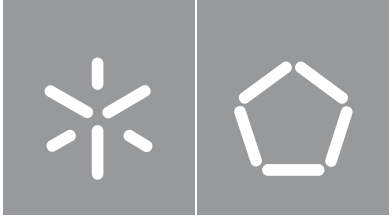
Universidade do Minho
Escola de Engenharia

Diana Sofia Nogueira Costa

Predicting Problems From Telecom Installation Processes

October 2020

Diana Costa **Predicting Probems From Telecom Installation Processes**
UMinho | 2020



Universidade do Minho
Escola de Engenharia

Diana Sofia Nogueira Costa

Predicting Problems From Telecom Installation Processes

Master's Dissertation
Integrated Master in Informatics Engineering

Dissertation supervised by
Professor Doctor Hugo Daniel Abreu Peixoto
Professor Doctor José Manuel Ferreira Machado

October 2020

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO POR TERCEIROS

Os homens são anjos nascidos sem asas,
é o que há de mais bonito, nascer sem asas e fazê-las crescer.

José Saramago in "Memorial do Convento".

ACKNOWLEDGEMENTS
AGRADECIMENTOS

To Profs. Drs. Hugo Peixoto and José Machado, for the way they guided this study, for the attention and availability, as they promoted the start and overcoming in each distinct stage.

To Dr. Carlos Pereira, for the teachings and pertinence of his critics and suggestions, for all the support, kindness, and dedication throughout the realization of this thesis, being the main propellant in the choice and development of this theme.

To the University of Minho, my second home for five years, for the demand and quality of the teachers I had the opportunity to meet, who made me grow at a personal and professional level.

To Sérgio and Vítor, for their friendship and mutual help throughout the academic years and this project. With you, any challenge becomes much easier.

To my friends, who have always been rooting for me. I thank you for your joy and companionship, and for always believing in me.

Lastly, knowing that without them, none of this would be possible, special thanks to my family, for all they taught me, for their patience, effort, and unconditional support, and for being models of courage and strength throughout my life.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Predicting Problems From Telecom Installation Processes

Improving customer experience is crucial in any industry, especially in telecommunications, where competition is a constant factor. Today, all telecommunications companies rely on the massive amount of data generated daily to get to know the customer, study their behavior, and create new effective strategies for their business. The information collected may include network information, representing the status of hardware and software components in the network, and the client's details and calls to support and customer assistance data, which describes problems, consumer complaints, and all other problem-solving interactions. By combining these mountains of raw data with data mining and machine learning techniques, companies can find solid patterns or systematic relationships that represent valuable information, that is, generate knowledge.

Within the most varied user experiences, the process of installing new services can be an event that raises doubts about their operation, degrade the user experience, or, in extreme cases, lead to maintenance interventions. Therefore, the use of advanced predictive models that can predict such occurrences become vital. With this, the company can anticipate the cases that will be problematic and reduce the number of negative experiences.

The main objective of this work is to create a predictive model that, through all the available data history, can predict which customers will contact the customer service with problems derived from the installation process and have a following maintenance intervention.

After analyzing an imbalanced dataset with approximately 560K entries from a Portuguese telecommunications company, and resorting to the CRISP-DM methodology for modeling, the best results were found with LightGBM, which obtained an AUPRC of 0.11 and AUROC of 0.62. The best trade-off between precision and recall was found with a threshold model of 0.43 in order to maximize recall while still avoiding a large number of false negatives. The strategies explored in this work and the challenges found may help the company understand which details should improve in its service provision, and which data still need to be investigated in the future.

Keywords: Customer, Data Mining, Installation, Machine Learning, Predict, Service, Telecommunications.

RESUMO

Previsão de Problemas Decorrentes de Instalações no Setor das Telecomunicações

Melhorar a experiência do cliente é crucial em qualquer setor, principalmente nas telecomunicações onde a concorrência é um fator constante. Atualmente, todas as empresas de telecomunicações recorrem à quantidade de dados colossal gerada diariamente para conhecer o cliente, estudar o seu comportamento e assim criar novas estratégias eficazes para o seu negócio. As informações recolhidas podem incluir relatórios de rede, que representam o estado dos componentes de hardware e software na rede, e detalhes sobre o cliente e contactos ao suporte e apoio ao cliente, que descrevem problemas, reclamações de consumidores e todas as demais interações para resolução de problemas. Juntando estas montanhas de dados brutos a técnicas de Data Mining e Machine Learning, as empresas são capazes de encontrar padrões sólidos ou relacionamentos sistemáticos que representem informações valiosas, ou seja, gerar conhecimento e inteligência.

Dentro das mais variadas experiências do utilizador, o processo de instalação de novos serviços pode ser um evento que origina dúvidas sobre a sua operação, degrada a experiência de utilização ou, em casos extremos, leva a intervenções de manutenção. Para tal, o uso de modelos avançados de previsão que consigam prever tais ocorrências torna-se vital. Com isto, é possível à empresa antecipar os casos que serão problemáticos e reduzir o número de experiências negativas.

Esta dissertação tem como principal objetivo criar um modelo de previsão que, através de todo o histórico de dados disponível, consiga prever quais os clientes que irão contactar o serviço de apoio ao cliente com problemas decorrentes do processo de instalação e ter uma intervenção de manutenção de seguida.

Após analisar um conjunto de dados não balanceado com aproximadamente 560K entradas de uma empresa portuguesa de telecomunicações, e recorrendo à metodologia CRISP-DM para modelação, os melhores resultados foram encontrados com o LightGBM, que obteve um AUPRC de 0.11 e AUROC de 0.62. O melhor *trade-off* entre a precisão e a *recall* foi encontrado com um *threshold* de modelo de 0.43, de forma a maximizar a *recall*, evitando um grande número de falsos negativos. As estratégias exploradas neste trabalho e os desafios encontrados podem ajudar a empresa a entender que detalhes do seu provisionamento de serviços devem melhorar e quais os dados que ainda precisam de ser investigados no futuro.

Palavras-chave: Cliente, *Data Mining*, Instalação, *Machine Learning*, Previsão, Serviço, Telecomunicações.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND ABBREVIATIONS

A

ANN Artificial Neural Network

AUPRC Area Under Precision-Recall Curve

AUROC Area Under the Receiver Operating Characteristic

C

CRISP-DM CRoss Industry Standard Process for Data Mining

CSV Comma Separated Values

D

DM Data Mining

DT Decision Trees

G

GNB Gaussian Naïve Bayes

GB Gradient Boosting

GBM Gradient Boosting Machine

I

IQR Interquartile Range

K

KDD Knowledge Discovery from Data

kNN k-nearest neighbour

L

LightGBM Light Gradient Boosting Machine

LinearSVC Linear Support Vector Classifier

LR Logistic Regression

M

ML Machine Learning

N

NB Naïve Bayes

R

RF Random Forests

ROC Receiver Operating Characteristic

ROS Random Oversampling

RUS Random Undersampling

S

SMOTE Synthetic Minority Oversampling

SVM Support Vector Machines

X

XGBoost EXtreme Gradient Boosting

1. INTRODUCTION

The growth and evolution of technology and telecommunications services are increasing the amount of data generated daily. This leads companies to adopt new strategies and improving the usability and utility of data and telecommunications services. In this very competitive environment, improving customer experience means loyalty, profit, and eventually culminates in a company's success. In fact, a growing number of companies choose customer satisfaction as their main performance indicator, and it is one of the fastest-growing segments of the marketing field.

However, many typical processes, like the installation of new services or their day-to-day utilization, are not always pleasant. Questions about equipment operation, network latencies or failure, lack of customer support or transparency, or repeated recurrences of a certain problem are some real-world complications that lead customers to look for better solutions and services. In Portugal, telecommunications have been in first place for over 12 years in the ranking of complaints coming to DECO, a Portuguese association for consumer protection (DECO 2019). Besides that, the massive amount of data generated daily is not always an astonishing thing. Due to its inappropriate semantic level in some cases, or the scalability of data mining methods used to extract information from databases, the attempts to improve customer satisfaction can be hampered.

