsistent, or malicious information, and would work with new services that did not exist in the moment of the network creation [50, 51]. The authors [50, 51] also mentioned that, when the goal is to solve network faults, it is necessary to first understand what is the problem and the respective cause. This is the base of data sets provided for this thesis.

The elements of the networks can not make intelligent adaptations, and this is one of the

reasons to implement cognitive networks, particularly in wireless [52]. Cognitive networks base their actions on observations of the network behavior with the cooperation of machine learning techniques [52]. A cognitive network should know what is happening in the network, to improve the end-to-end performance, and it should be self-configuring, self-optimizing, self-healing, and self-protecting [53]. While these type of networks are used to improve the performance and automate the software management, they are very complex [52] and they bring some challenges (e.g., the spectrum allocation [54, 55]).

The ETSI ZSM (Zero-touch network and Service Management)[3] group was formed with the goal to accelerate the definition of the end-to-end architecture and solutions required for end-to-end automation of network and service management. The ultimate automation target is to enable largely autonomous networks which will be driven by high-level policies and rules; these networks will be capable of self-configuration, self-monitoring, self-healing and self-optimization without further human intervention.

This literature review has shown that while classification algorithms are known and applied in the networking context for several endings, the study of the relationship between errors, causes, and fixes seems not to be in-depth. Regarding the issue of which numerical features are necessary to perform the classification, there are authors concerned about it. Respecting a practical approach, where the network is capable of acting by itself (and, e.g, fix its fails) there are several constraints to solve to join networking and ML efficiently. Cognitive networks seem to be a valid and practical approach that has been applied.

2.4 Machine learning for WiBACK

As aforementioned, the behavior of WiBACK networks, in Germany, has been studied [2], and ML algorithms have been applied to the collected data [3].

Before applying ML techniques to processed and organized data sets, there was a need to create new tools in WiBACK, originating a new software version (WiBACK 4.3.4). Numerical values were related to failures happening with wireless links, creating a set of errors, their causes,

[3]ETSI ZSM. https://www.etsi.org/technologies/zero-touch-network-service-management

and possible solutions.

The first thing to know was that if a failure happens, WiBACK generates an indicator that is saved in a log file. So, logs contain the information needed. However, there were issues to solve, because the information was not properly identified neither preselected regarding the context of identifying and fixing wireless links errors.

One of the troubles was that WiBACK reports much more information than needed for Alzamly [2] to create data sets. Regarding the set of indicators generated by WiBACK, the ones considered for logs analysis were i) *link-status*, when the Packet Error Indicator (PEI) was higher than 10% or if there were no packets received; ii) *link modified*, when the configuration link (such as modulation, capacity, channel, or transmitter power) changed; iii) *detected interference*, the main packet loss cause; iv) *radar* with a certain degree of certainty if three or more radar events were detected; and v) *transmitter and receiver queue overrun* if packets in the output or input queue started do being dropped.

Regarding link-status, there are three stages. First, the link is unassigned, then it is established, and finally, the calibration step, where the link configuration (range, modulation, coding, transmitter power, capacity, and latency) is determined and originates an uncalibrated or an assigned link.

The PEI is computed by WiBACK every one second and it is an important metric because it gives an idea of the end-to-end system performance, including the transmitter, the receiver, and the path between them. Queue overrun indicators are essential, because they indicate if some buffer exceeded its capacity. A receiver queue overrun happens when the transmitter interface is faster than the receiver, and it respects to the input queue. A transmitter queue overrun happens when there is an attempt of sending too many packets simultaneously, or if the network is already congested, and it respects to the output queue. In other words, a transmitter overrun means that packets are being routed by the processor faster than the interface can send them. Alzamly [2] has considered the receiver as the peer interface and focused on transmitter overruns.

As aforementioned, the link-status indicator was reported only if the PEI was higher than 10% or if there were no packets received, but it was always needed because it contains information

like the signal quality, airtime, or transmitter power, which was useful in the network study. Now, the software reports a message before and another after the transmitted queue overrun indicator. So, the user can determine if the link-status and the queue overrun are correlated, to understand the cause of links faults.

There was another case where information was missing, because the transmission rate was reported only instantaneously. It would be useful to have a rate over time. Still related to the fact that WiBACK reports events per second, the same error had several events related to it, but they were reported separately with no identification. So, one of the tools implemented in the software was to include number tags in logs, where logs with the same tag belong to the same event.

One of the main issues that Alzamly [2] fixed was the data synchronization. The reporting period, as well as the interval of messages from different nodes, was 1000 milliseconds, but they were pushed towards the controller with a constant interval shift, as monitoring clocks were not synchronized. Additionally, delay variations could occur when the message was pushed towards the controller. For example, if there was a time shift of 500 milliseconds, 50% of the sender data would overlap with the current time interval of the receiver, while the remaining 50% would be beyond this interval (they would fall within the next time interval). Hence, processing must be deferred until the next message is received so that the complete data set for the reported transmitter queue overrun can be prepared.

Having most of the issues solved, and the software updated with new respective tools, Alzamly [2] performed six case studies before applying ML techniques, to study the network behavior, which consisted of 16 links (or 8 bidirectional links) and 16 nodes. Studies were performed with data from different periods.

The first case study has focused on the analysis of the PEI, the signal quality, the detected interference, and the relation between them, to understand the cause for the high verified values of PEI.

In general, the PER was high, but the signal quality was very good, so the reason for errors could not be the poor quality of the received signal. Analyzing the detected interference, Alzamly [2] concluded that several radios were interfering with the transmission channel, causing the

high PEI. The proposed fix in this case was to recalibrate the link. It was also verified an abrupt signal quality decrease in one link, suggesting that its configuration has changed, but there was no sufficient information to understand why. Note that when the number of packets increased, it was expected that the PEI also increased, so this value could not be analyzed isolated.

The second case study regarded radar events, queue overrun indicators, links going down, and log analysis of the signal quality. These four issues were analyzed separately.

A lot of radar events in different nodes and times were captured. To be sure that reported events in different nodes corresponded to the same source, it was needed to check the hardware timestamp and verify if the events corresponded to the same nanosecond. Regarding the proposed solution, it was to change the channel.

Several transmitted overrun events were collected, and sometimes the interface tried to send 100 or 1000 packets at a time, but it was not enough to conclude if the buffer capacity (320 packets) was exceeded. As aforementioned, the WiBACK does not provide information about a periodic transmission rate. So, to try to explain overrun events, the existence of interference was investigated. Interference was detected, which could have caused these events.

Concerning links that went down, Alzamly [2] stated that there was a need to check the source and the destination interfaces. Note that, at the time that Alzamly [2] started his work, in some cases, information that was added after software updates was missing in WiBACK logs. Concerning links quality, it was verified that sometimes the signal quality decreased significantly, which has resulted in a low PEI and, consequently, reduced the packet received rate.

The signal strength mainly depends on the transmitter power, but the registered value for the power was zero, which was unlikely to be real because the link was active (even if it was uncalibrated). There was a bug with transmitter power values, that was then fixed. The most likely cause for the link quality decrease was the decrease of the transmitter power, but it could have also resulted from WLAN interference. Interfering sources were detected and it was verified that the quality of their signals was also decreasing. A possible reason was antenna connectors and cables being wet because of the rain. It can happen, because i) due to aging, the plastic cover of an antenna might leak; ii) due to damage, the plastic cover of an antenna might leak; iii) the connector is not fully water-proof, especially if not mounted correctly; and

iv) a cold connector (from an air-conditioned room) mounted at tropical humidity might collect condensing water. Figure 2.1 shows an example of a WiBACK node wrongly placed.

Figure 2.1: Example of a WiBACK node wrongly placed (connectors are facing upward and the water will leak over time).

The third case constitutes an open problem. It was verified that the signal quality of the WiBACK channel and interfering sources decreased, but there was no problem with the hardware. The cause was unlikely to be radar events because the problem remained for eighteen hours and the radar pulses are not constant. Probably there was a third external source causing packet losses and a possible solution was to change the channel, but it was not certain since the cause was unknown.

In the fourth case study, several indicators of radar frequency were reported. In some cases, the WiBACK certainty of these events, which was supposed to increase with the increase in the number of events detected, has reached 100%, but it was not enough to conclude about the radar presence. As it was not possible to check all timestamps (in nanoseconds), it was investigated if there were matches between radar pulses in different interfaces, but none were found, which was inconclusive. The solution proposed was to change the channel, avoiding these radar pulses.

The fifth case study focused on the transmitted queue overruns indicators reported. For eleven hours, packets sent were being all dropped, but the PEI was equal to zero because as no packets were being received, there were no corrupt packets. Now, if the packet received rate is zero, an overrun indicator will be also reported. Proceeding with the overrun analysis, the goal was to find the cause for these events.

The interface did not send too many packets when overruns occurred, and there was no information about the packet rate in a period. The Random Early Drop (RED) represents the number of dropped packets when the average queue size is higher than a threshold, but RED respects to the unmanaged traffic, not to the managed traffic by WiBACK. So, RED is not always useful. Regarding the link capacity, it was computed with 100% uncertainty, deciding if the cause for overruns was a reduced link capacity was not possible. Interference and radar events were not found.

A decrease of the transmitted power could have caused the overrun, but this value remained zero for nineteen hours (which was unlikely to be correct, as aforementioned in the second case study). The hardware may be deteriorated but, in this case, the error cause was unknown.

The study proceeded, but this time, there was a decrease in the transmitted power and link capacity. The decrease in the link capacity could have caused congestion, but the packet received rate was low compared with the capacity. Besides, overruns have stopped after the capacity has decreased, and the decrease in the transmitted power was not also a consequence of the capacity change. Wireless interference was detected just after overruns have occurred. One more time, the cause for these events was unknown.

Proceeding with the last analysis, with a high RED caused by a high average queue size, which has resulted in dropped packets. The link capacity seemed to be normal, so the problem was in the interface, and the suggestion was to reduce the transmission rate to decrease or annul the number of dropped packets. Besides, wireless interference was detected once more.

The last case study has also focused on the transmitted queue overrun problem. The wireless interference was probably the cause for packet loss, which has resulted in many retransmissions. The suggestion was newly to do a link calibration.

Note that the number of errors and logs, in the studied periods, changed with the channel and interface. Besides, the same link could have different behavior in two directions.

After these detailed studies, Alzamly [2] has created the first data set. There were cases where the information provided was inconsistent (for example, having a transmission rate higher than the link capacity). To delete these instances before classification processes, the data set was carefully and manually verified (in fact, it was needed in all data sets collected).

Gaps detected in the state of the art were the analysis and classification of logs in near real-time, and the study of the correlation between events collected from different sources in real-time. So, a framework to read Syslog files, analyze the data, correlate events, and perform the classification with the relevant data for the classification was created.

Table 2.1 shows a first version of sets of errors, causes, and fixes. Some of the more recent data sets include an *unknown* error and an *unknown* fix.

Table 2.1: First sets of errors, causes, and fixes [3].

Category	Description	Causes	Fixes
Link overload	High load on the link	Self-cell overload	Adjust shaping
Channel occupied	Medium is busy due to an external signal	Foreign WLAN interferer	Change channel
		Non WLAN interferer on TX	Recalibration
		Non WLAN interferer on RX	Increase power
		Radar interference	Change channel
Corrupt packets	Receiving corrupted frames	Path Loss. Self-interference	Check Fresnel zone clearance / Antenna height
Propagation	Problem with propagate signals between two outdoor antennas.	Antenna miss pointing or wiring Self-interference	Manual Antenna fixing / Check Fresnel zone clearance / Antenna height
Broken Hardware	Network device defect	Broken Antenna by harsh weather Damaged cables and connectors	Repair or change broken equipment
Weak link	link has low throughput	Bad signal quality.	Recalibrate link
Unusable link	Frequent recalibration	All the previous causes possible	Change channel / drop link

Table 2.2 shows a description of causes, relating each one of them with numerical variables that characterize them, based on presented case studies.

Table 2.2: Relation between causes and numerical variables [3].

Cause	Observations
Self-cell overload	High own Airtime Low Alien Airtime Low retransmission
Foreign WLAN interferer	High Alien Airtime High BER MCS lower than capacity
Non WLAN interferer:	
Sender side:	Low total Airtime, Low Alien High BER
Receiver side:	Retransmissions (>10%): transmitting packet rate << receiving packet rate. High BER MCS lower than capacity → Minstrel should reduce transmitting rate because of the high BER. SNR is good Low Alien Airtime
Radar interference:	Pattern of Radar pulses
Receiving corrupted frames	Very high BER at receiver Bad received signal, will lead to packets not decoded or error on receiving packets.
Propagation / Broken Hardware, etc.	Receiving rate << Capacity High BER Bad signal

The majority of the numerical data needed for data sets was collected from *txQueueOverrun*, *linkStatus*, and *alienTraffic* messages and the remaining numerical variables were computed based on mathematical formulas with the values extracted from messages. The data set created for Alzamly's work [3] had fifty-four numerical variables that have passed through a selection process.

Initially, fewer variables were being considered, particularly there was no discrimination related to the direction of the data transmission. The last updates include new formulas, and more variables, considering links bidirectionality. Ten of the fifty-four variables were selected.

Having sets of errors, causes and fixes, and numerical variables selected, the data set was labeled based on decision rules (and manual labeling). Some variables have been added over time, and decision rules have changed many times.

Algorithms were trained with GS 10-CV, considering 75% of the data for training and 25% to test. ML algorithms chosen were a DT and two neural networks with a hidden layer, five neurons, and a logistic activation function.

The classification procedure differs from the one that is presented in this thesis. In Alzamly's work [3], errors were classified first with a neural network considering numerical variables as input. Then, causes were classified also with a neural network, considering as input, not just numerical variables, but also classes of errors that constitute the output of the first network. Finally, considering the same classes of errors, and classes of causes in the output layer of the second network, fixes were classified with a DT. The accuracy obtained was 99% with neural networks and 100% with the DT.

This procedure was performed in an offline mode, but it was created a framework for the classification in near real-time. The concept of near real-time, referred several times, was justified with messages with a delay of two seconds. This framework includes an option for the user to chose between retrain the algorithms or perform the classification with the trained ones. If there is missing data, which may occur rarely, and if the interface is highly busy with overrun events, the classification is not performed.

There are also frameworks for non-artificial intelligence-based tasks to i) detect radar events in near real-time; ii) offline analysis of link recalibration, radar events, and WLAN interference events; iii) check the correlation between transmitted queue overrun errors and radar events offline; and iv) evaluation of the radar indicator.

To evaluate the first framework, two tests were performed. The first one was in a controlled environment regarding an indoor WiBACK network of four nodes and the controller. Traffic was generated to create link overload because of self-cell overload and occupied channel because of foreign WLAN interference. The second one was performed in an uncontrolled environment with a link of an outdoor network, which was suffering from transmitted queue overrun events. Traffic was generated in different periods, and instances were classified into occupied channel because of non-WLAN interference (mostly on the sender interface) and weak link because of a bad received signal.

Alzamly [3] has considered test results promising and acceptable, respectively. Alzamly [3] concluded that one of the advantages of applying ML algorithms instead of traditional techniques was to update the training set (and also include new classes) without the need to reprogram the software. However, it is needed to understand if suggested fixes will not cause more failures or

downtimes in the network.

This thesis considers Alzamly's work [2, 3] as a starting point, as aforementioned. However, there are differences.

Starting with numerical variables, the criteria used to select them were different, hence sets of variables considered for training were also different. In this thesis, not just original unbalanced data sets, but also balanced ones were considered to test if it improved classification results.

A PCA transformation of the data was also considered, not just to reduce the space dimensionality and test algorithms with the principal components as input, but also an exploratory study. A discussion is made on which components better represented the data visually regarding the classes of errors, causes, and fixes.

ML algorithms considered were also different. In [3], neural networks and a DT were applied and, in this thesis, DTs, KNN, and SVM were considered.

However, the most significant difference is the fact that, in [3], causes classification considered error classes, and fixes classification considered errors and causes classes, while in this thesis classification procedures were performed separately. The goal of the separated classification was to conclude if categorical variables were needed as input since there is a high correspondence between their classes and there is also a redundancy between numerical and categorical variables. The relation between categorical variables resulted from the fact that, in most cases, for a certain error, the cause and the fix are always the same (e.g., when the error is a link overload, that is always because of self cell overload, and the solution is always to adjust the shaping). The relation between categorical and numerical variables resulted directly from the problem context (e.g, when there is non-WLAN interference on the receiver, there are many packets retransmissions, high PEI, and the transmission bit rate of S is much lower than the estimated link capacity), and errors, causes, and fixes are labeled based on numerical variables.

In this chapter, several works about ML for network assistance were presented. The goal was to present the importance of the relation between errors, causes, and fixes of the networks, and a ML overview regarding networking applications. The work done with the application of ML techniques with WiBACK was summarized.

Chapter 3.

Data sets and algorithms considered

This chapter starts by providing a brief presentation of the WiBACK system, which is the source for the considered data sets. Then, data sets collection, the description and selection of numerical variables, and categorical variables are presented. Synthetic Minority Oversampling Technique (SMOTE), PCA, and ML algorithms considered for the classification (DTs, KNN, and SVM) are also presented with examples of the literature.

3.1 WiBACK system

WiBACK is a wireless software and hardware solution with directed radio technology that provides a communication backhaul in challenged areas (i.e., rural, sparse population, without expert readily available, digital illiteracy). It is designed to be self-configurable and self-managed, taking care of its setup and operation of the network, increasing reliability and performance.