Because of the efforts to address these issues, the telecommunications industry has been a leader in the data mining and machine learning area. DM techniques and ML models have been used to improve brand loyalty, increase company revenue, marketing and retail, network optimization, security and fraud analysis, among others. If a company, through data and information from the past, can learn from its mistakes, certain inconveniences will be avoided in the future. For example, suitable technicians can be chosen for each type of service, services and equipment can be customized depending on the client's needs, or customer churn can be predicted.

1.1 Motivation and Problem

Any company that provides services to a customer has to go through its installation processes, which usually involves either a specialist technician going to the client's house or self-installing it. In both cases, this is a procedure where the novelty factor exists, and where service may be problematic because of some external influencing factor. In the case of telecommunications, the process of installing new services requires a technician to visit the customer's home. An installation process is defined as the establishment of a service, such as the internet or telephone, where specialized knowledge of telecommunications is required for the person performing the craft.

This process of installing new systems can be a problematic event that leads to following maintenance interventions resulting in user experience degradation. If a particular defective device or service typically leads to bad experiences, or if the person is not properly helped by contacting the customer service system several times with a reduced resolution of some problem, these are

problems that the company wants to avoid. The integration in the telecommunications company's market and customer intelligence team made it possible to contact real problems in a business context, which weighed when choosing the theme of this dissertation. As stated in the previous section, improving the user experience results in loyalty and profit for the telecommunications company.

This requires advanced forecasting models that can predict such occurrences. There are already several successful implementations of churn prediction, fraud, among others. However, the focus on predicting problems arising from installing new services is not yet studied. That causes businesses to lose profit since equipment redirected to people's homes has high costs. A personalized service (suitable equipment or technician, for example) for each type of customer avoids many inconveniences for both the company and the customer, usually involving successive trips from the technician to the clients' home or successive calls from the customer to client support.

1.2 Goals

The main goal of this work is to create a predictive model that, through all the available historical data, can predict with high performance which customers will contact customer support with problems arising from the installation process and have a following maintenance intervention. In order to achieve this, data mining techniques and machine learning models will be used because they facilitate planning and provide managers with reliable forecasts based on past trends and current conditions. Data mining also allows for more efficient use and allocation of resources, which is the ultimate and subsequent goal. Thus, this dissertation aims to, ultimately:

- Know in advance who will complain and be aware of the customer's probable inconveniences, having a proactive action and dialogue with the client before the complaint call occurs, responding efficiently to the situation.
- By the past cases, knowing that a certain person will have an installation problem, a personalized service can be applied (suitable assistance, equipment, or technician), culminating in better customer experience and higher installation service efficiency.
- Reducing the number of problems also diminishes the time/equipment spent (number of trips to the customer's home or expensive equipment replacement), resulting in increased profit for the company.

1.3 Contributions

This dissertation had a scientific article reviewed and accepted for publication at the 21st International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2020. The article is detailed in Appendix I.

- Costa, D., Pereira, C., Peixoto, H. & and Machado, J. (2020) 'Anticipating Maintenance in Telecom Installation Processes', *IDEAL 2020: 21st International Conference on Intelligent Data Engineering and Automated Learning*, Guimarães, 4-6 November 2020.

1.4 Document Structure

This document has four chapters, organized as follows: Chapter 1 is an introductory chapter that addresses the motivation of this dissertation and the real-life problem that is explored. Here, the objectives for the thesis and its contributions are also defined.

Chapter 2 is a background chapter, which describes the main concepts and techniques that support this work. It starts by explaining machine learning types and making an approach to data mining, including processes and DM tasks, closing the thematic with a bridge between data mining and telecommunications. This chapter also addresses techniques used to improve classification accuracy and machine learning main models, including the performance metrics, model evaluation, and hyperparameter optimization methods relevant to this document. A study on imbalanced datasets and how to tackle them is taken. The chapter also summarizes typical data preprocessing techniques, including data integration, conventional data cleaning methods, and data transformation approaches. It ends with the related work about problems arising from installations problems, or articles that use similar techniques to those used in this thesis, whether in telecommunications or any other area.

Chapter 3 details the work methodology used and all the phases until reaching the results. This structured and well-proven methodology, CRISP-DM, is commonly used to plan a data science project and transform company data into knowledge and management information. Thus, with the concepts from the previous chapter, and through phases such as understanding the problem, comprehending and visualizing the data, preparing the data, modeling, and evaluating results, a robust resolution of the problem of this thesis was achieved. This chapter closes with a final discussion about the obtained results and contributions and checks whether the solution meets the initial work goals.

Finally, Chapter 4 makes a balance of the work developed so far and presents what can be done in future work to improve the achieved results.

2. BACKGROUND

In this section will be presented different data mining and machine learning concepts and techniques that support this dissertation, including the related studies that use similar approaches to those that will be used in this work.

2.1 Machine Learning

According to Stanford (2019), machine learning is the science of getting computers to act without being explicitly programmed. It is seen as a fundamental subset of artificial intelligence, using algorithms that can learn from data without relying on rule-based programming, making them perform better in the future. When exposed to new data, these computer programs can learn, change, develop, and grow by themselves. In short, the more the data, the higher will be the accuracy to learn and predict the results.

Machine learning implementation types can be classified into three broad categories - Supervised, Unsupervised, and Reinforcement learning - depending upon the nature of the data it receives, as shown in Figure 1:

- Supervised learning: a mathematical model of a set of data that contains both the inputs and the desired outputs is built. The "agent" observes some example input-output pairs and learns a function that maps from input to output (Russell & Norvig 2010). For example, a taxi agent knows the concept of "good traffic days" and "bad traffic days" by being given the labeled examples of each by a "teacher".

- Unsupervised learning: the "agent" learns patterns in the input even though no explicit feedback is supplied. The most common unsupervised learning task is clustering: detecting potentially useful clusters of input examples. For example, a taxi agent might gradually develop a concept of "good traffic days" and "bad traffic days" without ever being given labeled examples of each by a "teacher" (Russell & Norvig 2010).

- Reinforcement learning: the "agent" learns from a series of reinforcements — rewards or punishments. For example, the lack of a tip at the end of the journey gives the taxi agent an indication that it did something wrong. The two points for a win at the end of a chess game tell the agent it did something right. It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it (Russell & Norvig 2010).

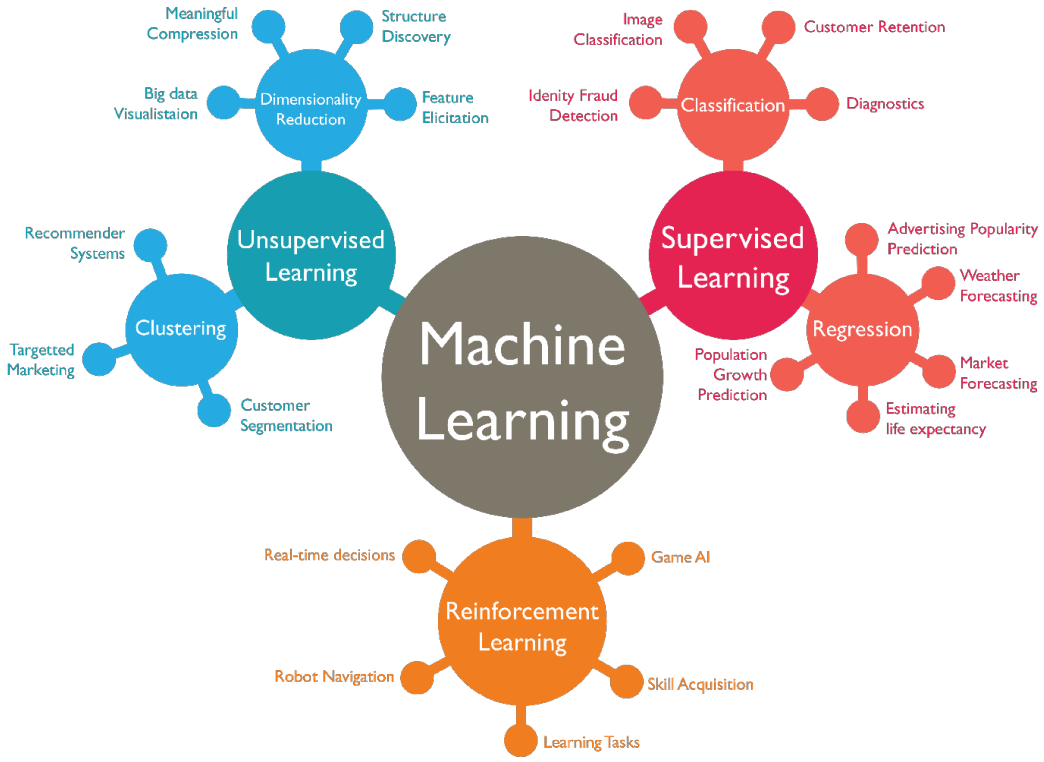


Figure 1: Types of Machine Learning (reproduced from Kundu (2018)).

There is also semi-supervised learning that falls between unsupervised and supervised learning, having a mix of labeled and unlabeled data. Many machine-learning researchers have found that unlabeled data, when used with a small amount of labeled data, can significantly improve learning accuracy.

2.2 Data Mining

There is a difference between machine learning and data mining, which often overlap significantly and use the same methods. However, while machine learning focuses on prediction based on known properties learned from training data, and data mining focuses on discovering (previously) unknown properties in the data (this is the step of evaluating the discovery of information in databases). The application of machine learning methods to large databases is called data mining. Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals) comes from the underlying assumptions they work with: in machine learning, performance is usually evaluated concerning the ability to reproduce known knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously unknown knowledge. Evaluated with respect to known knowledge, an uninformed

(unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data (Wikipedia 2019*b*).

With this, DM, as a truly interdisciplinary subject, can be defined in many different ways. To refer to gold mining from rocks or sand, it's said gold mining instead of rock or sand mining. Analogously, data mining should have been more appropriately named "knowledge mining from data", which is unfortunately somewhat long. However, in the shorter term, knowledge mining may not reflect the emphasis on mining from large amounts of data (Han, Kamber & Pei 2011). According to these authors, DM is the process of discovering interesting patterns and knowledge from large amounts of data. The data sources can include databases, data warehouses, the Web, other information repositories, or data streamed into the system dynamically.

Data mining is also commonly known as knowledge discovery from data (KDD), including the following steps:

1. Data cleaning, as removal of noisy and irrelevant data from a collection. Examples are missing values.
2. Data integration, defined as heterogeneous data from multiple data sources combined.
3. Data selection, where data relevant to the analysis, is decided and retrieved from the data collection.
4. Data transformation, defined as the process where data are transformed and consolidated into an appropriate form required by mining procedure.
5. Data mining, where clever techniques are applied to extract data patterns potentially useful.
6. Pattern evaluation, as identifying truly interesting patterns representing knowledge based on given measures.
7. Knowledge presentation, defined as the visualization and knowledge representation techniques to present the mined knowledge results to the user.

Steps 1 to 4 are different forms of data preprocessing, where data are prepared for mining. KDD should be seen as an iterative process in which measurements of evaluation can be improved, processing can be optimized, and new data can be implemented and converted to achieve specific and more acceptable results. This knowledge discovery process is illustrated in Figure 2.

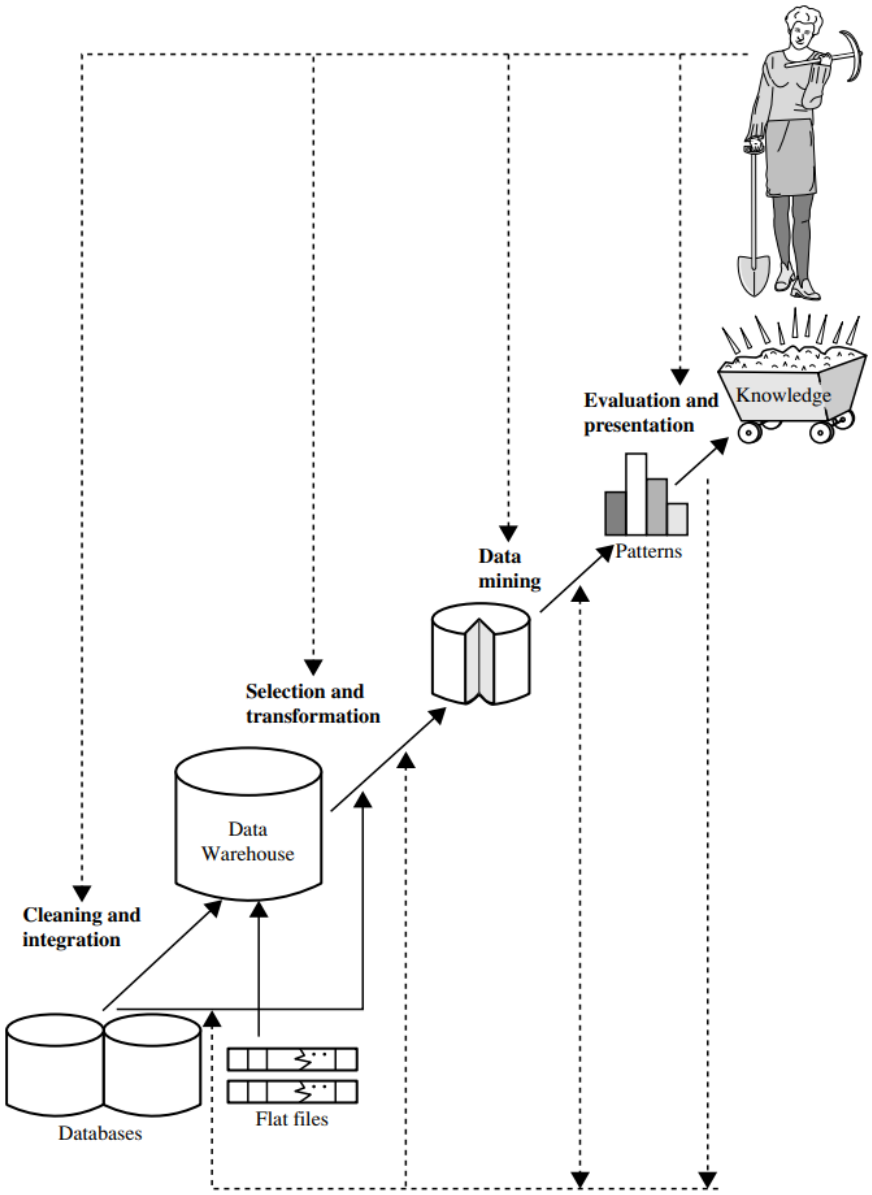


Figure 2: Data mining as a step in the process of knowledge discovery (reproduced from Han, Kamber & Pei (2011), Fig. 1.4, p.7).

There are a number of data mining tasks such as classification, time-series analysis, among others, and all these tasks are either predictive DM tasks or descriptive DM tasks. A data mining system can execute one or more of the above specified tasks as part of data mining:

- Predictive
  - Classification
  - Regression
  - Time-series analysis

- Descriptive
  - Clustering
  - Summarization
  - Association

Starting with predictive tasks, classification is learning a function that maps (classifies) a data item into one of several predefined classes (Fayyad, Piatetsky-Shapiro & Smyth 1996), while regression is learning a function that maps a data item to a real-valued prediction variable. An example of using classification methods is banks who might want to automatically decide whether future loan applicants will be given a loan. Regression applications are many, like predicting the amount of biomass present in a forest given remotely sensed microwave measurements or estimating the probability that a patient will survive given the results of the diagnostic tests. A time-series is a sequence of data points recorded at specific time points - most often in regular time intervals (seconds, hours, days, months). Outlier/anomaly detection or trend analysis are potential ways to take advantage of time-series datasets. Moving on to descriptive tasks, Fayyad, Piatetsky-Shapiro & Smyth (1996) define clustering as one seeking to identify a finite set of categories or clusters to describe the data. The categories can be mutually exclusive and exhaustive or consist of a richer representation, such as hierarchical or overlapping categories. Summarization involves methods for finding a compact description for a subset of data. A simple example would be tabulating the mean and standard deviations for all fields. For association rules, they can predict not only any attribute but also have the freedom to predict combinations of attributes (Witten, Frank & Hall 2011), as a relationship between variables. A typical and widely used example of association rules application is market basket analysis.