WiBACK is a cost-efficient multi-hop self-organizing wireless backhauling technology [1] for rural or suburban areas. To reduce the CAPEX, and especially the OPEX of such deployments, prosumer WiFi radios are utilized in the license-exempt spectrum with the optimization goal to form multi-hop networks based on minimum-interference long-distance Point-to-Point links. The WiBACK design builds upon SDN principles, and a Unified Technology Interface to program and monitor heterogeneous radios. Hence, spectrum allocations per link, as well as End-2-End data path across the network, can dynamically be adapted to changing network conditions, such as spectrum availability or traffic load.

[1] Introducing Fraunhofer's Wireless Backhaul Technology. Fraunhofer FOKUS. 2014. URL: https://www.wiback.org/content/dam/wiback/en/documents/Whitepaper_IntroducingWiBACK.pdf

The main hardware components are the controller and nodes. The controller is usually indoor, and it is the gateway between a WiBACK backhaul and a fixed network infrastructure (backbone). Most of the functionalities, such as the overall network management, are included in the controller.

3.2 Data sets collection

Different data sets (DS1 to DS5) were collected from three different networks in Fraunhofer FIT and provided for this work. Training and tests were performed with different data sets. It happened because new numerical variables were added, some formulas suffered changes, and decision rules to label errors, causes, and fixes also suffered changes resultant from the network study. On the other hand, data sets do not have always the same classes because failures that occur are not always the same, and inclusive there is a class for *unknown* error and another for *unknown* fix. Data sets have instances from different time periods, which can represent more heterogeneity.

DS1 was collected in July 23^{rd} 2019, while DS2, DS3, and DS4 have data collected from different periods. This suggests that DS1 could be more homogeneous, while DS2, DS3, and DS4 more heterogeneous, with different user behavior regarding different days. DS3 was obtained from the DS2 excluding the instances with the cause *foreign WLAN interference*. The goal of considering tests with DS3 was to compare performance results in the algorithms when a class that was not in the training set is introduced. The test with DS3 was performed also in order to analyze in which classes instances corresponding to the new class (*foreign WLAN interference*) were classified. DS5 is the data set more valid in the context because it considers the final formulas of numerical variables, the last software updates, and the last assumptions about WiBACK networks.

Numerical variables regard link characteristics, such as i) *time* spent sending the packets and used by external sources to the interfaces; ii) *signal quality*, important to inspect links faults; iii) *PEI*, which does not tell what the problem is, but how the network behavior is from a global perspective, so it is an appropriate measure to evaluate continuously; iv) *transmission and*

reception rates, which allows concluding if there is issue with links in a prior analysis; v) *link capacity* estimated by WiBACK; vi) randomly and forced *dropped packets* (due to several reasons, such as the large distance between interfaces, packets queue exceeding the buffer capacity, or weak signal link); vii) *transmission power*, which helps understand whether the number of dropped packets is due to low transmission power or other interference; and viii) *retransmission rate*, useful to compare with the number of packets sent that could be misleading if individually analyzed. Most of the variables are duplicated because it is considered the bidirectionality of the links, which means that both interfaces of the network send and receive packets. Categorical variables are errors, causes, and fixes, and represent classification targets.

Data was also collected from the WiBACK software in Porto to analyze the behavior of respective networks (one indoor and one outdoor), and test algorithms already trained. The goal of this additional data collection procedure was to compare test results with those collected by Fraunhofer FIT, with different equipment, and to understand the capability of each algorithm to generalize classification results. However, this data did not present enough numerical information, which could be explained by the fact that there was no significant number of users in the areas covered by networks and the software version that was not the most recent one.

3.3 Description and selection of numerical variables

Considering links bidirectionality is relevant because the behavior is not necessarily similar in different directions of the link, which means that having errors in one direction does not imply errors in the other. S was the interface that suffers from transmitted queue overrun, and that is why some values were monitored just in this direction, while R was the peer interface. Down (or outcoming) is when S is transmitting packets and R is receiving, and up (or incoming) is when R is transmitting packets and S is receiving them.

Constant variables, variables related to the identification of interfaces, timestamp, and auxiliary variables for the data synchronization were excluded because they did not give relevant information for the classification. Constant ones included the number of packets that the buffer interface could hold at once (queue length), and the number of radar events detected in both

directions. Radar events were always equal to 0 in data sets considered, as the certainty of those events. Some variables depend mathematically on others. Table 3.1 briefly describes each numerical variable, but some of them do not belong to all data sets, as the set of collected variables have changed over time.

Table 3.1: Numerical variables description.

Variable name	**Unit**	**Description**
AT_out	-	Fraction of time that packets sent by S and received by R stayed in the air. It was computed by WiBACK based on the number of packets sent by S, their size and the time they stayed in the air. Link distance is crucial for this calculus.
AT_in	-	Fraction of time that packets sent by R and received by S stayed in the air. It was computed by WiBACK based on the number of packets sent by R, their size and the time they stayed in the air. Link distance is crucial for this calculus.
Alien_out	-	Fraction of time used by other sources on S.
Alien_in	-	Fraction of time used by other sources on R.
AT_Alien	-	Total time used by other sources, given by the sum of *Alien_out* and *Alien_in*.
Total_Airtime	-	Total airtime used by transmissions, given by the percentage of the sum of *AT_Alien*, *Alien_out* and *Alien_in*.
SQ_down	dB	Quality of the received signal by R, or the strength of S's signal.
SQ_up	dB	Quality of the received signal by S, or the strength of R's signal.
PEI_down	%	Packet Error Indicator of S's transmission to R. It is calculated by $\frac{16(4(Lo+D)+La)}{G+Lo}$, being Lo, D, La, and G, respectively, the number of lost, duplicated, good, and late packets. It is calculated regarding the transmission from S to R. Note: this is the updated formula, but it suffered changes over time.
PEI_up	%	Packet Error Indicator of R's transmission to S. It is calculated by $\frac{16(4(Lo+D)+La)}{G+Lo}$, being Lo, D, La, and G, respectively, the number of lost, duplicated, good, and late packets. It is calculated regarding the transmission from R to S. Note: this is the updated formula, but it suffered changes over time.
MCS_down	Kbps	Average of the transmission bit rate on S during the period of link establishing, measured on R.

MCS_up	Kbps	Average of the transmission bit rate on R during the period of link establishing, measured on S.
Cap_down	Kbps	Link capacity estimated by WiBACK for the direction from S to R. Ideally, it would be between 75% and 80% of the *MCS_down*. When WiBACk estimated this value, it had in account the distance between antennas (or interfaces), the signal quality, MCS, and the transmission power in this direction. This is the major difference between *Cap_down* and *Rx_br_down*.
Cap_up	Kbps	Link capacity estimated by WiBACK for the direction from R to S. Ideally, it would be between 75% and 80% of the *MCS_up*. When WiBACK estimated this value, it had in account the distance between antennas (or interfaces), the signal quality, MCS, and the transmission power in this direction. This is the major difference between *Cap_up* and *Rx_br_up*.
Variable name	**Unit**	**Description**
Cap_of_MCS_down	%	Ratio between *Cap_down* and *MCS_down*.
Cap_of_MCS_up	%	Ratio between *Cap_up* and *MCS_up*.
Tx_br_down	Kbps	Transmission bit rate of S.
Tx_br_of_Cap_down	%	is the percentage ratio between *Tx_br_down* and *Cap_down*.
Rx_br_down	Kbps	Received bit rate of S.
Rx_br_of_Cap_down	%	Ratio between *Rx_br_down* and *Cap_down*.
Rx_br_up	Kbps	Received bit rate of R.
Rx_br_of_Cap_up	%	Ratio between *Rx_br_up* and *Cap_up*.
Cap_utilized	%	Sum of *Rx_br_of_Cap_down* and *Rx_br_of_Cap_up*.
Tx_pr_down	Pps	Transmission packet rate of S.
RED	Pps	Number of Random Early Drop packets per second on S.
Overrun	Pps	Number of forced dropped packets per second on S.
Drop_Ratio	%	Ratio between the number of dropped packets on S and the number of transmitted packets by S.
Rx_pr_down	Pps	Received packet rate of R.
Rx_pr_up	Pps	Received packet rate of S.
TxPower_down	dBm	Transmission power of S.
TxPower_up	dBm	Transmission power of R.
rTx	Pps	Number of retransmitted packets from S to R calculated by the TCP.

Units legend:

dB: decibel: **Kbps:** kilobits per second; **Pps:** packets per second; **dBm:** decibel milliwatt.

As it is usual in wireless communications, *Cap_up* and *Cap_down* were estimated by the software, which means that these values are not 100% accurate. Some of the instances exhibited values *Cap_of_MCS_down* and *Cap_of_MCS_up* higher than 100%, and consequently, *Tx_br_of_Cap_down*, *Rx_br_of_Cap_down* and *Rx_br_of_Cap_up* higher than 100% too, which was not an expected output. These instances were, therefore, removed from data sets.

Instances with *RED* equal to 1 and *overrun* equal to 0 were also excluded because that dropped packet was probably just a consequence of TCP trying to reach the maximum transmission rate and not a problem that needed a fix.

Considering more variables than needed as input to the classification could lead to overfitting, more computational effort, and higher training times. Consequently, a selection was performed in which variables with high correlations (absolute value of Spearman's correlation higher than 0.8) were excluded. Recommendations regarding their importance in the problem context were also taken into account, as the fact that some were more accurate than others regarding the way they were obtained.

Table 3.2 shows numerical variables of each set after this selection. DS5 - 1, DS5 - 2, and DS5 - 3 correspond to the instances of DS5 with different sets of numerical variables selected with different criteria to train algorithms. For DS5 - 1, variables were selected based on the Spearman's correlation (variables with absolute value of correlation higher than 0.8 were excluded). For DS5 - 2, variables were selected based on the Spearman's correlation (variables with absolute value of correlation higher than 0.8 were excluded), but less important or less accurate variables regarding the problem context had been also taken into account to the exclusion. For DS5 - 3, variables were the same used in one of the methodologies presented in the paper *Applying Machine Learning Methods to Classify Wireless Link Errors, their Causes and Solutions* mentioned in Section 1.3.

The data was scaled (has mean equal to zero and unit variance).

Table 3.2: Numerical variables considered in each set.

Variable name	DS1, DS2, DS3, DS4	DS5 - 1	DS5 - 2	DS5 - 3
AT_out	x		x	x
AT_in	x	x	x	x
Alien_out		x	x	
Alien_in	x			
AT_Alien	x	x	x	x
Total_Airtime		x	x	x
SQ_down	x	x		x
SQ_up	x			x
PEI_down	x	x	x	x
PEI_up	x	x	x	x
MCS_down	x			
Cap_down	x			
Cap_up	x			
Cap_of_MCS_down	x	x	x	
Cap_of_MCS_up		x	x	
Rx_br_down	x	x		
Rx_br_up	x	x		
Rx_br_of_Cap_up		x		
Tx_br_down	x			
Tx_br_of_Cap_down			x	
Cap_utilized			x	x
Tx_pr_down			x	x
RED	x	x	x	x
Overrun	x	x	x	x
Drop_Ratio		x	x	x
TxPower_down	x	x	x	
TxPower_up	x	x	x	
rTx	x	x	x	

3.4 Categorical variables

As aforementioned, classes of errors, causes, and fixes are not necessarily the same in different data sets, which leads to heterogeneity between them. Table 3.3 shows the relative frequency of classes in each training (T) and test set. It is organized with blocks that correspond to tests performed with each training set, in order to enable a comparison between the relative frequency of each class between the training and each test set.

Different categorical variables (errors, causes, and fixes) have some equal occurrence frequencies. For example, when there is a link overload, that is because of self cell overload, and the solution is to adjust the shaping. In fact, these instances are always the same. One of the goals of classifying errors, causes, and solutions separately was to analyze if the correspondence between misclassified instances of errors, causes, and solutions maintained, since these correspondences between categories represent a great part of data sets.

As Table 3.3 shows, classes are unbalanced. To understand the influence of that in the classification performance, the classification was performed not just with algorithms trained by the original unbalanced data sets, but also with data sets balanced by the SMOTE.

Table 3.3: Relative frequency of each class of errors, causes, and solutions in each training (T) and test set.

Classes of errors	**DS1 (T)**	**DS2**	**DS3**	**DS4**	**DS5 (T)**
E1 - Link overload	**0.625**	0.569	0.593	0.532	**0.454**
E2 - Occupied channel	**0.250**	0.431	0.407	0.468	**0.329**
E3 - Weak link	**0.122**	0	0	0	**0.211**
E4 - Unknown	**0.003**	0	0	0	**0**
E5 - Corrupt frames	**0**	0	0	0	**0.005**
Classes of causes	**DS1 (T)**	**DS2**	**DS3**	**DS4**	**DS5 (T)**
C1 - Self cell overload	**0.625**	0.569	0.593	0.532	**0.454**
C2 - Non-WLAN interference on the sender	**0.240**	0.273	0.284	0.291	**0.221**
C3 - Bad signal	**0.125**	0	0	0	**0.150**
C4 - Foreign WLAN interference	**0**	0.041	0	0.036	**0.037**
C5 - Non-WLAN interference on the receiver	**0.010**	0.118	0.123	0.142	**0.072**
C6 - Path loss/self interference	**0**	0	0	0	**0.067**
Classes of fixes	**DS1 (T)**	**DS2**	**DS3**	**DS4**	**DS5 (T)**
F1 - Adjust the shapping	**0.625**	0.569	0.593	0.532	**0.454**
F2 - Change the channel	**0.240**	0.314	0.284	0.327	**0.304**
F3 - Recalibrate the link	**0.122**	0	0	0	**0.150**
F4 - Increase the transmitted power	**0.010**	0.118	0.123	0.142	**0.026**
F5 - Unknown	**0.003**	0	0	0	**0**
F6 - Check the Fresnel zone clearance and the antenna height	**0**	0	0	0	**0.067**
Number of instances	**3250**	953	914	1093	**975**

3.5 Synthetic Minority Oversampling Technique

Unbalanced data sets can lead to suboptimal performance of classifiers [56], particularly DTs [57, 58, 59]. Minority classes of unbalanced data sets might be identified by classification algorithms as noise. However, in some contexts, it is particularly important to classify the minority or abnormal class, because it can represent the interesting behavior that users are trying to identify (for example, in fault detection). The misclassification of these instances represent, in some contexts, high costs (identifying a fault as normal behavior or, in the case of

this thesis, classifying a fault as another, leading to a wrong fix). It is possible to undersample the majority class, oversample the minority classes, or choose a hybrid approach. In the case of undersampling (or the hybrid approach), part of the information is excluded from the problem, which could mean to exclude valuable information.

The SMOTE technique was proposed to substitute the oversampling of data by replication, which caused overfitting and did not give additional information to classifiers. In SMOTE, new instances are generated by interpolation. By default, 5 neighbors of one instance are considered, one of the neighbors is chosen, and the difference between the feature vector of the two samples is considered. Then, the difference is multiplied by a random number between 0 and 1, and this is the feature vector of the new instance that is added to the data set.

In a noisy data set, undersampling the data can decrease the performance, because it removes samples with valuable information, and oversampling can increase it, but it also depends on the percentage of instances removed or generated [60].

The SMOTE technique was applied to balance classes of data sets provided for this thesis, and classification results were compared with balanced and unbalanced sets.

3.6 Principal components analysis

PCA [61, 62, 63] can be applied before ML classification algorithms. It is usually applied to reduce the complexity of interpreting data sets in multidimensional contexts. Principal components correspond to new variables as linear combinations of initial variables of the data set, created with eigenvectors of the covariance or correlation matrix of the original data. The information contained initially is statistically preserved by the variance of components. The variance threshold to select the number of components to work with is not defined, and authors suggest, for example, more than 70%, or values between 90% and 99% [64, 65]. In this thesis, the threshold was equal to 90%, as in [66]. PCA can also provide a simpler visualization of the data. Usually, the first two (which, by definition, are the pair that best preserve the variance) or three principal components are explored to the visualization. Although in this thesis it is presented the projection of the data in two first components, the other components were also

analyzed to understand which is the pair of components that represents better the groups of data in terms of each class. For this purpose, biplots of each pair of components were observed, and the Euclidean distance was applied. It is described in Section 4.1.

A centered PCA was applied with numerical variables normalized since units and the order of magnitude are the same for all numerical variables.

3.7 Machine learning algorithms

DTs, KNN, and SVM were the algorithms chosen to perform the classification. Below, a brief description of each one and respective parameters are presented, as a justification for the choice of the algorithms.

DTs [67, 68, 69] (as KNN and SVM) are non-parametric algorithms, where any distribution about the data is assumed. There are different tree algorithms, being that the one performed for this thesis is Classification And Regression Trees (CART). One of the DTs' advantages is that they are built-in *white box* models, meaning that decisions are easily explained by the decision Boolean rules. So, DTs are somehow more interpretable. This type of construction is interesting to compare variables that are considered in nodes (variables considered relevant by the tree), and the ones that might be considered more important regarding the application context (as it will be presented, DTs have left some variables out). These are the reasons that have motivated the choice of DTs, besides their application on the networking context, as shown in Chapter 2.

DTs are constituted by root, internal, and leaf nodes. A leaf is where the decision rule ends, and a class is attributed. An internal node splits into another based on a threshold relative to one of the numerical input variables. CART constructs binary trees based on numerical variables and thresholds that originate the largest information gain at each node. Information gain represents the quantity of information achieved in a variable from the observation of another one. One pertinent question could be which are the more relevant variables for the tree to consider in nodes, and the information gain is the decisive metric, where variables with higher information gain are preferred.