2.2.1 Data Mining in Telecommunications Industry

Data mining and machine learning have been used in many industries. Whether it is fraud detection, customer segmentation, market basket analysis, life, and death. From Europe to the Antipodes. Family and business. Machine learning is a burgeoning new technology for mining knowledge from data, a technology that many people are starting to take seriously (Witten, Frank & Hall 2011). The data mining applications for any industry depend on two factors: the available data and the business problems facing the industry (Weiss 2008).

The telecommunications industry was one of the first to adopt DM technologies. These have proven effective in detecting fraud, managing networks better, gaining knowledge about the customers (e.g., customer segmentation), predicting churn and retaining clients, and improving their marketing efforts, knowing what products and services yield the highest amount of profit. This is most likely because telecommunication companies routinely generate and store enormous amounts

of high-quality data, have a huge customer base, and operate in a rapidly changing and highly competitive environment (Weiss 2008). Telecommunication companies can no longer afford not to make use of their Big Data. Companies like AT&T and Sprint always top the list of the largest databases in the world, reaching storage in the order of thousands or even millions of terabytes. The huge amount of data available can also be a key concern because of the scalability of DM methods, the inappropriate semantic level (records in the form of transactions/events), and the data generated in real-time and applications that need to operate in real-time, such as fraud and network fault analysis (Weiss 2008). Because of the efforts to address these issues, the telecommunications industry has been a leader in the DM area.

2.3 Techniques to Improve Classification Accuracy

There are some tricks for increasing classification accuracy. A classifier's accuracy on a given test set is the percentage of test set tuples correctly classified by the classifier (Han et al. 2011). It is also referred to as the classifier's overall recognition rate; that is, it reflects how well the classifier recognizes tuples of the various classes. This section focuses on ensemble methods to achieve a higher classification accuracy. An ensemble for classification is a composite model, made up of a combination of classifiers. The individual classifiers vote, and a class label prediction is returned by the ensemble based on the collection of votes. Ensembles (e.g., bagging and boosting) tend to be more accurate than their component classifiers (Han et al. 2011).

Combining the decisions of different models means amalgamating the various outputs into a single prediction. The simplest way to do this in the case of classification is to take a vote (perhaps a weighted vote); in numeric prediction, it is to calculate the average (perhaps a weighted average). Bagging and boosting both adopt this approach, but they derive the individual models in different ways. In bagging, the models receive equal weight, whereas in boosting weighting is used to give more influence to the more successful ones — just as an executive might place different values on different experts' advice depending on how successful their predictions were in the past (Witten et al. 2011).

2.3.1 Bagging

In detail, bagging can be seen as a method of increasing accuracy. Han et al. (2011) make the following analogy: suppose that a patient would like to have a diagnosis made based on his/her symptoms. Instead of asking one doctor, he/she may choose to ask several. If a particular diagnosis occurs more than any other, he/she may choose this as the final or best diagnosis. That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote. Replacing

each doctor by a classifier, and this the basic idea behind bagging. Intuitively, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.

Given a set, D, of d tuples, bagging works as follows. For iteration i(i = 1, 2,..., k), a training set, D_i, of d tuples is sampled with replacement from the original set of tuples, D. Note that the term bagging stands for bootstrap aggregation, and each training set is a bootstrap sample. Because sampling with replacement is used, some of the original tuples of D may not be included in D_i, whereas others may occur more than once. A classifier model, M_i, is learned for each training set, D_i. To classify an unknown tuple, X, each classifier, M_i, returns its class prediction, which counts as one vote. The bagged classifier, M$*$, counts the votes and assigns the class with the most votes to X. Bagging can be applied to predicting continuous values by taking the average value of each prediction for a given test tuple. The bagged classifier often has significantly greater accuracy than a single classifier derived from D, the original training data. It will not be considerably worse and is more robust to the effects of noisy data and overfitting. The increased accuracy occurs because the composite model reduces the variance of the individual classifiers.

2.3.2 Boosting

Like bagging, boosting can be explained following the analogy: suppose that a patient has certain symptoms. Instead of consulting one doctor, he/she choose to consult several. Suppose the patient assigns weights to the value or worth of each doctor's diagnosis, based on the accuracies of previous diagnoses they have made. The final diagnosis is then a combination of the weighted diagnoses. This is the essence behind boosting (Han et al. 2011). In boosting, weights are also assigned to each training tuple. A series of k classifiers is iteratively learned. After a classifier, M_i, is learned, the weights are updated to allow the subsequent classifier, $M_i + 1$, to "pay more attention" to the training tuples that were misclassified by M_i. The final boosted classifier, M$*$, combines each individual classifier's votes, where the weight of each classifier's vote is a function of its accuracy. While both bagging and boosting can significantly improve accuracy compared to a single model, boosting tends to achieve higher accuracy. AdaBoost (short for Adaptive Boosting) is a popular boosting algorithm.

2.4 Models

When it comes to ML models, they are far more complex than simple methods like association rules. The principles are the same. So are the inputs and outputs — methods of knowledge representation. However, machine learning algorithms have to deal robustly and sensibly with real-world problems such as numeric attributes, missing values, and — most challenging of all — noisy data

(Witten et al. 2011). The next sections introduce some concepts and relevant ML models, explaining how they work and their features. The taxonomy of machine learning models is similar to the presented in Peter Flach's book (Flach 2012) - Logical Models (tree or rule-based), Geometric Models (linear or distance-based), and Probability models. The first group involves models that use a logical expression, the geometric group consists of using the geometry of the instance space, and the probability models use probability to classify the instance space.

2.4.1 Linear

Linear models that can be understood in terms of lines and planes. Linearity plays a fundamental role in mathematics and related disciplines, and the mathematics of linear models is well-understood and relatively simple.

To explain Logistic Regression, a model used in this work's modeling process, first linear regression will be introduced. Linear regression is a linear approach to modeling the relationship between two variables by fitting a linear equation to observed data. One variable is considered an explanatory (or independent) variable, and the other is considered a dependent variable. The best fit line is where the total error of prediction (all data points) is as small as possible. For more than one explanatory variable, the process is called multiple linear regression. This concept is distinct from multivariate linear regression, where multiple correlated dependent variables are predicted instead of a single scalar variable. What distinguishes a logistic regression model from the linear regression model is that the logistic regression outcome variable is dichotomous (binary). This difference between logistic and linear regression is reflected both in the model's form and its assumptions (Hosmer & Lemeshow 2000). Like all regression analysis, the logistic regression is a predictive analysis and is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval, or ratio-level independent variables (Solutions 2020). This algorithm is named for the function used at the core of the method: it uses the logistic function to squeeze the linear equation's output between 0 and 1. With this, the step from linear regression to logistic regression is straightforward. In the linear regression model, it is modeled the relationship between the outcome and features with a linear equation. For classification, there are probabilities between 0 and 1, so it is wrapped the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

Another linear model is support vector machines (SVM). As defined by Han et al. (2011), SVM is a method for the classification of both linear and nonlinear data. This approach is often used for a set of training examples that fall into two categories since the SVM's objective is to find the best classification function that allows the distinction of members of both classes. In a nutshell, an SVM is an algorithm that works as follows. It uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating

hyperplane $f(x)$ (i.e., a "decision boundary" separating the tuples of one class from another). With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. With this determined, a new instance x_n is classified according to the function signal $f(x)$ and x_n belongs to the positive class if $f(x_n) > 0$. The SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors).

Figure 3 shows the representation of an SVM, with evidence of hyperplanes, a margin, and a linear separation of two distinct classes. This specific example shows the concept of buying a computer, with the classes buys_computer = yes and buys_computer = no, meaning the prediction of an electronic store customer acquiring a computer.

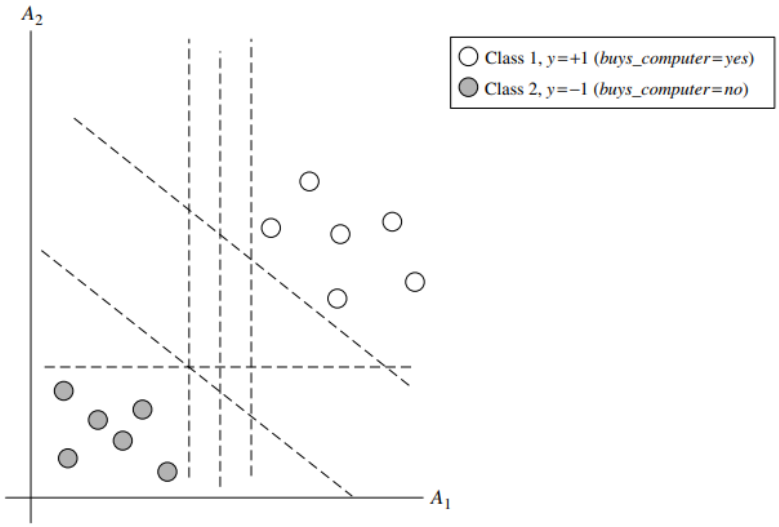


Figure 3: The 2D training data are linearly separable. There are an infinite number of possible separating hyperplanes or "decision boundaries", some of which are shown as dashed lines (reproduced from Han et al. (2011), Fig. 9.7, p.409).

When there are many hyperplanes, an SVM approaches this problem by searching for the maximum marginal hyperplane, corresponding to the shortest distance between a set of points closer to each other and a certain point on the hyperplane. Although there is an infinite number of hyperplanes, only one is constituted as the solution for the SVM in question. Furthermore, given this method, high precision levels are evident, but SVMs are also very slow when processing large datasets and do not perform very well when the data set has more noise, i.e., target classes overlap. On the other hand, SVM is relatively memory efficient is more effective in high dimensional spaces. A usual solution is to change the kernel function, such as the linear one, to transform data into another dimension where a more natural separation can be achieved for the data, maintaining reasonable performance.

2.4.2 Probabilistic

In this section, probabilistic models are introduced as the idea of probability to classify new entities. Probabilistic models see features and target variables as random variables. The process of modeling represents and manipulates the level of uncertainty concerning these variables. In machine learning, Naïve Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (Naïve) independence assumptions between the features. They are among the simplest Bayesian network models. As cited by Rish (2001), Bayesian classifiers assign the most likely class to a given example described by its feature vector. Learning such classifiers can be greatly simplified by assuming that features are independent given class, that is, $P(X|C) = \prod_{i=1}^{n} P(X_i|C)$, where $X = (X_1, ..., X_n)$ is a feature vector and C is a class. Despite this unrealistic assumption, the resulting classifier known as naive Bayes is remarkably successful in practice, often competing with sophisticated techniques. The success of Naïve Bayes in the presence of feature dependencies can be explained as follows: optimality in terms of zero-one loss (classification error) is not necessarily related to the quality of the fit to a probability distribution (i.e., the appropriateness of the independence assumption). Instead, an optimal classifier is obtained as long as both the actual and estimated distributions agree on the most-probable class (Rish 2001).

Classical Naïve Bayes supports categorical features and models, each as conforming to a Multinomial Distribution. On the other hand, Gaussian Naive Bayes supports continuous-valued features and models, each conforming to a Gaussian (normal) distribution. So, the classical Naïve Bayes classifier is appropriate when the features of the dataset are all categorical. When the features are all continuous, the Gaussian Naive Bayes classifier is appropriate. It is also assumed that all the features are following a Gaussian distribution, i.e., normal distribution.

2.4.3 Tree-Based

Tree models are among the most popular models in machine learning. According to Han et al. (2011), decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node. A typical decision tree is shown in Figure 4. It represents the concept buys computer; that is, it predicts whether a customer at a company is likely to purchase a computer. Internal nodes are denoted by rectangles, and ovals denote leaf nodes. Some decision tree algorithms produce only binary trees (where each internal node branches to exactly two other nodes), whereas others can produce nonbinary trees.

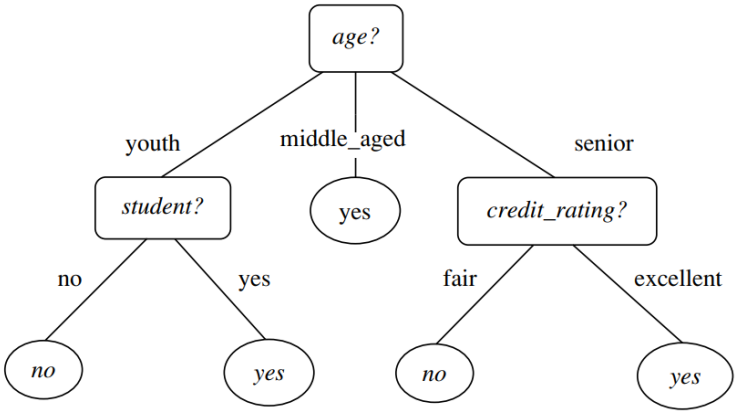


Figure 4: A decision tree for the concept buys computer, indicating whether a customer is likely to purchase a computer (reproduced from Han et al. (2011), Fig. 8.2, p.331).

Decision trees can be of two types - classification trees or regression trees - and the distinction arises from the type of the predicted outcome, that can be the class (discrete) to which the data belongs, or a real number (e.g., the price of a car). There are many specific decision-tree algorithms. Differences in decision tree algorithms include how the attributes are selected in creating the tree (variance in regression trees, and entropy or Gini impurity for classification trees) and the mechanisms used for pruning (a technique for reducing the size of the tree, that, in its turn, reduces the complexity of the final classifier, and hence improves predictive accuracy by the reduction of overfitting).

Decision trees have numerous advantages. They are simple to understand and require little preprocessing of the data. Besides performing well with large datasets, decision trees handle both numerical and categorical data. On the other hand, they can become an NP-complete problem when learning an optimal decision tree, and they are known for being very non-robust, as a small change in training data will lead to a significant change in the tree and, therefore, the final predictions. Another limitation of decision trees is that they are biased in favor of attributes with more levels. Fortunately, this can be avoided using approaches like conditional inference.

Over time, tree-based algorithms have evolved into more complex models. From the usage of simple decision trees to the utilization of ensemble methods, such as bagging and boosting, several efforts have been made to improve the model's performance and computational resources. Random forests (RF) and gradient boosting (GB) are more elaborate ensembles. Both typically use decision trees as their base estimator. For RF, the individual decision trees are generated using a random selection of attributes at each node to determine the split. More formally, each tree depends on a random vector's values sampled independently and with the same distribution for all trees in the forest. During classification, each tree votes, and the most popular class is returned (Han et al. 2011). Random forests can be built using bagging or random linear combinations. As for performance, random forests (RF) are comparable in accuracy to AdaBoost, yet are more robust to

errors and outliers. The generalization error for a forest converges as long as the number of trees in the forest is large. Thus, overfitting is not a problem. The accuracy of a random forest depends on the individual classifiers' strength and a measure of the dependence between them. The ideal is to maintain the strength of individual classifiers without increasing their correlation. Random forests are insensitive to the number of attributes selected for consideration at each split. Because RF considers many fewer attributes for each split, they are efficient on very large databases. They can be faster than either bagging or boosting, and give internal estimates of variable importance (Han et al. 2011).

GB also appeared to change the paradigm. Leo Breiman originally stated that boosting could be interpreted as an optimization algorithm on a suitable cost function (Breiman 1997). Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees - Gradient Boosting Tree. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error. Gradient boosting involves three elements: a loss function to be optimized, a weak learner to make predictions, and an additive model to add weak learners to minimize the loss function. The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:

- If a small change in the prediction for a case causes a large drop in error, then the next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
- If a small change in the prediction for a case causes no change in error, then the next target outcome of the case is zero. Changing this prediction does not decrease the error.

The name gradient boosting arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case (Hoare 2020).