One crucial aspect is the size (and, consequently, the complexity) of the tree, regarding its

depth and the number of leaves per node. It is controlled by the pruning step, performed after the growing step. The problem of having a big and very complex tree is that it might overfit, because it probably has captured too many details of the training data, and will achieve a lower performance in the test. Pruning the tree could even avoid noisy information to be captured. Scikit-learn algorithm [69] performs the minimal cost-complexity pruning, which finds the subtree that minimizes a complexity measure (that depends on the criterion selected to measure the quality of the splits). Then, it is defined as an effective criterion. The non-terminal node with the lower effective criterion is the weakest link, so it is pruned. The pruning ends when the minimal effective criterion of the pruned tree is greater than the α parameter, which represents the complexity parameter of the pruning algorithm.

The classification was performed with GS 10-CV (for DTs, KNN, and SVM). GS finds the optimal parameters of each algorithm. Optimized parameters for DTs [70] were:

- The criterion to measure the quality of a split: *Gini* and *entropy*;
- The strategy to chose the split at each node: *best* and *random*;
- Maximum depth of the tree: range between *2* and *14*;
- Maximum leaf nodes: range between *2* and *9*;
- The complexity parameter for the pruning step: range between *0* and *0.035*, with steps of 0.005. This range was chosen based on the scikit-learn example [71].

Parameters left by default were the minimum number of instances required to split an internal node, the minimum number of instances required to be at a leaf node, the minimum weighted fraction of total of weights required to be at a leaf node, the number of features to consider when looking for the best split, the minimum impurity decrease to split a node, and class weights.

The choice of KNN is justified by being a non-parametric algorithm, being interpretable in the sense that its functioning is understandable, and by works presented in Chapter 2.

To classify each instance of the data set, KNN [72, 73, 74, 75] computes the distance between each instance and its k nearest neighbors. The classification decision is taken based on the majority class among the k neighbors.

Parameters optimized for KNN [76] were:

- The number of neighbors: range between *2* and *9*. Note that the class that is less represented in DS1 has 9 instances, so choosing a higher value for k did not seem to make sense;
- The function to weight data points according to its distance to the prediction instance: consider *higher weights for closer points* of the instance to classify or *equal weights for all points*;
- The distance metric: *Euclidean, Chebyshev, Manhattan, Mahalanobis,* or Minkowiski with p in the range 3 to 5.

Parameters left by default were the algorithm to compute nearest neighbors and leaf size. The algorithm, by default, is *auto*, which means that it finds the best one regarding the training data. Leaf size is one parameter of some algorithms.

The choice of k is very sensitive, and it could be affected by noisy points. A high value could introduce noise in calculations because the boundary between classes gets less well defined. A low value is less inclusive and could make the algorithm overfits.

SVM were chosen because they constitute a non-parametric algorithm, they have the advantage of being powerful in the sense that they adapt to different types of data, and regarding its importance in the networking context as shown in Chapter 2.

In SVM [77, 78, 79, 80, 81, 82], the goal is to find the hyper-plane that maximize the distance between points of different classes. The point of each class closest to the hyper-plane is called support vector. The task of the search for this unique hyper-plane is done with the kernel function (first layer), and a linear function (second layer).

The parameter being optimized for SVM was the kernel function: *linear, radial basis function, sigmoid* or *polynomial* with degree equal to 2 or 3.

The shrinking option was set as true because it can reduce the training time. The regularization parameter, the kernel coefficient for radial basis function, polynomial, and sigmoid kernels (gamma), the independent term in polynomial and sigmoid kernel functions, the tolerance for stopping criterion, the cache size of the kernel, the parameter to weight the classes, verbose, the limit of iterations, the classification approach applied, and break ties were parameters left as default [83].

The classification approach is one-versus-rest, meaning that the number of trained classifiers for each training set is equal to the number of classes. Each classifier is trained to distinguish between one class and the others [84, 85]. It happens because, originally, SVM were designed to binary classification problems, not multiclass problems. The one-versus-one approach is slower because it considers all pairs of classes [86].

This chapter started with a brief presentation of the WiBACK system, to provide the reader with some knowledge about the use case of this dissertation. The data sets collection process was described, as the data sets, and the criteria for the numerical variable selection. SMOTE, PCA, and ML algorithms applied were introduced.

Chapter 4.
Results and discussion

In this chapter, all results and discussion are presented. First, an explanation of the Euclidean distance calculations made with the PCA data is presented, as graphical representations of principal components. Classification results are presented, regarding weighted F1-score, balanced accuracy, and accuracy. Confusion matrices are also analyzed to discuss classes with more misclassifications between errors, causes, and fixes. As algorithms were trained with different data sets, the chapter is divided considering the training set.

4.1 Train with DS1

Before the application of ML algorithms, a PCA was applied to the training data DS1. PCA was considered, not just to reduce the space dimensionality and test algorithms with principal components as input, but also as an exploratory study. The goal was to conclude which were the principal components that best separate the data by its classes of errors, causes, and fixes.

The first 10 principal components were chosen to represent the data because they explained more than 90% of the data variance. *Overrun* and *TxPower_down* were the variables with greater influence, in absolute value, in principal components. They had a great influence on components 7 and 5, respectively.

In Figure 4.1 to Figure 4.6, one can see that, as components increase, data points get more scattered, and consequently, class groups get less defined. To understand it beyond the visual perception, the Euclidean distance was calculated between data points. This study was performed to conclude which pairs of components represent better the data in terms of its grouping into each class, and separation between different classes. The Euclidean distance was

calculated:

- Between all points of each class, in each component;
- Between all points of each class, in all pairs of components;
- Between different classes, in each component;
- Between different classes, in all pairs of components;
- Between all points of the data set, in each component;
- Between all points of the data set, in all pairs of components.

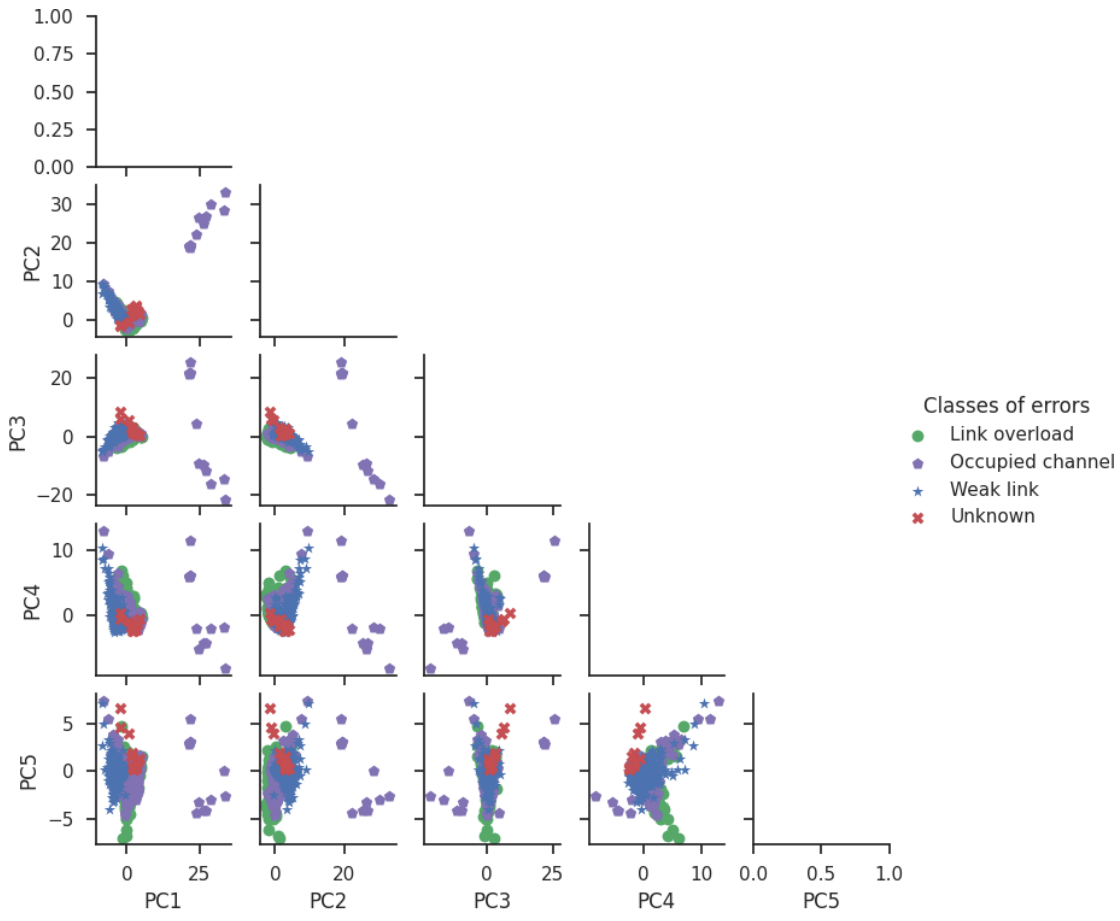


Figure 4.1: Data projection into first five principal components with classes of errors colored.

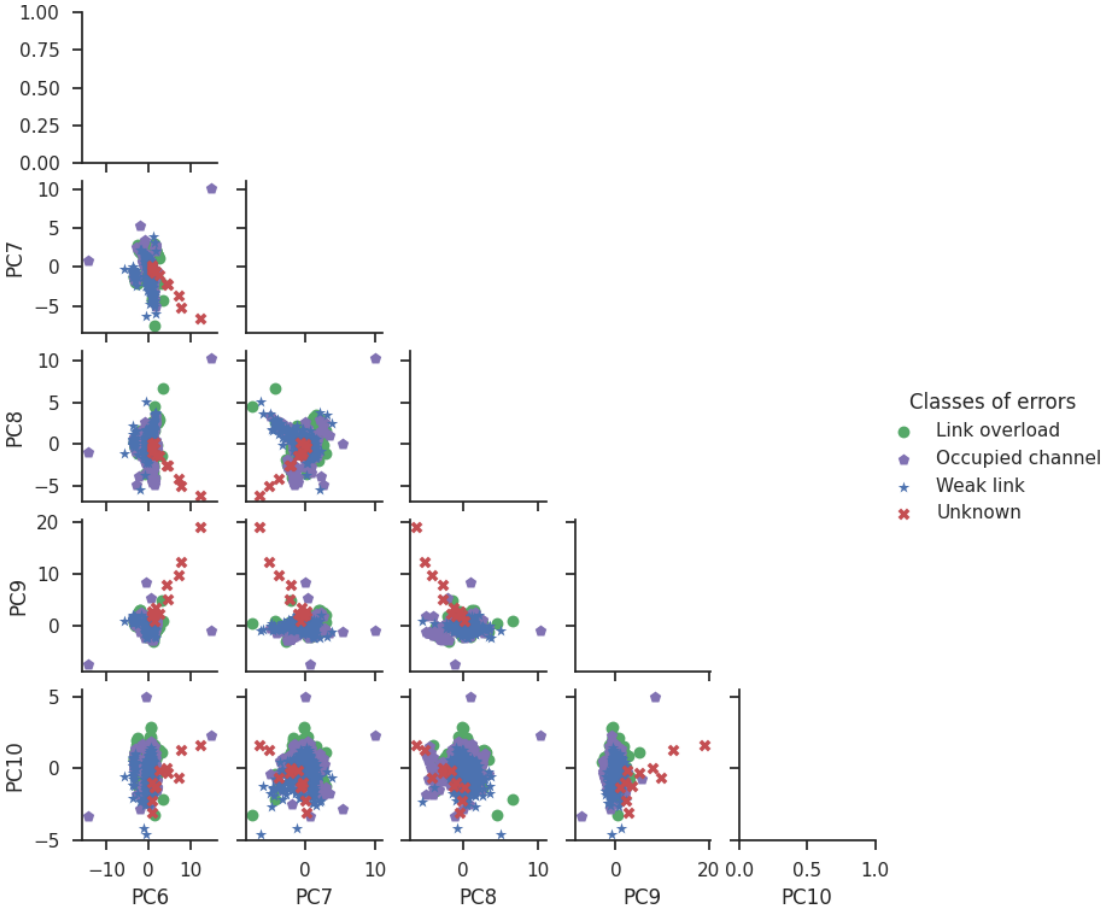


Figure 4.2: Data projection into last five principal components with classes of errors colored.

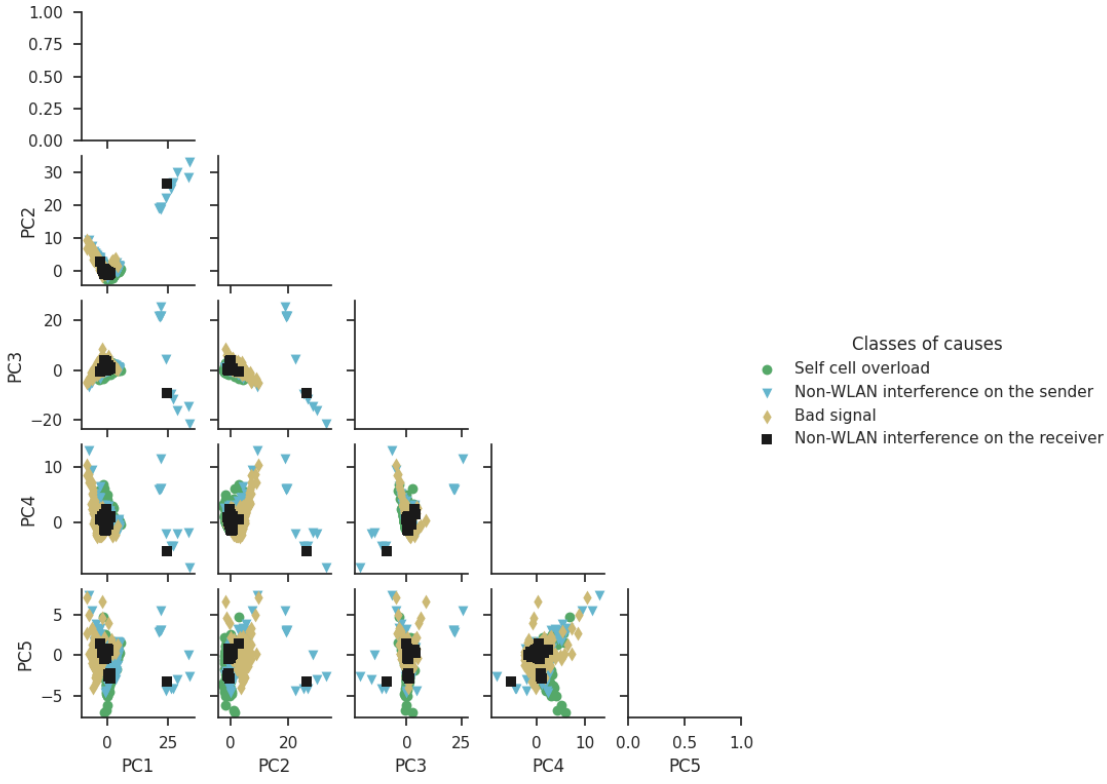


Figure 4.3: Data projection into first five principal components with classes of causes colored.

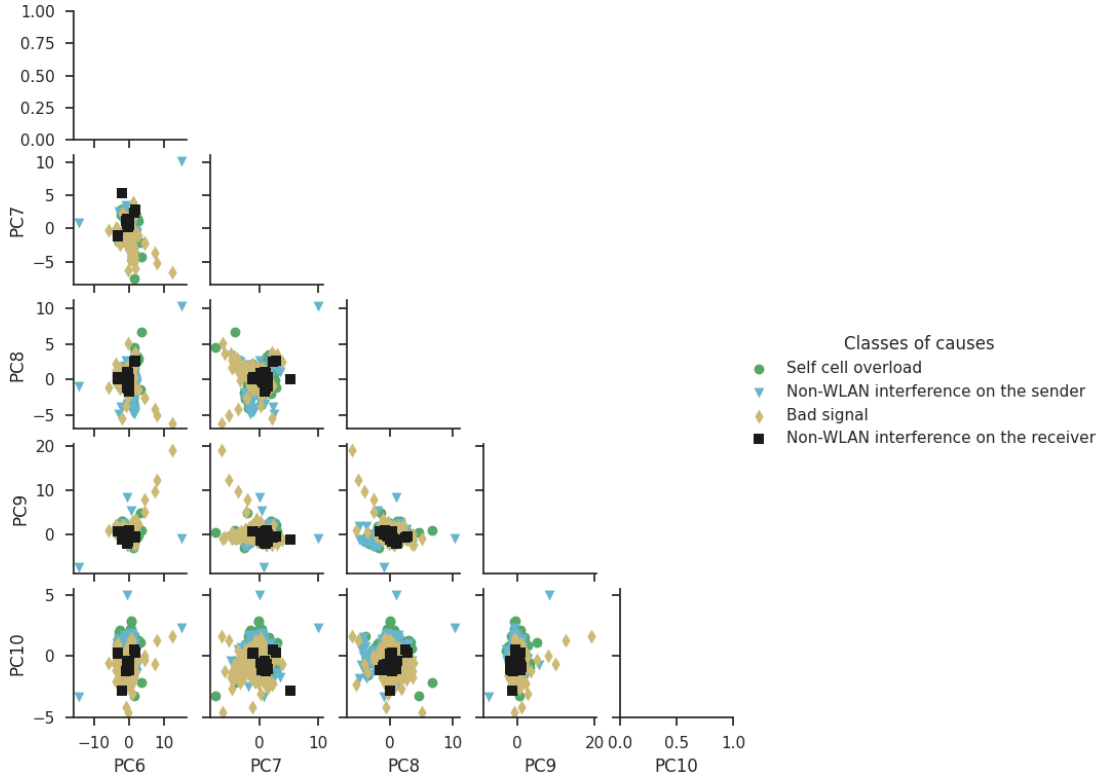


Figure 4.4: Data projection into last five principal components with classes of causes colored.