After gradient boosting trees, Tianqi Chen proposed a scalable machine learning system for tree boosting called XGBoost (Chen & Guestrin 2016). This system's impact has been widely recognized in several machine learning and data mining challenges. Examples of the problems in these winning solutions include store sales prediction, high energy physics event classification, web text classification, customer behavior prediction, motion detection, ad click-through rate prediction, malware classification, product categorization, hazard risk prediction, and massive online course dropout rate

prediction. The most crucial factor behind the success of XGBoost is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. The scalability of XGBoost is due to several important systems and algorithmic optimizations. These innovations include: a novel tree learning algorithm for handling sparse data and a theoretically justified weighted quantile sketch procedure that enables handling instance weights in approximate tree learning. Parallel and distributed computing make learning faster, which enables quicker model exploration. More importantly, XGBoost exploits cache awareness and out-of-core computation and enables data scientists to process hundreds of millions of examples on a desktop. Finally, it is even more exciting to combine these techniques to make an end-to-end system that scales to even larger data with the least amount of cluster resources (Chen & Guestrin 2016).

The efficiency and scalability of XGBoost, although better than its previous models, was still unsatisfactory when the feature dimension is high and data size is large. A major reason is that it needs to scan all the data instances for each feature to estimate the information gain of all possible split points, which is very time-consuming. To tackle this problem, (Ke et al. 2017) proposed two novel techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB). With GOSS, they excluded a significant proportion of data instances with small gradients, and only use the rest to estimate the information gain. The authors proved that, since the data instances with larger gradients play a more important role in the computation of information gain, GOSS can obtain quite an accurate estimation of the information gain with much smaller data size. With EFB, they bundle mutually exclusive features (i.e., they rarely take nonzero values simultaneously) to reduce the number of features. It was proved that finding the optimal bundling of exclusive features is NP-hard, but a greedy algorithm can achieve a quite good approximation ratio (and thus can effectively reduce the number of features without hurting the accuracy of split point determination by much). The authors called their new GBDT implementation, with GOSS and EFB, LightGBM. This ML technique speeds up the training process of conventional GBDT by up to over 20 times while achieving almost the same accuracy. Summing up, the main advantages of LightGBM, when compared to XGBoost, are (Vidhya 2017):

- Faster training speed and higher efficiency: LightGBM uses a histogram-based algorithm, i.e., it buckets continuous feature values into discrete bins that fasten the training procedure.
- Lower memory usage: it replaces continuous values to discrete bins, which result in lower memory usage.
- Better accuracy than any other boosting algorithm: it produces much more complex trees by following leaf wise split approach rather than a level-wise approach, which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting that can be avoided by setting the max_depth parameter.

- Compatibility with large datasets: it can perform equally well with large datasets with a significant reduction in training time compared to XGBoost.
- Parallel learning supported.

The entire evolution of models found throughout this section, from simple decision trees to LightGBM, is summarized in the image below (Figure 5):

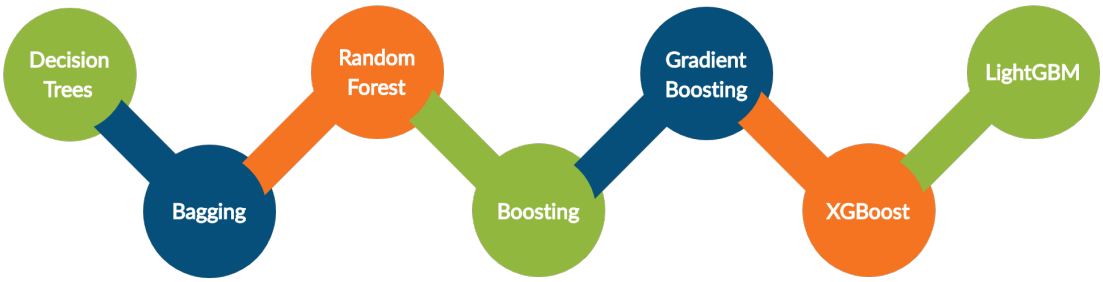


Figure 5: Evolution of LightGBM algorithm from Decision Trees.

2.4.4 Hyperparameter Optimization

Machine learning methods attempt to build models that capture some element of interest, based on given data. Most common learning algorithms feature a set of hyperparameters that must be determined before training commences. From biologically inspired neural networks over kernel methods to ensemble models, a common trait in these methods is that they are parameterized by a set of hyperparameters, which must be set appropriately by the user to maximize the usefulness of the learning approach. The choice of hyperparameters can significantly affect the resulting model's performance, but determining good values can be complex; hence a disciplined, theoretically sound search strategy is essential. Hyperparameter search is commonly performed manually, via rules-of-thumb or by testing sets of hyperparameters on a predefined grid (Claesen & Moor 2015).

Manual search, grid search, and random search are the most widely used strategies for hyperparameter optimization (or tuning). The first technique was one of the first to be used and consists of tuning hyperparameters manually by trial and error. This is still commonly done, and students, experienced engineers, and data scientists can "guess" parameter values that will culminate in very high accuracy for machine learning models. However, there is a constant search for a better and faster strategy to optimize hyperparameters automatically. Since the critical step in hyperparameter optimization is to choose the set of trials, grid search becomes useful and eases this process.

Grid search requires the choice of a set of values for each variable (or parameter). In grid search, the set of trials is formed by assembling every possible combination of values. The experiments allocate too many trials to explore dimensions that do not matter, so the number of trials in a

grid search suffers from the curse of dimensionality because the number of joint values grows exponentially with the number of hyperparameters. With this, despite automating the tuning process, grid search alone does very poorly in practice (Bergstra & Bengio 2012).

Bergstra & Bengio (2012) proposed random search as a substitute and baseline that is both reasonably efficient (roughly equivalent to or better than combining manual search an grid search, in their experiments) and keeping the advantages of implementation simplicity and reproducibility of pure grid search. The only real difference between grid search and random search is on the first step of the strategy cycle – random search picks the points randomly from the configuration space. The analysis of Bergstra & Bengio (2012) about the hyper-parameter response surface suggests that random experiments are more efficient because not all hyperparameters are equally important to tune, and random search found better models in most cases and required less computational time.

Figure 6 illustrates how point grids and uniformly random point sets differ in how they cope with low effective dimensionality. A grid of points gives even coverage in the original 2-d space, but projections onto either the x_1 or x_2 subspace produce an inefficient coverage of the subspace. In contrast, random points are slightly less evenly distributed in the original space, but far more evenly distributed in the subspaces.

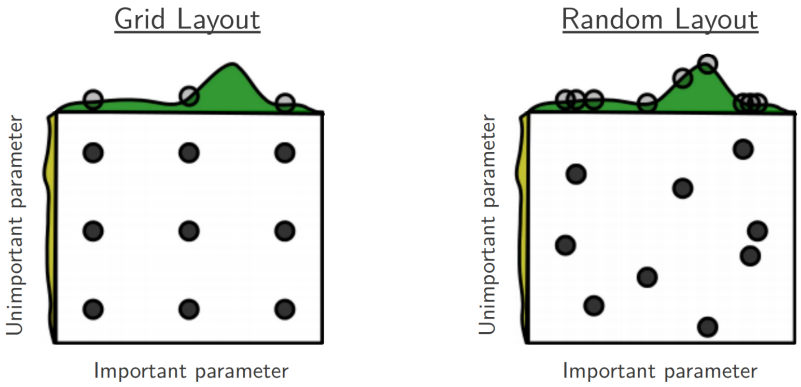


Figure 6: Grid and random search of nine trials for optimizing a function $f(x, y) = g(x) + h(y) \approx g(x)$ with low effective dimensionality. Above each square $g(x)$ is shown in green, and left of each square $h(y)$ is shown in yellow. With grid search, nine trials only test $g(x)$ in three distinct places. With random search, all nine trials explore distinct values of g. This failure of grid search is the rule rather than the exception in high dimensional hyperparameter optimization (reproduced from Bergstra & Bengio (2012)).

In summary, as random search thrives on low effective dimensionality, the use of grid search is appropriate for these situations, but not recommended for searching spaces containing more than 3 to 4 dimensions.