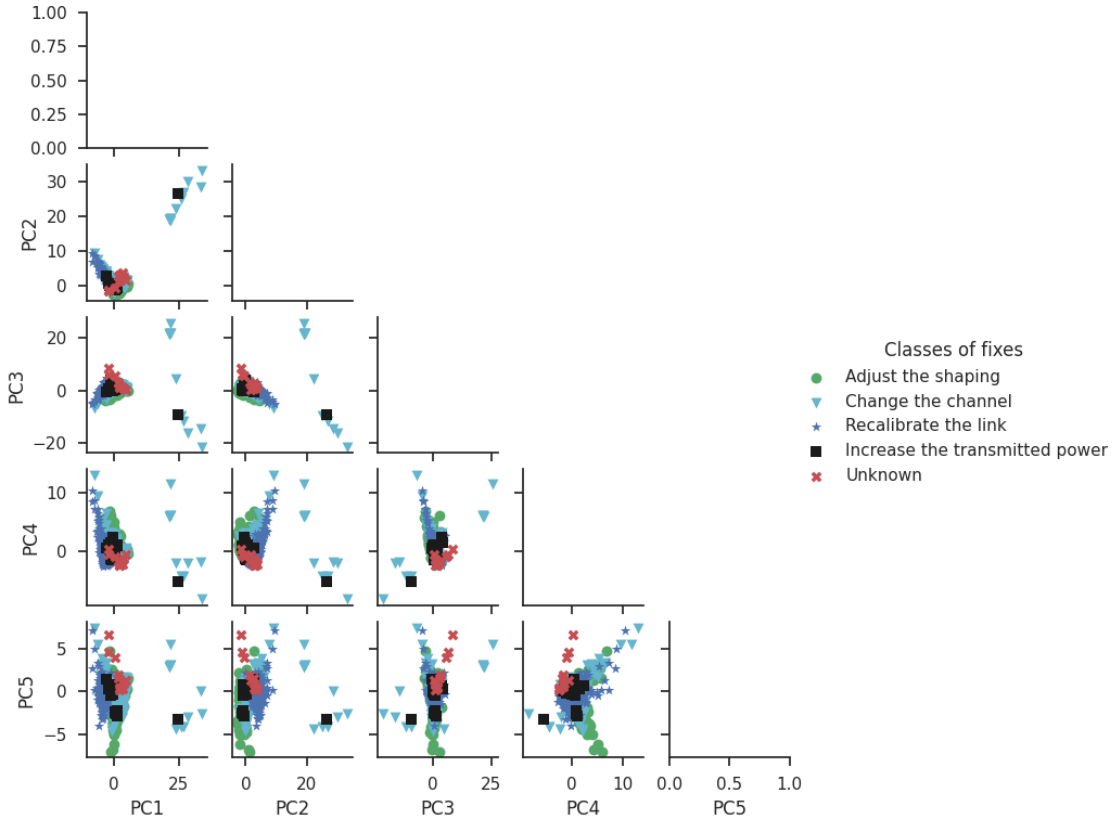


Figure 4.5: Data projection into first five principal components with classes of fixes colored.

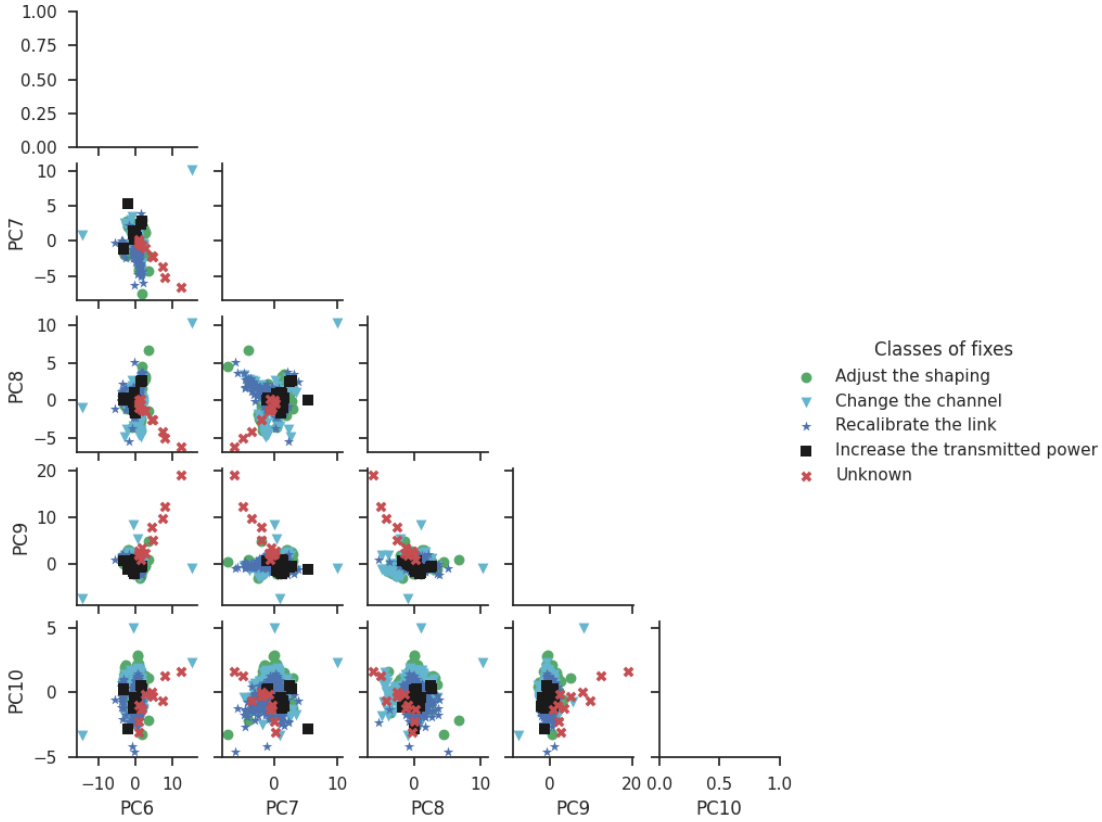


Figure 4.6: Data projection into last five principal components with classes of fixes colored.

Figure 4.7 to Figure 4.9 show the distance evolution as we move in pairs of components, regarding classes of errors, causes, and fixes, respectively. As components increase, the distance between points increases, which seems to be compatible with the visual notion that points distributions get somehow confusing in the last pairs of components. Note that the distance was computed in all pairs of components, but Figure 4.1 to Figure 4.6 show only pairs of consecutive components. The *unknown* class of errors and fixes, that correspond only to 9 instances of the data set, achieved the highest distances, and it represents curves (both in errors and fixes) with more variations. Besides, Figure 4.7 and Figure 4.9 show that from the twenty-eighth pair of components, this distance started increasing more. It seems that the representation of the class *unknown*, from the twenty-eighth pair of components, got more dispersed than the representation of other classes.

These distances were computed in the pairs of components to be coherent with the visual data projections. However, the distance between points of each class was also computed for each component separately. It was verified that the distance between points of each class increases as we move in the components.

Concluding, besides the first two components represent (by definition) better the data, and represent more variance than any other pair of components, it seems that they also better represent the data visually in the classes. The Euclidean distance was used to support this graphical suggestion.

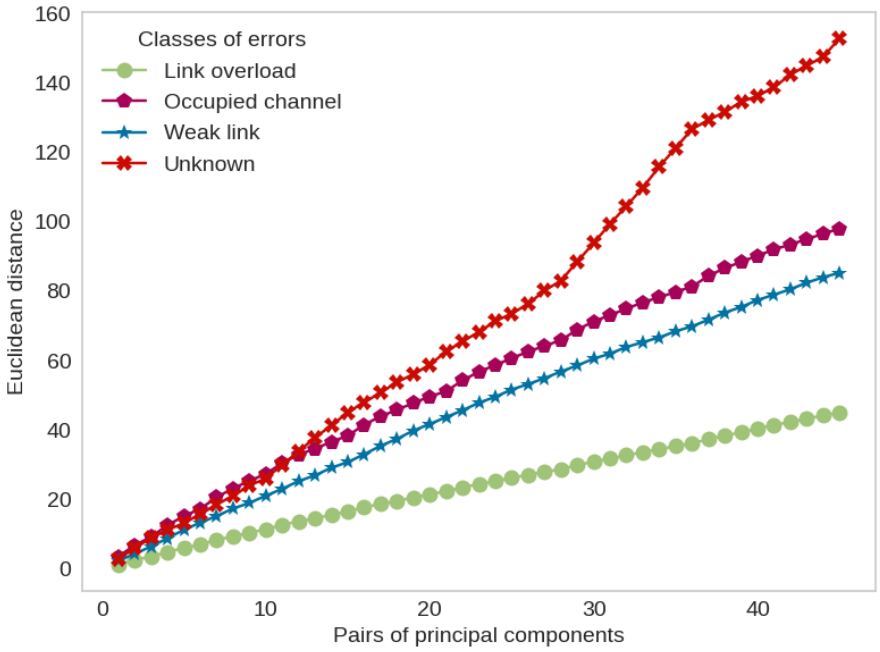


Figure 4.7: Euclidean distance between the points of each class of errors in the pairs of principal components. For i in {1,2,3,4,5,6,7,8,9,10}, the distance was computed between components C_i and C_j, for $j > i$ and j in {1,2,3,4,5,6,7,8,9,10}.

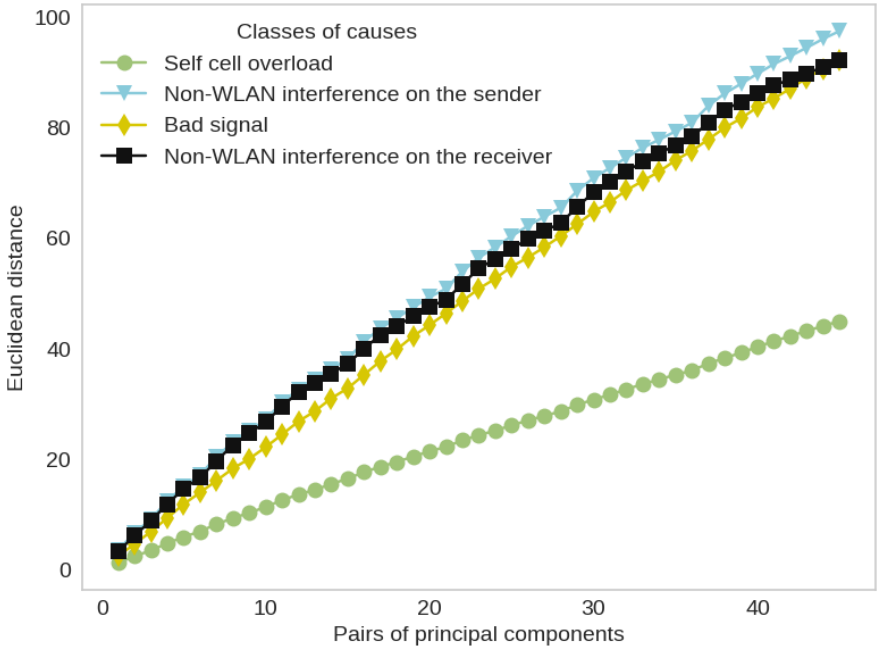


Figure 4.8: Euclidean distance between the points of each class of causes in the pairs of principal components. For i in {1,2,3,4,5,6,7,8,9,10}, the distance was computed between components C_i and C_j, for $j > i$ and j in {1,2,3,4,5,6,7,8,9,10}.

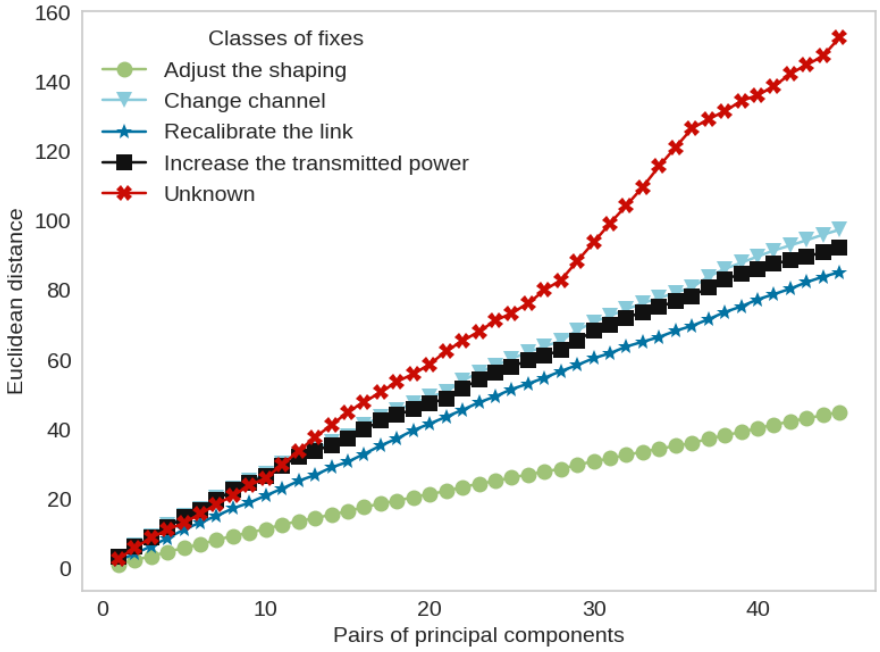


Figure 4.9: Euclidean distance between the points of each class of fixes in the pairs of principal components. For i in {1,2,3,4,5,6,7,8,9,10}, the distance was computed between components C_i and C_j, for $j > i$ and j in {1,2,3,4,5,6,7,8,9,10}.

Euclidean distance was also computed between different classes, in each component. It was done by considering the mean of each class, in each component, and the difference between classes. The same was done in all pairs of components, by considering the mean of each class in each pair of components, and then the difference between classes. These calculations were made to confirm if as components increase, the separation between different classes would be also verified, or if the data representation would just get more dispersed. However, these values were inconclusive since there was not a pattern.

Also to conclude if, in general, independently of classes, the data representation got more dispersed, the Euclidean distance was computed between all points, in each component and in all pairs of components. Results showed that points were more distant as components increase.

After this analysis, DTs, KNN, and SVM were trained with DS1 and tested with DS2, DS3, and DS4. DS3 is the DS2 without the cause *foreign WLAN interference*, which is not in the training set.

DTs, KNN, and SVM were trained with unbalanced, balanced, and PCA data sets to classify errors, causes, and solutions.

The metric chosen to optimize the GS was a weighted F1-score. In the case of balanced data

sets, the original data was balanced and given as input to GS, but tests were performed with unbalanced (original) data [87]. This means that, in these cases, the data distribution from training with DS1 to tests with DS2, DS3, and DS4 changed considerably.

F1-score is a weighted average of the precision and recall. Weighted F1-score by scikit-learn, in a multiclass problem, is the average of the F1-score of each class with weighting [88]. Accuracy [89] gives the number of correct predictions. Balanced accuracy [90] is the average recall of each class. The three metrics can present values between 0 and 1, being 1 the best value and 0 the worse [91, 92].

Two criteria were used to evaluate algorithms performance. The first one was the direct evaluation of weighted F1-score, accuracy, and balanced accuracy values. The second one was the difference between weighted F1-score from GS and weighted F1-score from each test set (DS2, DS3, and DS4). With the second criterion, the objective was to understand if algorithms were able to generalize their learning to new data sets since GS uses one data set that is split into train and test (several times). This means that the weighted F1-score from GS may not reflect the generalization capability of algorithms (depending on the heterogeneity of the data set provided for GS).

Figure 4.10 to Figure 4.12 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of DTs regarding different test sets (DS2, DS3, and DS4) for errors classification. Figure 4.10 shows that the weighted F1-score from GS was higher with the balanced data, but the difference between unbalanced and PCA data was acceptable. It is more relevant to analyze the differences between weighted F1-score from GS and the remaining test sets. Differences were acceptable with the unbalanced data, contrarily to balanced and PCA data. Regarding accuracy and balanced accuracy, both were higher with unbalanced data, with two exceptions. The first one was DS4, where the three metrics were equal for balanced and unbalanced data. The second exception was DS3, where the balanced accuracy was higher with the PCA data.

Figure 4.13 to Figure 4.15 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of DTs regarding different test sets (DS2, DS3, and DS4) for causes classification. Figure 4.13 shows that the weighted F1-score from GS did not present

unacceptable differences between unbalanced, balanced, and PCA data. The weighted F1-scores of unbalanced DS2, DS3, and DS4 were similar to the value from GS, contrarily to what has happened with balanced and PCA data. Regarding accuracy and balanced accuracy, values were higher with unbalanced data from DS2 and DS4. With DS3, the balanced accuracy was higher for PCA data.

Figure 4.16 to Figure 4.18 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of DTs regarding different test sets (DS2, DS3, and DS4) for fixes classification. Figure 4.16 shows that the weighted F1-score from GS did not present unacceptable differences between unbalanced, balanced, and PCA data. One more time, weighted F1-score, accuracy, and balanced accuracy were higher with unbalanced data. There was one exception because balanced accuracy from DS3 was higher with PCA data.

Regarding a comparison between classification performance for errors, causes, and solutions, algorithms did not present unacceptable differences.

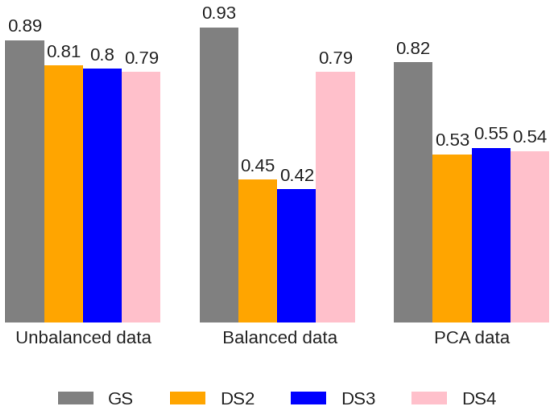


Figure 4.10: Weighted F1-score of DTs trained to classify errors.

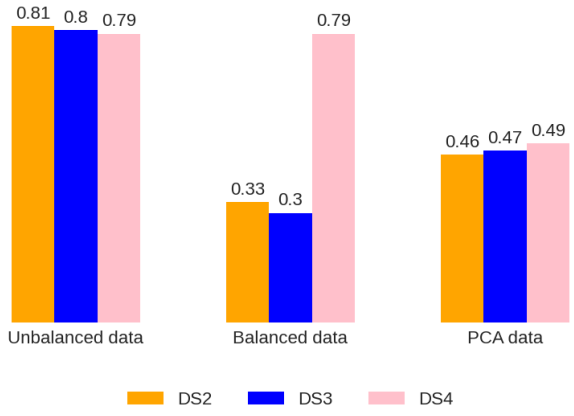


Figure 4.11: Accuracy of DTs trained to classify errors.

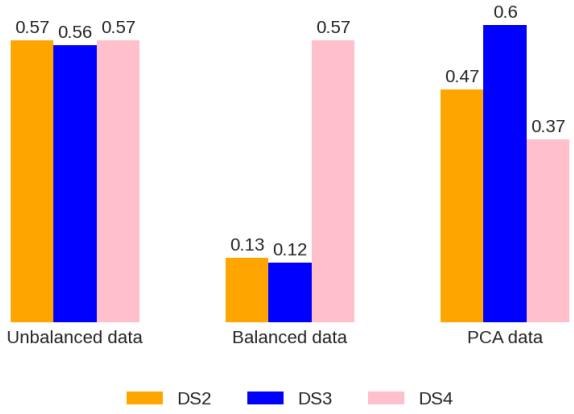


Figure 4.12: Balanced accuracy of DTs trained to classify errors.

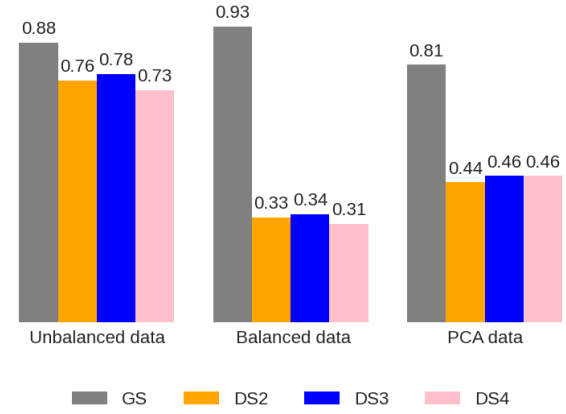


Figure 4.13: Weighted F1-score of DTs trained to classify causes.

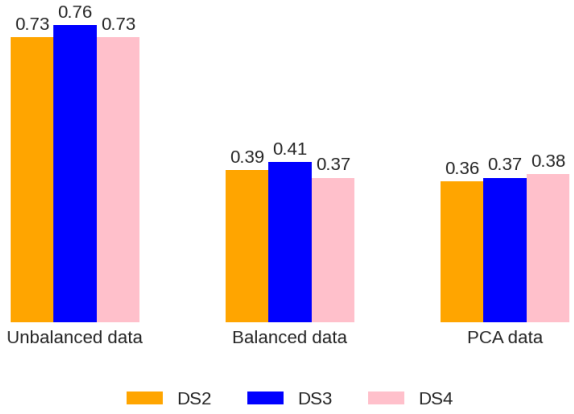


Figure 4.14: Accuracy of DTs trained to classify causes.

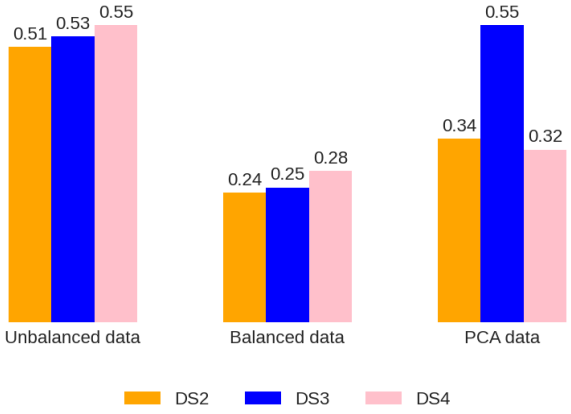


Figure 4.15: Balanced accuracy of DTs trained to classify causes.

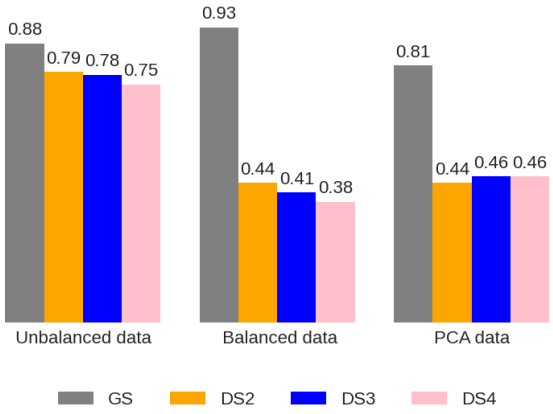


Figure 4.16: Weighted F1-score of DTs trained to classify fixes.

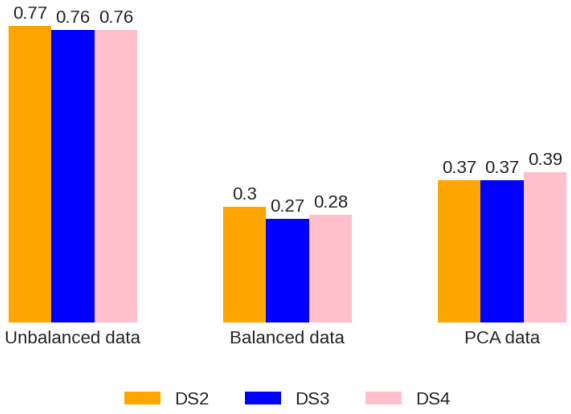


Figure 4.17: Accuracy of DTs trained to classify fixes.

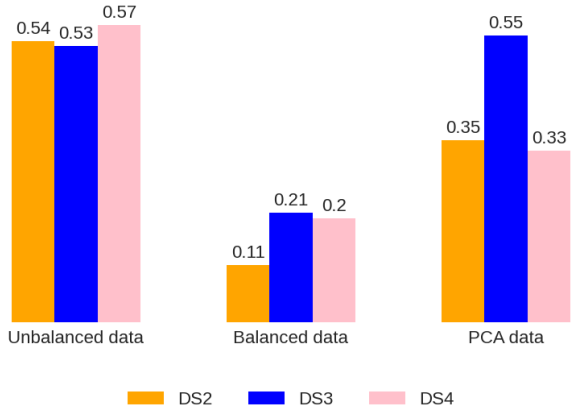


Figure 4.18: Balanced accuracy of DTs trained to classify fixes.

Figure 4.19 to Figure 4.21 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of KNN regarding different test sets (DS2, DS3, and DS4) for errors classification. Figure 4.19 shows that the weighted F1-score from GS did not present unacceptable differences between unbalanced, balanced, and PCA data. However, the weighted F1-score presented inappropriate differences between GS and the remaining test sets. Weighted F1-score and accuracy were very similar for all data from DS2, DS3, and DS4, contrarily to balanced accuracy, which showed some variations between data sets. However, the three metrics presented very low values (between 0.21 and 0.6) for the classification of DS2, DS3, and DS4.

Figure 4.22 to Figure 4.24 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of KNN regarding different test sets (DS2, DS3, and DS4) for causes classification. Conclusions of KNN classifiers for causes are the same that were presented for KNN classifiers of errors.

Figure 4.25 to Figure 4.27 show the weighted F1-score from GS and the weighted F1-score, accuracy, and balanced accuracy of KNN regarding different test sets (DS2, DS3, and DS4) for fixes classification. The conclusions of KNN classifiers for fixes are the same that were presented for KNN classifiers of errors and causes. There were no unacceptable differences in the classification performance of errors, causes, and fixes with KNN.

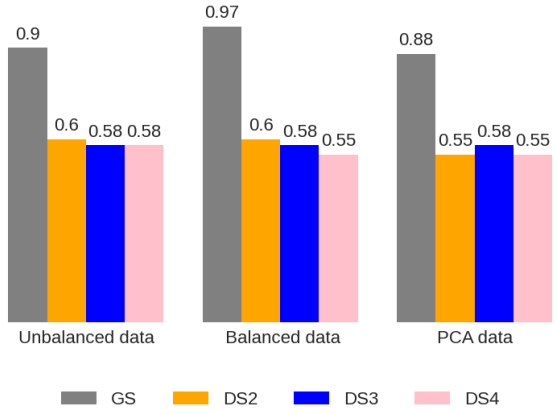


Figure 4.19: Weighted F1-score of KNN trained to classify errors.

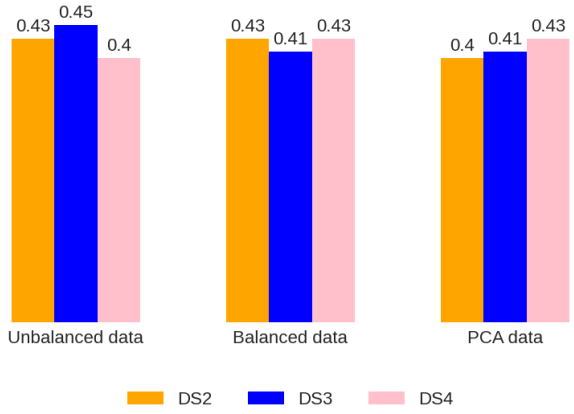


Figure 4.20: Accuracy of KNN trained to classify errors.

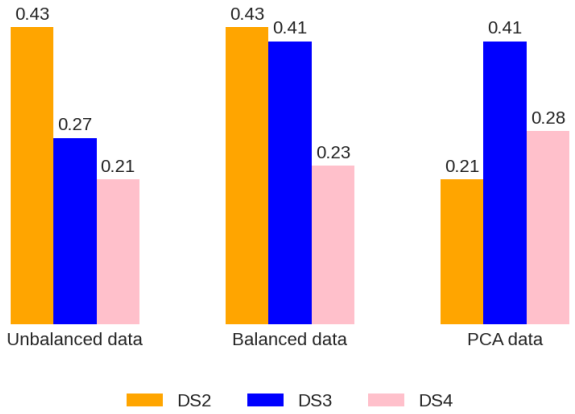


Figure 4.21: Balanced accuracy of KNN trained to classify errors.

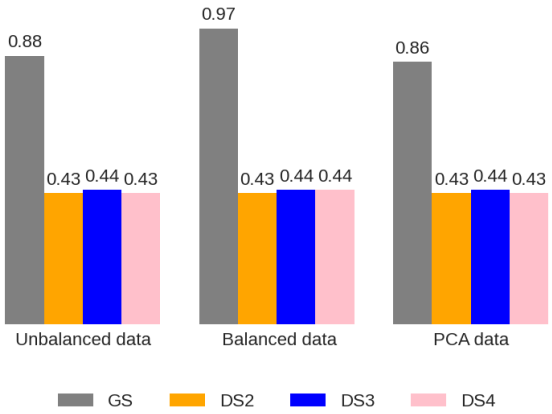


Figure 4.22: Weighted F1-score of KNN trained to classify causes.

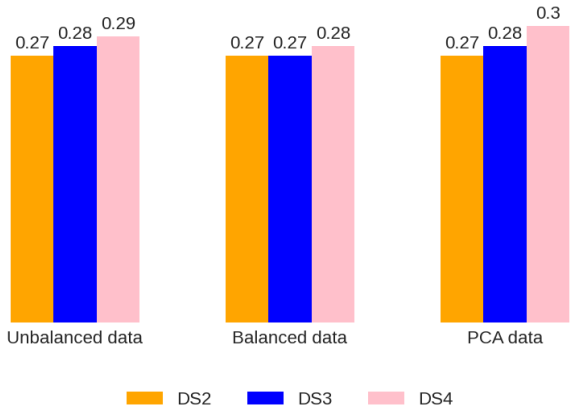


Figure 4.23: Accuracy of KNN trained to classify causes.

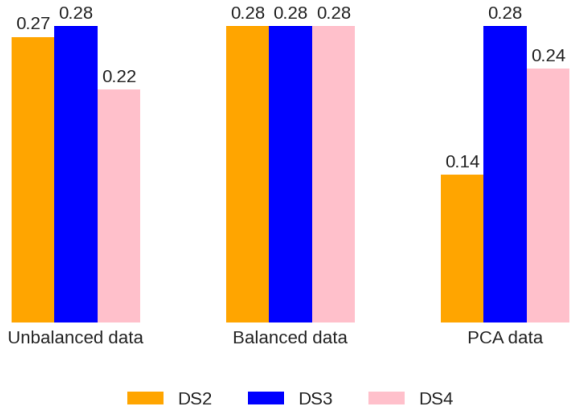


Figure 4.24: Balanced accuracy of KNN trained to classify causes.

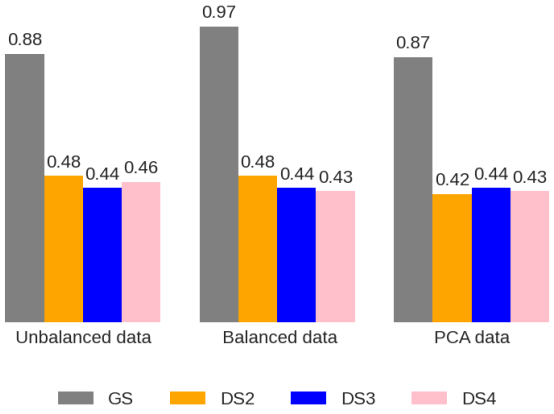


Figure 4.25: Weighted F1-score of KNN trained to classify fixes.

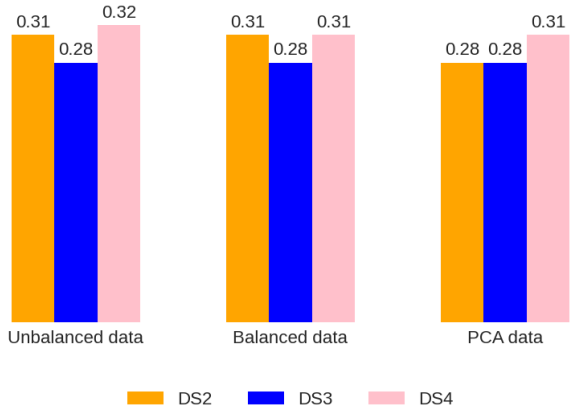


Figure 4.26: Accuracy of KNN trained to classify fixes.

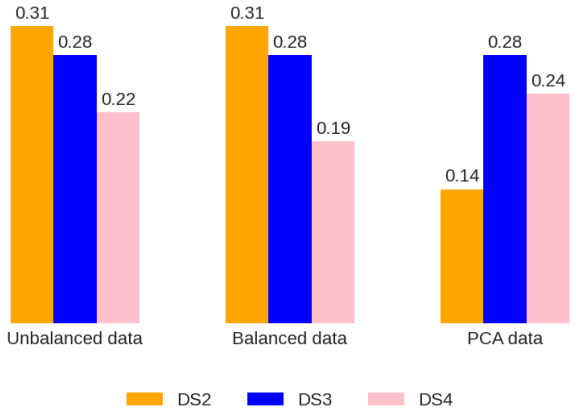


Figure 4.27: Balanced accuracy of KNN trained to classify fixes.

Figure 4.28 to Figure 4.30 show the weighted F1-score from GS and the weighted F1-score, accuracy, and balanced accuracy of SVM regarding different test sets (DS2, DS3, and DS4) for errors classification. Figure 4.28 shows that the weighted F1-score from GS and the weighted F1-score of unbalanced data sets were equal to 0.96, excepting with the unbalanced DS4, where it was equal to 0.93 (but this difference is acceptable). Figure 4.29 shows that the accuracy of unbalanced data sets was between 0.94 and 0.96. This appears to be an appropriate performance, but Figure 4.30 shows lower values of the balanced accuracy. Results with balanced and PCA data were not as promising.

Figure 4.31 to Figure 4.33 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of SVM regarding different test sets (DS2, DS3, and DS4) for causes classification. Figure 4.31 shows that the weighted F1-score from GS was high (0.95 and 0.98) and very similar with weighted F1-score from balanced and unbalanced data sets. Figure

4.32 shows adequate values of accuracy (between 0.77 and 0.85) for balanced and unbalanced data sets, but Figure 4.33 shows low values of balanced accuracy for these two data sets (between 0.42 and 0.5). Results for PCA data were lower.

Figure 4.34 to Figure 4.36 show the weighted F1-score from GS, as the weighted F1-score, accuracy, and balanced accuracy of SVM regarding different test sets (DS2, DS3, and DS4) for fixes classification. Figure 4.34 shows that the weighted F1-score with unbalanced data set was high with all data sets (between 0.87 and 0.95), but with balanced and PCA data, the differences from GS were inappropriate. One more time, Figure 4.35 shows that accuracy values were appropriate for the unbalanced data (between 0.81 and 0.84), but Figure 4.36 shows lower values of balanced accuracy.

Regarding a comparison between errors, causes, and fixes classification with SVM, results varied a little bit more than with DTs and KNN, but, in general, the differences were acceptable.