2.4.5 Performance Metrics and Model Evaluation

Evaluating a model is a core part of building a useful machine learning model, and the performance metrics are used to find out how effective the model is. Different performance metrics are used to evaluate different machine learning algorithms, and their selection influences how the performance of ML algorithms is measured and compared. The next subsections present different metrics and how they are used to measure quality.

In addition to accuracy-based measures, classifiers can also be compared concerning the following additional aspects (Han et al. 2011):

- Speed: refers to the computational costs involved in generating and using the given classifier.
- Robustness: this is the classifier's ability to make correct predictions given noisy data or missing values. Robustness is typically assessed with a series of synthetic data sets representing increasing degrees of noise and missing values.
- Scalability: concerns the ability to efficiently construct the classifier, given large amounts of data. Scalability is typically assessed with a series of data sets of increasing size.
- Interpretability: this refers to the level of understanding and insight that is provided by the classifier or predictor. Interpretability is subjective and, therefore, more difficult to assess. Decision trees and classification rules can be easy to interpret, yet their interpretability may diminish the more they become complex.

Herewith, this section covers all the metrics used to evaluate models or mentioned in this thesis.

Confusion matrix

A confusion matrix is an N x N matrix that shows the predicted and the actual classification (Kohavi & Provost 1998). Simplifying, it shows how a classification model is confused when it makes predictions, giving insights not only into the errors being made by a classifier but also the types of errors that are made. It is used for classification problems where the output can be of two or more types of classes. Table 1 shows a possible confusion matrix schema.

		Predicted class: P	Predicted class: N
Actual class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Table 1: Confusion matrix schema

In the above schema, P stands for positive, while N represents negative. TP means the number of correct predictions for positive instances, and FN corresponds to the number of incorrect predictions for a positive instance. FP stands for the number of incorrect predictions for a negative value, and, finally, TN is the number of correct predictions for a negative instance. With this matrix alone, it is possible to extract relations, resulting in other metrics explained next.

Accuracy

The accuracy of a classifier on a given test set is the percentage of test set tuples (e.g., buys computer = yes) that are correctly classified by the classifier (Han et al. 2011). That is,

$$accuracy = \frac{TP + TN}{P + N}$$

In the pattern recognition literature, this is also referred to as the overall recognition rate of the classifier; that is, it reflects how well the classifier recognizes tuples of the various classes. By the contrast, there is also an error rate or misclassification rate of a classifier M, which is simply 1 – accuracy(M), where accuracy(M) is the accuracy of M. This also can be computed as

$$error\ rate = \frac{FP + FN}{P + N}$$

Precision

According to Han et al. (2011), precision can be thought of as a measure of exactness (i.e., what percentage of tuples labeled as positive are actually such). High precision indicates an example labeled as positive is indeed positive (a small number of FP). Precision is given by the relation:

$$precision = \frac{TP}{TP + FP}$$

Recall or Sensitivity

While precision is a measure of exactness, recall is a measure of completeness (what percentage of positive tuples are labeled as such). If recall seems familiar, that is because it is the same as sensitivity (or the true positive rate). High Recall indicates the class is correctly recognized (a small number of FN). This measures can be computed as

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}$$

False Positive Rate

False positive rate, as the name says, quantifies the proportion of negative tuples that were incorrectly classified as positives. This is given by the relation:

$$false\,positive\,rate = \frac{FP}{FP + TN}$$

False Negative Rate

False negative rate quantifies the proportion of positive tuples that were incorrectly classified as negatives. That is,

$$false\,negative\,rate = \frac{FN}{TP + FN} = 1 - sensitivity$$

ROC Curve

This is one of the most popular metrics in the industry. A ROC (receiver operating characteristic) curve for a given model shows the trade-off between the true positive and false positive rate (Han et al. 2011). The most significant advantage of using the ROC curve is that it is independent of the change in the proportion of responders. That is, for each sensitivity, there is a different specificity, and as one rises, the other decreases. The ROC curve is the plot between sensitivity and (1-specificity). (1- specificity) is the false positive rate, and sensitivity is the true positive rate.

Figure 7 shows a ROC curve showing two tests. The red test is closer to the diagonal and is, therefore, less accurate than the blue test.

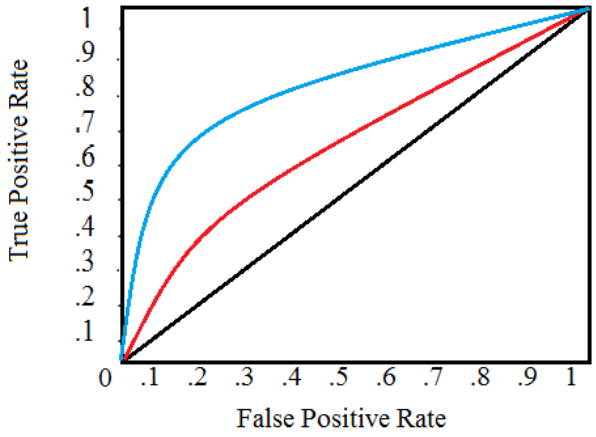


Figure 7: ROC curve example (reproduced from Statistics How To (2019)).

Train-Test-Validation Split

Machine learning algorithms build a mathematical model from input data. Typically, the data used to construct the final model is originated from different datasets. In particular, three datasets are commonly used during the stages of model creation: training, validation, and test. Brownlee (2017) provides a good explanation of the three-way data splits. According to him, a training dataset is a sample of data used to fit the model; that is, the model is run with the training dataset and produces a result. A validation dataset is a sample of data used to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters. The evaluation becomes more biased as the skill on the validation dataset is incorporated into the model configuration. After choosing the best model, the test dataset is used to provide an unbiased evaluation of a final model fit on the training dataset, based on various metrics. Another option is to fit the final model on the aggregate of the training and validation datasets. Figure 8 is a visualization of each of the splits.

Figure 8: Slicing a dataset into a training, validation and test set.

There is not an ideal ratio for the division of these datasets. Depending on the size of the data, the most common ratios used are: (1) 70% train, 15% val, 15% test, (2) 80% train, 10% val, 10% test, or (3) 60% train, 20% val, 20% test.

2.5 Imbalanced Datasets

When working on classification problems, data scientists may encounter data with imbalanced classes. An imbalanced dataset occurs when there is an unequal representation of classes. The issue with the imbalance in the class distribution became more pronounced with the applications of machine learning algorithms to the real world. One focus of the initial works in this area was primarily the performance evaluation criteria for mining imbalanced datasets (Chawla 2010). The limitation of the accuracy as the performance measure was quickly established, and ROC curves soon emerged as a popular choice. Also, the imbalance in the data can be more characteristic of "sparseness" in feature space than the class imbalance. Various re-sampling strategies have been used, such as random oversampling with replacement, random undersampling, focused oversampling, focused undersampling, oversampling with a synthetic generation of new samples based on the known information, and combinations of the above techniques. In the next sections, some approaches and methods are introduced in order to tackle the imbalance problem.

2.5.1 Scoring Metrics and Threshold-Optimization

Many real-world data mining applications involve obtaining predictive models using data sets with strongly imbalanced distributions of the target variable. Frequently, the least common values of this target variable are associated with highly relevant events for end-users (e.g., fraud detection, unusual returns on stock markets, or the anticipation of catastrophes). Moreover, the events may have different costs and benefits, which, when associated with the rarity of some of them on the available training data, creates severe problems in predictive modeling techniques. In these cases, where the specific sub-ranges of the target variable are poorly represented in the available training sample, there is the need of (1) special purpose evaluation metrics that are biased towards the performance of the models on these rare cases, and, moreover, it is needed means for (2) making the learning algorithms focus on these rare events. Without addressing these two questions, models will tend to be biased to the most frequent (and uninteresting for the user) cases, and the results of the "standard" evaluation metrics will not capture the competence of the models on these rare cases (Branco et al. 2016).