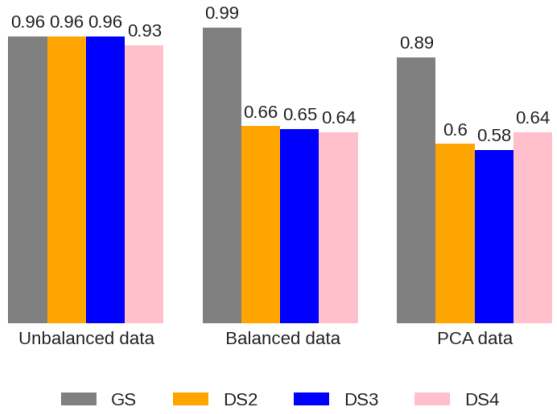


Figure 4.28: Weighted F1-score of SVM trained to classify errors.

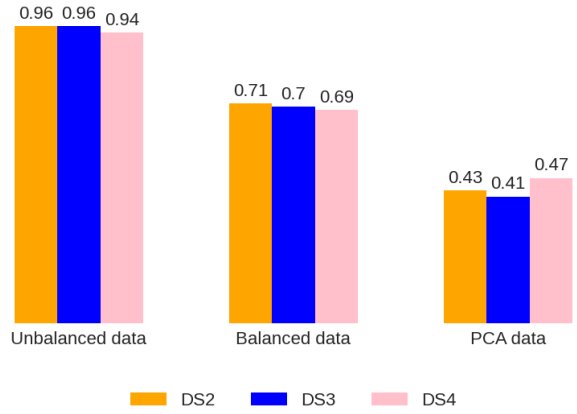


Figure 4.29: Accuracy of SVM trained to classify errors.

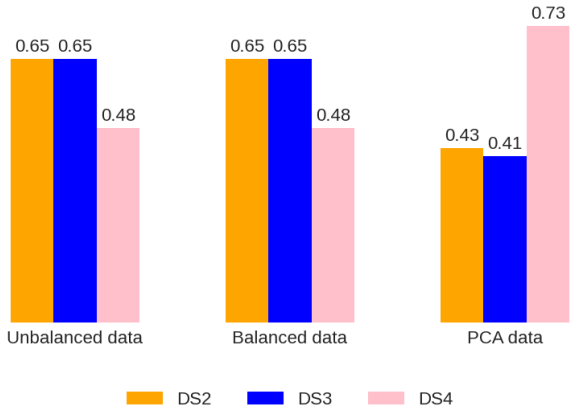


Figure 4.30: Balanced accuracy of SVM trained to classify errors.

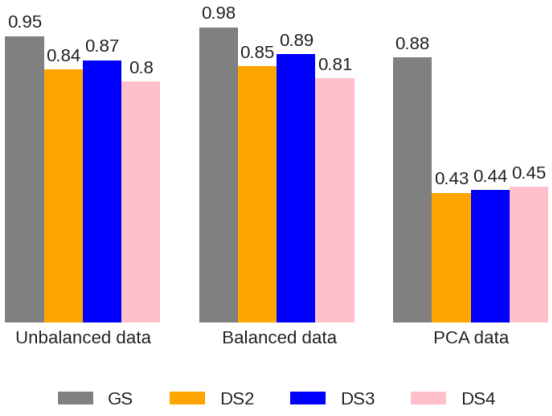


Figure 4.31: Weighted F1-score of SVM trained to classify causes.

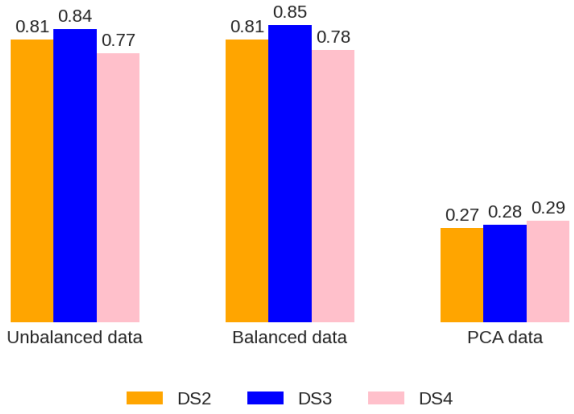


Figure 4.32: Accuracy of SVM trained to classify causes.

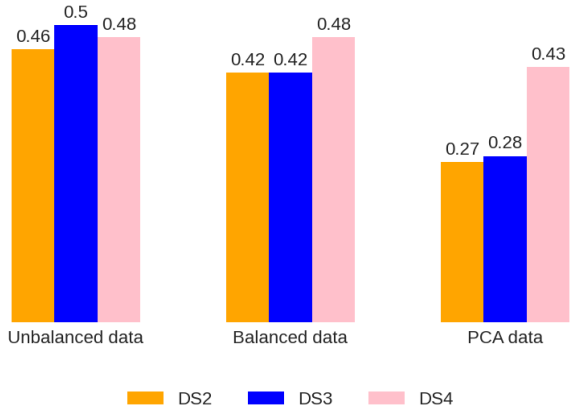


Figure 4.33: Balanced accuracy of SVM trained to classify causes.

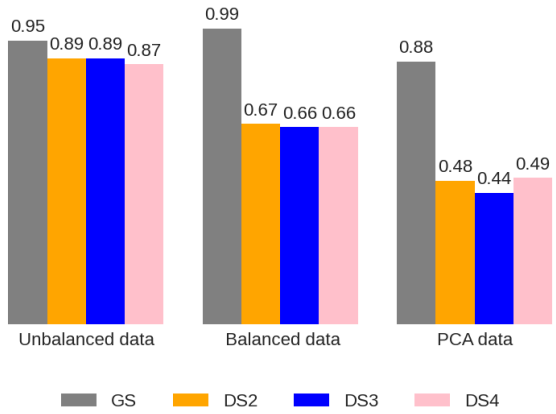


Figure 4.34: Weighted F1-score of SVM trained to classify fixes.

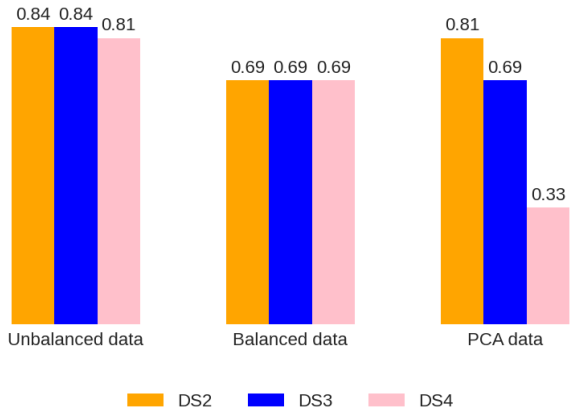


Figure 4.35: Accuracy of SVM trained to classify fixes.

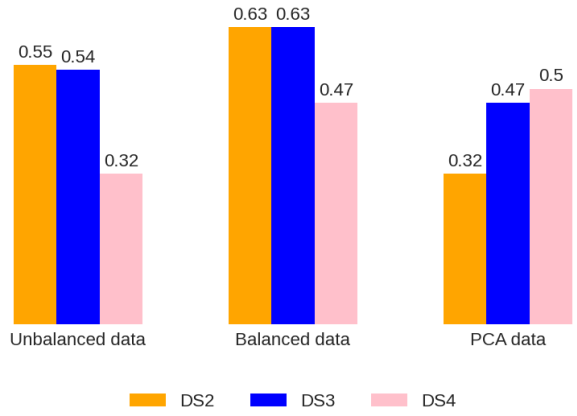


Figure 4.36: Balanced accuracy of SVM trained to classify fixes.

From a general perspective, with DTs, unbalanced data sets presented the best results, and balanced ones to worst. With KNN, there were no unacceptable differences between weighted F1-score and accuracy from unbalanced, balanced, and PCA, with the exception of balanced accuracy results, where the performance changed more with the test set evaluated. With SVM,

results were better with the unbalanced data and worse with PCA.

Considering DS2 and DS3, it could be somehow expected that the results with DS2 (at least for causes classifiers) were worse than DS3 because the former has a class that is not in the training set. Weighted F1-score and accuracy, in causes classification, were always higher in DS3. However, results were very similar, probably because the amount of instances of the new class represents only 4.1% of DS2. Regarding errors and fixes classifiers, weighted F1-score and accuracy were very similar between DS2 and DS3, which was somehow expected because the class frequencies of errors and fixes are almost equal.

Regarding balanced accuracy in causes classification with PCA, conclusions were the same, excepting that differences were higher with this metric.

Regarding DS4, which also includes the cause *foreign WLAN interference*, like DS2, the weighted F1-score and accuracy were very similar with DS2 and DS4, excepting the classification of errors with DTs, in which the balanced data for DS4 stood out.

Balanced accuracy results were a little bit more heterogeneous between DS2 and DS4.

Nonetheless, the performance of SVM was better and without unacceptable differences regarding GS weighted F1-score and the unbalanced data. Hence, regarding the data presented, SVM was an algorithm with better capacity to generalize the classification.

KNN has a very sensitive parameter, k. It is fixed from the training data, hence if the distribution in test sets is different, the classification will probably have wrong predictions. This is probably what happened with DS2, DS2, and DS4 regarding the train with DS1. KNN presented a higher weighted F1-score from GS than DTs, but results decreased in tests, probably because k was not the best for the test data.

The data could have noise, provenience from classes, or numerical variables [93], which may affect results. This also probably affected k. Noise instances were excluded from data sets, but it is possible that it still exists since some variables are not 100% accurate in the wireless context.

Attempting to improve results, algorithms were trained with new parameters. In DTs, the range of the complexity parameter for the pruning step became between 0 and 0.075 with steps of 0.005. In KNN, the range for the number of neighbors became from 2 to 19, and the range for parameter p of the Minkowski distance became between 3 to 9. In SVM, the range for the

polynomial kernel became from 2 to 9.

Previously, it was said that having k values higher than 9 did not seem to make sense since the smallest class of DS1 had only 9 instances. However, this class is *unknown*, hence it is preferable to have more right predictions in other classes even if the number of wrong predictions rises in this one. However, the results with old and new parameters were the same for all algorithms.

One of the difficulties of this work was to chose numerical variables, regarding the importance of each one in the application context, and that some of them are not 100% accurate. Looking for all the DTs built, *AT_out*, *AT_in*, *Rx_br_down* and *Rx_br_up* were considered in all DTs trained with the unbalanced and the balanced data.

One of the particularities of this work is the correspondence between errors, causes, and fixes, so confusion matrices [94] of classifiers were analyzed. The goal was to understand if that correspondence held or not.

Each cell of the diagonal of each matrix presented in Appendix A represents the number of right predictions for each class. Each one of the remaining cells represents the number of wrong predictions, being that columns represent predicted classes and rows represent labeled classes. For simplicity of analysis, abbreviations of Table 3.3 were used.

Table A.1 to Table A.3 show confusion matrices of errors classification, regarding all data sets and algorithms. From a general perspective, DTs and KNN misclassified E1 as E2 with all data sets. With SVM, E2 was classified as E1, E2 as E4, and E1 as E2, with the unbalanced, balanced, and PCA data sets, respectively.

Table A.4 to Table A.6 show confusion matrices (which also contain the class that is only in DS2 and DS4) of causes classification, regarding all data sets and algorithms. From a general perspective, KNN misclassified C1, C4, and C5 as C2, such as SVM with the PCA data set. C4 as C5, C5 as C1, and C5 as C2 were the misclassifications made by SVM with unbalanced and balanced data sets. Also C2 as C5, with the first unbalanced data. With DTs, it got more disperse, and the dispersion was around all classes.

Table A.7 to Table A.9 show confusion matrices of fixes classification, regarding all data sets and algorithms. From a general perspective, F1 was misclassified as F2, excepting with SVM trained with unbalanced and balanced data sets. F4 was also misclassified as F2, excepting with

DT and SVM trained with the balanced data. F4 and F2 were misclassified as F5, and F4 as F1. F2 and F4 were misclassified as F3 with KNN trained with PCA and DT trained with the unbalanced data, respectively.

With DS2, DS3, and DS4, misclassifications, in general, happened in the same classes (excepting C4, that does not exist in DS3).

In general, the correspondence between errors, causes, and fixes was verified in the wrong predictions, but it did not happen always. The correspondence was mostly verified between causes and fixes.

4.1.1 Retrain with DS1

To make the algorithms learn with the new class of causes, *foreign WLAN interference*, and more instances from more periods, the algorithms were retrained joining the different instances of DS2, DS3, and DS4.

Table 4.1 to Table 4.3 show the weighted F1-score from GS provenience from this retraining moment.

Table 4.1: F1-weighted score of DTs when algorithms were retrained with DS1 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	**E**	**C**	**F**
Unbalanced	0.86	0.84	0.85
Balanced	0.91	0.9	0.89
PCA	0.84	0.79	0.78

Table 4.2: F1-weighted score of KNN when algorithms were retrained with DS1 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	**E**	**C**	**F**
Unbalanced	0.89	0.89	0.88
Balanced	0.97	0.98	0.98
PCA	0.88	0.87	0.86

Table 4.3: F1-weighted score of SVM when algorithms were retrained with DS1 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	**E**	**C**	**F**
Unbalanced	0.95	0.93	0.93
Balanced	0.98	0.97	0.97
PCA	0.89	0.88	0.87

These values were much higher when compared not just with tests with DS2, DS3, and DS4, but also with the weighted F1-score from GS of the first train. Coincidentally with results from the first train with GS, SVM showed the highest values, then KNN, and finally DTs. Regarding only these tests with GS, it is not possible to conclude if the algorithms will not overfit, but they should be more prepared to deal with different data.

Regarding numerical variables in nodes of DTs, they were the same that in the first train, excepting *AT_out*, that, in this case, was not considered to classify errors with the balanced data. In PCA, variables with more influence in some components were *AT_down*, *Overrun*, and *MCS_down*.

4.2 Train with DS5

After retraining algorithms with DS1, algorithms were retrained with the last data set provided. As aforementioned, data sets suffered changes over time. So, the results of testing algorithms trained with DS5 are more significant in the problem context, because this data set is the one with final updates and considerations. It would be more conclusive if there was a new data set to perform tests.

As presented in Table 3.2, different sets of numerical variables were considered (DS5 - 1, DS5 - 2, DS5 - 3).

Table 4.4 to Table 4.6 show the weighted F1-score from GS provenience from training with DS5 - 1.

Table 4.4: F1-weighted score of DTs trained with DS5 - 1 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.99	0.99	0.99
Balanced	1	1	1
PCA	0.91	0.79	0.89

Table 4.5: F1-weighted score of KNN trained with DS5 - 1 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.92	0.9	0.91
Balanced	0.98	0.9	0.99
PCA	0.91	0.87	0.91

Table 4.6: F1-weighted score of SVM trained with DS5 - 1 to classify the errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.93	0.92	0.92
Balanced	0.97	0.99	0.98
PCA	0.90	0.90	0.91

Table 4.7 to Table 4.9 show the weighted F1-score from GS provenience from training with DS5 - 2.

Table 4.7: F1-weighted score of DTs trained with DS5 - 2 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.91	0.92	0.91
Balanced	0.99	1	1
PCA	0.88	0.89	0.88

Table 4.8: F1-weighted score of KNN trained with DS5 - 2 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.92	0.89	0.9
Balanced	0.98	0.99	0.99
PCA	0.91	0.92	0.93

Table 4.9: F1-weighted score of SVM trained with DS5 - 2 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.93	0.9	0.92
Balanced	0.98	0.99	0.98
PCA	0.92	0.93	0.94

Table 4.10 to Table 4.12 show the weighted F1-score from GS provenience from training with DS5 - 3.

Table 4.10: F1-weighted score of DTs trained with DS5 - 3 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.97	0.99	0.91
Balanced	0.98	0.99	0.99
PCA	0.91	0.86	0.88

Table 4.11: F1-weighted score of KNN trained with DS5 - 3 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.92	0.92	0.92
Balanced	0.98	0.99	0.99
PCA	0.91	0.89	0.9

Table 4.12: F1-weighted score of SVM trained with DS5 - 3 to classify errors (E), causes (C), and fixes (F) with unbalanced, balanced, and PCA data sets.

Data set	E	C	F
Unbalanced	0.93	0.9	0.92
Balanced	0.97	0.93	0.94
PCA	0.9	0.89	0.91

Contrarily to what had been concluded previously, with these three new data sets, balanced data showed better results than the unbalanced. PCA continued showing the lowest weighted F1-score.

With DS5 - 1, DTs presented the highest values of weighted F1-score, then SVM, and finally KNN. With DS5 - 2, results were quite similar with the three algorithms. With DS5 - 3, DTs

were better than SVM and KNN with unbalanced data, and SVM were worse than DTs and KNN with balanced data. However, the differences were acceptable.

As mentioned when algorithms were retrained with DS1, more data is needed, to perform more tests and conclude if one of the algorithms stands out.

Regarding DTs trained with DS5 - 1, *Cap_utilized* and *PEI_down* were always considered with the unbalanced an the balanced data. With DS5 - 2, *AT_out*, *SQ_down* and *BER_down*. With DS5 - 3, *Cap_utilized* and *BER_down*.

In PCA, variables of DS5 - 1 with more influence in some components were *Total_Airtime*, *Overrun*, and *Rx_br_up*. Variables of DS5 - 2 with more influence were *AT_in* and *Rx_br_up*. Variables of DS5 - 3 with more influence were *RED* and *Cap_utilized*.

Looking to DTs and PCA, it is clear that some variables stood out. Besides, DTs left out much variables, which could mean that there is no need to train them with so many variables. This conclusion stands for all training sets (DS1, DS5 - 1, DS5 - 2, and DS5 - 3). Regarding results presented in this section, as differences with these three data sets were not inappropriate, the data set chosen would be DS5 - 3. Reasons to chose this set are that it is smaller, reducing computational costs and eventually chances of overfitting.

4.3 Final Remarks

Regarding PCA, the two first principal components represented data classes more clearly. However, PCA did not bring better classification results in terms of algorithms performance (and it is not the main goal with the use of PCA).

Regarding the correspondence between classes of errors, causes, and fixes, in general, this correspondence maintained in the wrong predictions. However, it was maintained mainly between classes of causes and fixes.

Concerning the unbalanced and balanced data sets, it was not conclusive if balancing the data is a procedure to apply with eventually new training sets. With DS1, in general, results were better with unbalanced data, but with DS5, differences were acceptable between balanced and unbalanced data.

DTs left out several numerical variables considered as input. It could mean that some of them are not needed to train the algorithms. Regarding the different sets of variables considered and the most updated data set (DS5), a numerical set of variables was chosen.

In general, it seems that KNN and DTs were overfitting. However, it is necessary to have into account that, as aforementioned, the problem of errors, causes, and fixes was not settled over time. The study about the WiBACK networks was evolving, meaning that the data and assumptions had suffered changes. The data set provided for the last training moment (DS5) contains the last updates in data, making this training the most reliable one regarding the WiBACK conditions and the study of its networks. Besides, Table 3.3 shows that there are classes in the training that are not in DS2, DS3, neither DS4, which also shows these differences. Regarding both training sets (DS1 and DS5), SVM presented better capability of generalization.

Chapter 5.
Conclusions and Future Work

The goal of this work was to contribute for the further automatization of WiBACK. It was done with the application of different ML algorithms to classify errors, causes, and solutions, and the analysis of the relation between them. All the objectives of the work were accomplished.

A theoretical study about errors in a wireless system and respective causes and fixes was performed. Errors, causes, and fixes of the application case WiBACK were classified with DTs, KNN, and SVM after a selection of numerical variables to consider as input. Classification of errors, causes, and fixes was performed separately (three independent classification procedures). A PCA transformation was applied to data not just to reduce the dimensionality of the problem, but also to carry an exploratory study to conclude which principal components better explained visually the data according to classes of errors, causes, and fixes.

Regarding the tests and results presented, it was concluded that SVM are valid, which means that it is possible to detect and classify errors, causes, and solutions of the WiBACK link's faults accurately. Particularly, the classification of causes can be accurate without considering errors classes, and the classification of fixes can be accurate without considering errors and causes classes. It means that the automated monitoring of WiBACK traffic behavior is possible and can be accurate with a ML approach.

5.1 Future Work

The areas identified for improvement respect the collection of more data, running algorithms online, the relation between errors, causes, and solutions, and an autonomous implementation of fixes by the software.

The first step of the future work is to collect and treat more data, to test algorithms with more data, collected with the latest version of WiBACK software, to further study unforeseen behavior using new data. This data will have the last updates considered. To test algorithms in different locations, the WiBACK software in Porto should be updated. Afterwards, all the procedures could be applied in regions without technicians that WiBACK is installed. Additionally, as it is expected these ML algorithms to run online for fixing faults, it is also needed to understand the computational burden that may be imputed to the WiBACK controller. It is also needed to understand whether this can influence the system's reactions to such faults as well as the software's overall performance.

Regarding the relation between errors, causes, and fixes, to develop a methodology to identify the misclassified instances and use that relation to correct them is needed. If the performance is satisfactory online, the last stage of the work would be an implementation phase where the machine is capable of deciding whether the proposed solution to a specific problem is the best solution for the network as a whole.

For this purpose, the network behavior has to be taken into account, as a list of parameters that interfere with the network operation. Then, the labeled solutions might have weights defined according to not just the problem itself, but also these parameters, and changes with negative impact in the network that each solution may originate. This would be a possibility for how the network shall decide which solution is the best for each error.

Future work also includes a process of re-fixing errors, even if with weighted solutions, the network does not respond satisfactorily.

Bibliography

[1] Raouf Boutaba et al. "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities". In: *Journal of Internet Services and Applications* 9.1 (2018), p. 16.

[2] Hossam Alzamly. *Evaluation of errors on WiBACK wireless links*. Tech. rep. Cologne University of Applied Sciences, Faculty of Information, Media and Electrical Engineering, 2018.

[3] Hossam Alzamly. "Automated Error Detection and Classification using Machine Learning Techniques for WiBACK Wireless Links". MA thesis. Faculty of Information, Media and Electrical Engineering, 2019.

[4] *scikit-learn*. accessed on June 25. URL: https://scikit-learn.org/stable/.

[5] Chen-Mou Cheng et al. "Wsn07-1: Adjacent channel interference in dual-radio 802.11 a nodes and its impact on multi-hop networking". In: *IEEE Globecom 2006*. IEEE. 2006, pp. 1–6.

[6] Nadeem Ahmed, S Kanhere Salil, and Sanjay Jha. "Mitigating the effect of interference in wireless sensor networks". In: *IEEE Local Computer Network Conference*. IEEE. 2010, pp. 160–167.

[7] Gang Zhou et al. "RID: Radio interference detection in wireless sensor networks". In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 2. IEEE. 2005, pp. 891–901.

[8] Paul Fuxjager, Danilo Valerio, and Fabio Ricciato. "The myth of non-overlapping channels: interference measurements in IEEE 802.11". In: *2007 Fourth Annual Conference on Wireless on Demand Network Systems and Services*. IEEE. 2007, pp. 1–8.

[9] Hristo D Hristov. *Fresnal Zones in Wireless Links, Zone Plate Lenses and Antennas*. Artech House, Inc., 2000.

[10] Colin O'Reilly et al. "Anomaly detection in wireless sensor networks in a non-stationary environment". In: *IEEE Communications Surveys & Tutorials* 16.3 (2014), pp. 1413–1432.

[11] Sutharshan Rajasegarar et al. "Centered hyperspherical and hyperellipsoidal one-class support vector machines for anomaly detection in sensor networks". In: *IEEE Transactions on Information Forensics and Security* 5.3 (2010), pp. 518–533.

[12] Salah Zidi, Tarek Moulahi, and Bechir Alaya. "Fault detection in wireless sensor networks through SVM classifier". In: *IEEE Sensors Journal* 18.1 (2017), pp. 340–347.

[13] Thaha Muhammed and Riaz Ahmed Shaikh. "An analysis of fault detection strategies in wireless sensor networks". In: *Journal of Network and Computer Applications* 78 (2017), pp. 267–287.

[14] Muhammad Naveed Aman and Biplab Sikdar. "A CART based mechanism for collision detection in IEEE 802.11". In: *2012 IEEE Latin-America Conference on Communications*. IEEE. 2012, pp. 1–6.

[15] Tarun Joshi et al. "SARA: Stochastic automata rate adaptation for IEEE 802.11 networks". In: *IEEE Transactions on Parallel and Distributed Systems* 19.11 (2008), pp. 1579–1590.

[16] Pasi Sarolahti, Markku Kojo, and Kimmo Raatikainen. "F-RTO: an enhanced recovery algorithm for TCP retransmission timeouts". In: *ACM SIGCOMM Computer Communication Review* 33.2 (2003), pp. 51–63.

[17] Majid Gholipour, Abolfazl Toroghi Haghighat, and Mohammad Reza Meybodi. "Hop-by-Hop Congestion Avoidance in wireless sensor networks based on genetic support vector machine". In: *Neurocomputing* 223 (2017), pp. 63–76.

[18] Andrew L Beam and Isaac S Kohane. "Big data and machine learning in health care". In: *Jama* 319.13 (2018), pp. 1317–1318.

[19] Jun Chin Ang et al. "Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection". In: *IEEE/ACM transactions on computational biology and bioinformatics* 13.5 (2015), pp. 971–989.

[20] Filipe Rodrigues et al. "Learning supervised topic models for classification and regression from crowds". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2409–2422.

[21] Mirza Golam Kibria et al. "Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks". In: *IEEE access* 6 (2018), pp. 32328–32338.

[22] Nigel Williams, Sebastian Zander, and Grenville Armitage. "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification". In: *ACM SIGCOMM Computer Communication Review* 36.5 (2006), pp. 5–16.

[23] Giorgio Roffo, Simone Melzi, and Marco Cristani. "Infinite feature selection". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4202–4210.

[24] Jan Hauke and Tomasz Kossowski. "Comparison of Values of Pearson's and Spearman's Correlation Coefficients on the Same Sets of Data". In: *Quaestiones Geographicae* 30.2 (2011), pp. 87–93.

[25] Douglas M. Hawkins. "The Problem of Overfitting". In: *Journal of Chemical Information and Computer Sciences* 44.1 (2004), pp. 1–12.

[26] scikit-learn developers. *Grid Search*. accessed on June 25. 2007-2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[27] Álvaro Barbero Jiménez, Jorge López Lázaro, and José R Dorronsoro. "Finding optimal model parameters by discrete grid search". In: *Innovations in Hybrid Intelligent Systems*. Springer, 2007, pp. 120–127.

[28] Yoshua Bengio and Yves Grandvalet. "No unbiased estimator of the variance of k-fold cross-validation". In: *Journal of machine learning research* 5.Sep (2004), pp. 1089–1105.

[29] Thomas G Dietterich. "Approximate statistical tests for comparing supervised classification learning algorithms". In: *Neural computation* 10.7 (1998), pp. 1895–1923.

[30] Tom Dietterich. "Overfitting and Undercomputing in Machine Learning". In: *ACM Computing Surveys (CSUR)* 17.3 (1995).

[31] Tom M Mitchell, Sridbar Mabadevan, and Louis I Steinberg. "LEAP: A learning apprentice for VLSI design". In: *Machine Learning*. Elsevier, 1990, pp. 271–289.

[32] Mowei Wang et al. "Machine learning for networking: Workflow, advances and opportunities". In: *Ieee Network* 32.2 (2017), pp. 92–99.

[33] Shijun Huang et al. "A statistical-feature-based approach to internet traffic classification using machine learning". In: *2009 International Conference on Ultra Modern Telecommunications & Workshops*. IEEE. 2009, pp. 1–6.

[34] Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. "BLINC: multilevel traffic classification in the dark". In: *Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*. 2005, pp. 229–240.

[35] Anderson Santos Da Silva et al. "Identification and selection of flow features for accurate traffic classification in SDN". In: *2015 IEEE 14th International Symposium on Network Computing and Applications*. IEEE. 2015, pp. 134–141.

[36] Rui Wang et al. "Solving the app-level classification problem of P2P traffic via optimized support vector machines". In: *Sixth International Conference on Intelligent Systems Design and Applications*. Vol. 2. IEEE. 2006, pp. 534–539.

[37] Bryan Ng, Matthew Hayes, and Winston KG Seah. "Developing a traffic classification platform for enterprise networks with SDN: Experiences & lessons learned". In: *2015 IFIP Networking Conference (IFIP Networking)*. IEEE. 2015, pp. 1–9.

[38] Alice Este, Francesco Gringoli, and Luca Salgarelli. "Support vector machines for TCP traffic classification". In: *Computer Networks* 53.14 (2009), pp. 2476–2490.

[39] Matthew Roughan et al. "Class-of-service mapping for QoS: a statistical signature-based approach to IP traffic classification". In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. 2004, pp. 135–148.

[40] Jun Zhang et al. "Network traffic classification using correlation information". In: *IEEE Transactions on Parallel and Distributed systems* 24.1 (2012), pp. 104–117.

[41] Paulo Valente Klaine et al. "A survey of machine learning techniques applied to self-organizing cellular networks". In: *IEEE Communications Surveys & Tutorials* 19.4 (2017), pp. 2392–2431.

[42] Samer Fayssal, Salim Hariri, and Youssif Al-Nashif. "Anomaly-based behavior analysis of wireless network security". In: *2007 Fourth Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services (MobiQuitous)*. IEEE. 2007, pp. 1–8.

[43] Hamid Alipour et al. "Wireless anomaly detection based on IEEE 802.11 behavior analysis". In: *IEEE transactions on information forensics and security* 10.10 (2015), pp. 2158–2170.

[44] John Felix Charles Joseph et al. "Cross-layer detection of sinking behavior in wireless ad hoc networks using SVM and FDA". In: *IEEE Transactions on Dependable and Secure Computing* 8.2 (2010), pp. 233–245.

[45] Pavel Laskov et al. "Incremental support vector learning: Analysis, implementation and applications". In: *Journal of machine learning research* 7.Sep (2006), pp. 1909–1936.

[46] Pierre Geurts, Ibtissam El Khayat, and Guy Leduc. "A machine learning approach to improve congestion control over wireless computer networks". In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*. IEEE. 2004, pp. 383–386.

[47] Oluwakayode Onireti et al. "A cell outage management framework for dense heterogeneous networks". In: *IEEE Transactions on Vehicular Technology* 65.4 (2015), pp. 2097–2113.

[48] Ahmed Zoha et al. "Data-driven analytics for automated cell outage detection in Self-Organizing Networks". In: *2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE. 2015, pp. 203–210.

[49] Christian M Mueller et al. "A cell outage detection algorithm using neighbor cell list reports". In: *International Workshop on Self-Organizing Systems*. Springer. 2008, pp. 218–229.

[50] David D Clark et al. "A knowledge plane for the internet". In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. 2003, pp. 3–10.

[51] Thomas Glen Dietterich, Pat Langley, et al. "Machine learning for cognitive networks: Technology assessment and research challenges". In: (2003).

[52] Ryan W Thomas et al. "Cognitive networks: adaptation and learning to achieve end-to-end performance objectives". In: *IEEE Communications magazine* 44.12 (2006), pp. 51–57.

[53] Carolina Fortuna and Mihael Mohorcic. "Trends in the development of communication networks: Cognitive networks". In: *Computer networks* 53.9 (2009), pp. 1354–1376.

[54] Beibei Wang and KJ Ray Liu. "Advances in cognitive radio networks: A survey". In: *IEEE Journal of selected topics in signal processing* 5.1 (2010), pp. 5–23.

[55] Won-Yeol Lee and Ian F Akyildiz. "Optimal spectrum sensing framework for cognitive radio networks". In: *IEEE Transactions on wireless communications* 7.10 (2008), pp. 3845–3857.

[56] Xue-wen Chen and Michael Wasikowski. "FAST: A ROC-based Feature Selection Metric for Small Samples and Imbalanced Data Classification Problems". In: *KDD08: The 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2008, pp. 124–132.

[57] David A. Cieslak and Nitesh V. Chawla. "Learning Decision Trees for Unbalanced Data". In: *Machine Learning and Knowledge Discovery in Databases*. Springer, 2008, pp. 241–256.

[58] Nitesh V. Chawla et al. "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357.

[59] Alberto Fernández et al. "SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary". In: *Journal of artificial intelligence research* 61 (2018), pp. 863–905.

[60] Prabhjot Kaur and Anjana Gosain. "Comparing the behavior of oversampling and undersampling approach of class imbalance learning by combining class imbalance problem with noise". In: *ICT Based Innovations*. Springer, 2018, pp. 23–30.

[61] Jean-Francois Le Téno. "Visual data analysis and decision support methods for non-deterministic LCA". In: *The International Journal of Life Cycle Assessment* 4.1 (1999), pp. 41–47.

[62] Ian T Jolliffe and Jorge Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.

[63] Jonathon Shlens. "A tutorial on principal component analysis". In: *arXiv preprint arXiv:1404.1100* (2014).

[64] Anderson Santos Da Silva et al. "Identification and selection of flow features for accurate traffic classification in SDN". In: *2015 IEEE 14th International Symposium on Network Computing and Applications*. IEEE. 2015, pp. 134–141.

[65] Y Le Borgne and Gianluca Bontempi. "Unsupervised and supervised compression with principal component analysis in wireless sensor networks". In: *Proceedings of the Workshop on Knowledge Discovery from Data, 13th ACM International Conference on Knowledge Discovery and Data Mining*. 2007, pp. 94–103.

[66] Fouzi Harrou, M Nounou, and H Nounou. "A statistical fault detection strategy using PCA based EWMA control schemes". In: *2013 9th Asian Control Conference (ASCC)*. IEEE. 2013, pp. 1–4.

[67] Lior Rokach and Oded Maimon. "Decision trees". In: *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 165–192.

[68] J. Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (1986), pp. 81–106.

[69] scikit-learn developers. *Decision Trees*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/tree.html.

[70] scikit-learn developers. *DecisionTreeClassifier*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[71] scikit-learn developers. *Post pruning decision trees with cost complexity pruning*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py.

[72] Sayali D Jadhav and HP Channe. "Comparative study of K-NN, naive Bayes and decision tree classification techniques". In: *International Journal of Science and Research (IJSR)* 5.1 (2016), pp. 1842–1845.

[73] Sadegh Bafandeh Imandoust and Mohammad Bolandraftar. "Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background". In: *International Journal of Engineering Research and Applications* 3.5 (2013), pp. 605–610.

[74] Padraig Cunningham and Sarah Jane Delany. "k-Nearest Neighbour Classifiers–". In: *arXiv preprint arXiv:2004.04523* (2020).

[75] scikit-learn developers. *Nearest Neighbors*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/neighbors.html.

[76] scikit-learn developers. *KNeighborsClassifier*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[77] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine learning* 20.3 (1995), pp. 273–297.

[78] Léon Bottou and Chih-Jen Lin. "Support vector machine solvers". In: *Large scale kernel machines* 3.1 (2007), pp. 301–320.

[79] Rui Wang et al. "Solving the app-level classification problem of P2P traffic via optimized support vector machines". In: *Sixth International Conference on Intelligent Systems Design and Applications*. Vol. 2. IEEE. 2006, pp. 534–539.

[80] Vladimir N Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999.

[81] Vladimir N. Vapnik. *Statistical Learning Theory*. 1998.

[82] scikit-learn developers. *Support Vector Machines*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/svm.html.

[83] scikit-learn developers. *SVC*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[84] Gexiang Zhang, Zhexin Cao, and Yajun Gu. "A hybrid classifier based on rough set theory and support vector machines". In: *International Conference on Fuzzy Systems and Knowledge Discovery*. Springer. 2005, pp. 1287–1296.

[85] Ryan Rifkin and Aldebaro Klautau. "In defense of one-vs-all classification". In: *Journal of machine learning research* 5.Jan (2004), pp. 101–141.

[86] Jun Guo, Norikazu Takahashi, and Wenxin Hu. "An efficient algorithm for multi-class support vector machines". In: *2008 International Conference on Advanced Computer Theory and Engineering*. IEEE. 2008, pp. 327–331.

[87] Andrea Dal Pozzolo et al. "Calibrating probability with undersampling for unbalanced classification". In: *2015 IEEE Symposium Series on Computational Intelligence*. IEEE. 2015, pp. 159–166.

[88] scikit-learn developers. *f1-score*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.metrics.f1_score.

[89] scikit-learn developers. *accuracy score*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html#sklearn.metrics.accuracy_score.

[90] scikit-learn developers. *balanced accuracy score*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html#sklearn.metrics.balanced_accuracy_score.

[91] Zanifa Omary and Fredrick Mtenzi. "Dataset threshold for the performance estimators in supervised machine learning experiments". In: *2009 International Conference for Internet Technology and Secured Transactions,(ICITST)*. IEEE. 2009, pp. 1–8.

[92] Mohammad Hossin and MN Sulaiman. "A review on evaluation metrics for data classification evaluations". In: *International Journal of Data Mining & Knowledge Management Process* 5.2 (2015), p. 1.

[93] Xingquan Zhu and Xindong Wu. "Class noise vs. attribute noise: A quantitative study". In: *Artificial intelligence review* 22.3 (2004), pp. 177–210.

[94] scikit-learn developers. *confusion matrix*. accessed on June 25. 2007-2019. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

Appendices

Chapter A.

Confusion matrices

Table A.1: Confusion matrices for errors classification with DS2 as the test set.

DS	**DTs**					**KNN**					**SVM**				
U		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	369	173	0	0	E1	0	542	0	0	E1	539	3	0	0
	E2	1	403	7	0	E2	0	411	0	0	E2	29	377	0	5
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0
B		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	0	526	1	15	E1	0	542	0	0	E1	539	0	0	3
	E2	0	317	1	93	E2	0	411	0	0	E2	36	139	0	236
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0
PCA		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	84	458	0	0	E1	0	542	0	0	E1	0	542	0	0
	E2	32	355	24	0	E2	0	378	33	0	E2	0	411	0	0
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0

Table A.2: Confusion matrices for errors classification with DS3 as the test set.

DS	**DTs**					**KNN**					**SVM**				
U		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	369	173	0	0	E1	0	542	0	0	E1	539	3	0	0
	E2	1	364	7	0	E2	0	372	0	0	E2	29	338	0	5
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0
B		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	0	526	1	15	E1	0	542	0	0	E1	539	0	0	3
	E2	0	278	1	93	E2	0	372	0	0	E2	36	100	0	236
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0
PCA		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	84	458	0	0	E1	0	542	0	0	E1	0	542	0	0
	E2	24	348	0	0	E2	0	372	0	0	E2	0	372	0	0
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0

Table A.3: Confusion matrices for errors classification with DS4 as the test set.

Data set	**DTs**					**KNN**					**SVM**				
Unbalanced		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	369	212	0	0	E1	20	561	0	0	E1	557	4	0	0
	E2	1	498	13	0	E2	38	473	1	0	E2	45	446	13	8
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0
Balanced		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	369	212	0	0	E1	20	561	0	0	E1	557	1	0	3
	E2	1	498	13	0	E2	63	448	1	0	E2	56	173	15	262
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0
PCA		E1	E2	E3	E4		E1	E2	E3	E4		E1	E2	E3	E4
	E1	105	476	0	0	E1	20	562	0	0	E1	2	579	0	0
	E2	61	427	24	0	E2	28	450	34	0	E2	0	512	0	0
	E3	0	0	0	0	E3	0	0	0	0	E3	0	0	0	0
	E4	0	0	0	0	E4	0	0	0	0	E4	0	0	0	0

Table A.4: Confusion matrices for causes classification with DS2 as the test set.

DS	**DTs**						**KNN**						**SVM**					
U		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	447	92	3	0	0	C1	0	542	0	0	0	C1	539	3	0	0	0
	C2	3	247	10	0	0	C2	0	260	0	0	0	C2	4	219	7	0	30
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	39	0	0	0	C4	0	39	0	0	0	C4	0	0	0	0	39
	C5	48	15	49	0	0	C5	0	112	0	0	0	C5	26	72	1	0	13
B		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	362	97	83	0	0	C1	0	542	0	0	0	C1	540	2	0	0	0
	C2	23	1	13	0	223	C2	0	233	0	0	0	C2	4	235	8	0	13
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	0	0	0	39	C4	0	39	0	0	0	C4	0	1	0	0	38
	C5	2	1	101	0	0	C5	0	112	0	0	0	C5	31	80	1	0	0
PCA		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	81	461	0	0	0	C1	0	542	0	0	0	C1	0	542	0	0	0
	C2	2	258	0	0	0	C2	0	260	0	0	0	C2	0	260	0	0	0
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	8	10	21	0	0	C4	0	2	37	0	0	C4	0	39	0	0	0
	C5	22	90	0	0	0	C5	0	112	0	0	0	C5	0	112	0	0	0

Table A.5: Confusion matrices for causes classification with DS3 as the test set.

DS	**DTs**						**KNN**						**SVM**					
U		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	447	92	3	0	0	C1	0	542	0	0	0	C1	539	3	0	0	0
	C2	3	247	10	0	0	C2	0	260	0	0	0	C2	4	219	7	0	30
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	0	0	0	0	C4	0	0	0	0	0	C4	0	0	0	0	0
	C5	48	15	49	0	0	C5	0	112	0	0	0	C5	26	72	1	0	13
B		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	362	97	83	0	0	C1	0	542	0	0	0	C1	540	2	0	0	0
	C2	23	1	13	0	223	C2	0	260	0	0	0	C2	4	235	8	0	13
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	0	0	0	0	C4	0	0	0	0	0	C4	0	0	0	0	0
	C5	2	1	101	0	98	C5	0	112	0	0	0	C5	31	80	1	0	0
PCA		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	81	461	0	0	0	C1	0	542	0	0	0	C1	0	542	0	0	0
	C2	2	258	0	0	0	C2	0	260	0	0	0	C2	0	260	0	0	0
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	0	0	0	0	C4	0	0	0	0	0	C4	0	0	0	0	0
	C5	22	90	0	0	0	C5	0	112	0	0	0	C5	0	112	0	0	0

Table A.6: Confusion matrices for causes classification with DS4 as the test set.

DS	DTs						KNN						SVM					
U		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	477	101	3	0	14	C1	20	561	0	0	0	C1	577	3	0	0	1
	C2	3	288	16	0	11	C2	15	302	1	0	0	C2	4	244	37	0	33
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	39	0	0	0	C4	0	39	0	0	0	C4	0	0	0	0	39
	C5	48	15	49	0	43	C5	20	135	0	0	0	C5	57	74	1	0	23
B		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	371	97	91	0	22	C1	20	561	0	0	0	C1	578	2	0	0	1
	C2	23	1	23	0	271	C2	40	177	1	0	0	C2	4	157	41	0	16
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	0	0	0	0	39	C4	0	39	0	0	0	C4	0	1	0	0	38
	C5	2	1	0	0	152	C5	20	135	0	0	0	C5	53	83	1	0	18
PCA		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5		C1	C2	C3	C4	C5
	C1	102	479	0	0	0	C1	19	562	0	0	0	C1	2	579	0	0	0
	C2	8	310	0	0	0	C2	8	309	1	0	0	C2	0	317	1	0	0
	C3	0	0	0	0	0	C3	0	0	0	0	0	C3	0	0	0	0	0
	C4	8	10	21	0	0	C4	0	2	37	0	0	C4	0	39	0	0	0
	C5	45	110	0	0	0	C5	20	135	0	0	0	C5	0	154	1	0	0

Table A.7: Confusion matrices for fixes classification with DS2 as the test set.

DS	DTs						KNN						SVM					
U		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	447	92	3	0	0	F1	0	542	0	0	0	F1	539	3	0	0	0
	F2	3	286	10	0	0	F2	0	299	0	0	0	F2	35	260	0	0	4
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	48	15	49	0	0	F4	0	112	0	0	0	F4	28	83	0	0	1
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0
B		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	0	525	2	0	15	F1	0	542	0	0	0	F1	539	0	0	0	3
	F2	0	286	0	0	13	F2	21	266	0	0	155	F2	21	123	0	0	155
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	0	29	3	0	80	F4	0	112	0	0	0	F4	29	2	0	0	81
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0
PCA		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	81	461	0	0	0	F1	0	542	0	0	0	F1	0	542	0	0	0
	F2	10	268	21	0	0	F2	0	266	33	0	0	F2	0	299	0	0	0
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	22	90	0	0	0	F4	0	112	0	0	0	F4	0	112	0	0	0
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0

Table A.8: Confusion matrices for causes classification with DS3 as the test set.

DS	DTs						KNN						SVM					
U		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	447	92	3	0	0	F1	0	542	0	0	0	F1	539	3	0	0	0
	F2	3	247	10	0	0	F2	0	260	0	0	0	F2	30	226	0	0	4
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	48	15	49	0	0	F4	0	112	0	0	0	F4	28	83	0	0	1
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0
B		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	0	525	2	0	15	F1	0	542	0	0	0	F1	539	0	0	0	3
	F2	0	247	0	0	13	F2	0	247	0	0	13	F2	16	89	0	0	155
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	0	29	3	0	80	F4	0	112	0	0	0	F4	29	2	0	0	81
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0
PCA		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	81	461	0	0	0	F1	0	542	0	0	0	F1	0	542	0	0	0
	F2	2	258	0	0	0	F2	0	260	0	0	0	F2	0	260	0	0	0
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	22	90	0	0	0	F4	0	112	0	0	0	F4	0	112	0	0	0
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0

Table A.9: Confusion matrices for fixes classification with DS4 as the test set.

DS	DTs						KNN						SVM					
U		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	463	101	3	14	0	F1	20	561	0	0	0	F1	578	3	0	0	0
	F2	3	327	16	11	0	F2	21	335	1	0	0	F2	35	308	7	1	6
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	48	15	49	0	43	F4	20	135	0	0	0	F4	60	94	0	0	1
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0
B		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	0	545	2	19	15	F1	20	561	0	0	0	F1	578	0	0	0	3
	F2	0	287	9	1	60	F2	40	316	1	0	0	F2	21	174	4	0	158
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	0	29	26	20	80	F4	20	135	0	0	0	F4	45	6	0	0	104
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0
PCA		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5		F1	F2	F3	F4	F5
	F1	92	479	0	0	0	F1	19	562	0	0	0	F1	2	579	0	0	0
	F2	16	320	21	0	0	F2	8	315	34	0	0	F2	0	357	0	0	0
	F3	0	0	0	0	0	F3	0	0	0	0	0	F3	0	0	0	0	0
	F4	45	110	0	0	0	F4	20	135	0	0	0	F4	1	154	0	0	0
	F5	0	0	0	0	0	F5	0	0	0	0	0	F5	0	0	0	0	0

Chapter B.

Hyperparameters of the trained algorithms

Table B.1: Optimized hyperparameters of DTs trained with DS1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Maximum depth	Split metric	Maximum leaf nodes	Node split	Complexity parameter
E	**Unbalanced**	4	Entropy	9	Best	0
	Balanced	5	Gini	9	Best	0
	PCA	4	Entropy	9	Best	0
C	**Unbalanced**	5	Gini	9	Best	0
	Balanced	5	Gini	9	Best	0
	PCA	4	Gini	7	Best	0
F	**Unbalanced**	5	Gini	9	Best	0
	Balanced	4	Gini	9	Best	0
	PCA	4	Gini	7	Best	0

Table B.2: Optimized hyperparameters of KNN trained with DS1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	K	More important points to the query point	Distance
E	**Unbalanced**	5	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	6	The closer ones	Euclidean
C	**Unbalanced**	6	All equal	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	4	All equal	Euclidean
F	**Unbalanced**	8	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	6	The closer ones	Euclidean

Table B.3: Optimized hyperparameters of SVM trained with DS1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Kernel
E	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Radial basis function
C	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Radial basis function
F	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Radial basis function

Table B.4: Optimized hyperparameters of DTs when the algorithms were retrained with DS1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Maximum depth	Split metric	Maximum leaf nodes	Node split	Complexity parameter
E	**Unbalanced**	4	Gini	8	Best	0.015
	Balanced	3	Gini	8	Best	0
	PCA	5	Gini	9	Best	0
C	**Unbalanced**	3	Gini	8	Best	0
	Balanced	5	Entropy	9	Best	0
	PCA	4	Gini	9	Best	0
F	**Unbalanced**	3	Gini	8	Best	0
	Balanced	5	Gini	9	Best	0
	PCA	3	Entropy	6	Best	0

Table B.5: Optimized hyperparameters of KNN when the algorithms were retrained with DS1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	K	More important points to the query point	Distance
E	**Unbalanced**	5	All equal	Euclidean
	Balanced	2	The closer ones	Euclidean
	PCA	5	The closer ones	Euclidean
C	**Unbalanced**	8	The closer ones	Chebyshev
	Balanced	2	The closer ones	Euclidean
	PCA	9	The closer ones	Manhattan
F	**Unbalanced**	4	The closer ones	Euclidean
	Balanced	2	The closer ones	Euclidean
	PCA	9	The closer ones	Manhattan

Table B.6: Optimized hyperparameters of SVM when the algorithms were retrained with DS1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Kernel
E	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear
C	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear
F	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear

Table B.7: Optimized hyperparameters of DTs trained with DS5 - 1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Maximum depth	Split metric	Maximum leaf nodes	Node split	Complexity parameter
E	**Unbalanced**	4	Entropy	9	Best	0
	Balanced	5	Entropy	8	Best	0
	PCA	4	Entropy	9	Best	0
C	**Unbalanced**	4	Gini	9	Best	0
	Balanced	5	Gini	7	Best	0
	PCA	4	Gini	9	Best	0
F	**Unbalanced**	4	Entropy	7	Best	0
	Balanced	5	Entropy	9	Best	0
	PCA	5	Entropy	9	Best	0.025

Table B.8: Optimized hyperparameters of KNN trained with DS5 - 1 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	K	More important points to the query point	Distance
E	**Unbalanced**	4	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	3	The closer ones	Chebyshev
C	**Unbalanced**	9	The closer ones	Manhattan
	Balanced	9	The closer ones	Euclidean
	PCA	9	The closer ones	Manhattan
F	**Unbalanced**	6	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	8	The closer ones	Euclidean

Table B.9: Optimized hyperparameters of DTs by GS to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Kernel
E	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear
C	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Radial basis function
F	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear

Table B.10: Optimized hyperparameters of DTs trained with DS5 - 2 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Maximum depth	Split metric	Maximum leaf nodes	Node split	Complexity parameter
E	**Unbalanced**	4	Entropy	9	Best	0
	Balanced	4	Entropy	9	Best	0
	PCA	4	Entropy	8	Best	0
C	**Unbalanced**	4	Gini	9	Best	0
	Balanced	7	Entropy	9	Best	0
	PCA	5	Entropy	9	Best	0
F	**Unbalanced**	5	Gini	9	Best	0
	Balanced	5	Gini	8	Best	0
	PCA	4	Gini	9	Best	0

Table B.11: Optimized hyperparameters of KNN trained with DS5 - 2 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	K	More important points to the query point	Distance
E	**Unbalanced**	4	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	3	All equal	Chebyshev
C	**Unbalanced**	7	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	4	The closer ones	Chebyshev
F	**Unbalanced**	4	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	2	The closer ones	EUclidean

Table B.12: Optimized hyperparameters of SVM trained by DS5 - 2 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Kernel
E	**Unbalanced**	Linear
	Balanced	Polynomial with degree
	PCA	Linear
C	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear
F	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear

Table B.13: Optimized hyperparameters of DTs trained with DS5 - 3 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Maximum depth	Split metric	Maximum leaf nodes	Node split	Complexity parameter
E	**Unbalanced**	5	Entropy	8	Best	0
	Balanced	4	Entropy	7	Best	0
	PCA	3	Entropy	5	Best	0
C	**Unbalanced**	5	Entropy	9	Best	0
	Balanced	4	Entropy	9	Best	0
	PCA	5	Entropy	8	Best	0
F	**Unbalanced**	5	Entropy	7	Best	0
	Balanced	4	Entropy	8	Best	0
	PCA	6	Gini	8	Best	0

Table B.14: Optimized hyperparameters of KNN trained with KNN to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	K	More important points to the query point	Distance
E	**Unbalanced**	3	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	4	The closer ones	Euclidean
C	**Unbalanced**	8	The closer ones	Euclidean
	Balanced	9	The closer ones	Manhattan
	PCA	9	The closer ones	Chebyshev
F	**Unbalanced**	7	The closer ones	Manhattan
	Balanced	2	The closer ones	Manhattan
	PCA	4	The closer ones	Manhattan

Table B.15: Optimized hyperparameters of SVM trained with DS5 - 3 to classify the errors (E), causes (C), and fixes (F) with the unbalanced, balanced, and PCA data sets.

	Data set	Kernel
E	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear
C	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Radial basis function
F	**Unbalanced**	Linear
	Balanced	Linear
	PCA	Linear