Because most of the standard metrics that are widely used assume a balanced class distribution, and because typically not all classes, and therefore, not all prediction errors, are equal for imbalanced classification, it is usually considered the following metrics for evaluation: AUPRC (Area Under Precision-Recall Curve), AUROC (Area Under the Receiver Operating Characteristic), Precision and Recall. A precision-recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds, much like the ROC curve. Two examples of these curves are shown in Figure 9.

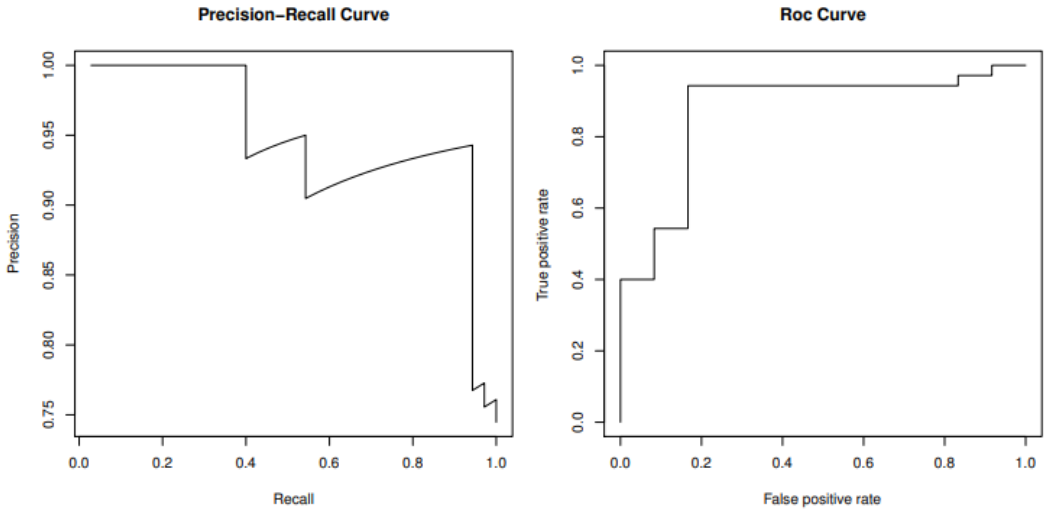


Figure 9: Examples of precision-recall curve and ROC curve (reproduced from Branco et al. (2016)). The AUPRC and AUROC are the area under the precision-recall curve and the roc curve, respectively.

The precision-recall curve can focus only on the class of interest, whereas the ROC curve covers both classes, and that is the main reason that one of the most common metrics for these situations is AUPRC. The well-known accuracy is not suitable because the impact of the least represented, but more crucial examples is reduced compared to that of the majority class (Branco et al. 2016).

Normally, in this type of study, the results can be applied in a real context in the company. So, the trade-off between precision and recall is commonly analyzed, adapting the threshold as needed (Fawcett 2006): a higher recall or a greater precision. A higher recall makes it possible to avoid false negatives, which, in a business context, would mean picking up more possible cases of having an installation problem, requiring a second screening action on the part of the company to verify the true positives of this population. On the other hand, a greater precision means that it only predicts an installation problem if the results are confident; that is, the customers selected by the model for candidates for installation problems are almost certain. The classification model threshold, by default 0.5 (Fawcett 2006), can be lowered or increased to achieve the company's objectives. This threshold-moving approach to the class imbalance problem applies to classifiers that, given an input tuple, return a continuous output value. That is, for an input tuple, X, such a classifier returns as output a mapping, $f(X) \rightarrow [0, 1]$. Rather than manipulating the training tuples, this method returns a classification decision based on the output values. In the simplest approach, tuples for which $f(x) \geq t$, for some threshold, t, are considered positive, while all other tuples are considered negative (Han et al. 2011).

2.5.2 Balancing Techniques

In addition to techniques such as threshold-moving or the choice of evaluation metrics, there are other approaches to improve the classification accuracy of class-imbalanced data. In a data-level approach, the researchers have tried to balance the datasets before applying traditional classification algorithms so that results may not be impacted by the majority class. This method includes oversampling, undersampling, and hybrid sampling. According to Kaur & Gosain (2018), oversampling is the process of increasing the number of instances into the minority class either randomly through replication of the same data or generating synthetically by using some technique to improve the imbalance ratio so that same classification algorithms can be used to classify the data. The advantage of this technique is that there is no loss of any important information from the dataset, and the original dataset is retained, although new information is added to it to balance the data. The limitation of oversampling is that it takes more time to execute than the undersampling approach, as there is an increase in the number of instances. It may also cause the problem of overfitting because it replicates the same instances.

In the undersampling approach, the working area in the dataset is majority class, wherein the instances from the majority class are removed either randomly or by using some technique to balance the classes, and then traditional classification algorithms are applied to classify the data. Hybrid sampling is the combination of oversampling and undersampling approaches, where both techniques, i.e., oversampling and undersampling, are applied on the unbalanced data to improve the imbalance ratio, and then classes are identified using traditional classification algorithms. The main advantage of this technique is that it is a fast and simple approach compared to oversampling. Nevertheless, as the instances are removed from the majority class, it may lead to the loss of some potentially useful information contained in the majority class (Kaur & Gosain 2018). Figure 10 outlines the oversampling and undersampling techniques.

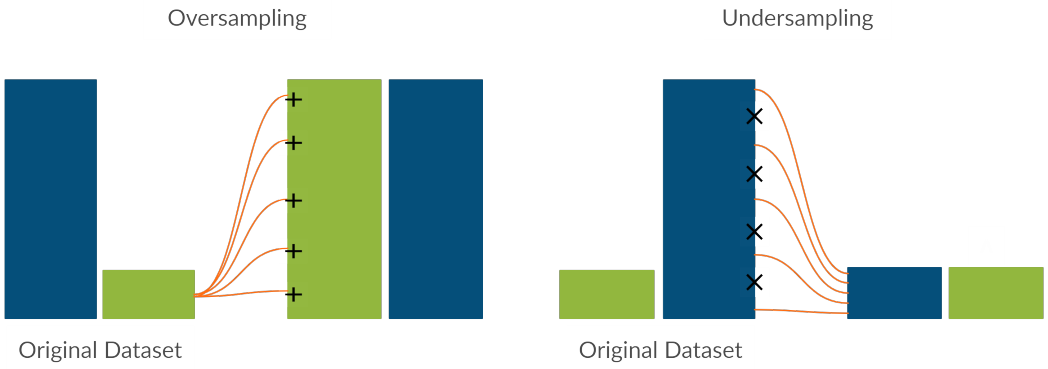


Figure 10: Oversampling: copies of the minority class versus Undersampling: Samples of majority class.

Synthetic Minority Oversampling (SMOTE)

In 2002, Chawla et al. (2002) proposed an over-sampling approach in which the minority class is over-sampled by creating "synthetic" examples rather than by over-sampling with replacement. They generate synthetic examples by operating in "feature space" rather than "data space". The minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. Synthetic samples are basically generated by taking the difference between the feature vector (sample) under consideration and its nearest neighbor first. Then, this difference is multiplied by a random number between 0 and 1, and the result is added to the feature vector under consideration. This causes the selection of a random point along the line segment between two specific features. This approach effectively forces the decision region of the minority class to become more general. The global steps of the algorithm are as follows (Kaur & Gosain 2018):

1. Load dataset and identify the minority and majority class.
2. Calculate the number of instances to be generated based on the percentage of oversampling.
3. Identify a random instance from minority class and find its nearest neighbors.
4. Select any nearest neighbors and find the difference between random instance and its selected nearest neighbor.
5. Multiply the difference by a random number generated between 0 and 1.
6. Add this difference to the selected random instance.
7. Repeat the process from 3 to 6 until the number of instances is generated as per the given percentage.

Random Oversampling (ROS) and Random Undersampling (RUS)

RUS is a straightforward undersampling approach that randomly removes instances from the majority class to balance the dataset before applying a classification technique. The concept of RUS is very simple, and it is fast as compared to SMOTE. The only limitation with this technique is that it can remove the important information contained in the majority class, which may not be acceptable in some cases. It can be illustrated in Figure 11.

In the figure, the green line shows the ideal decision boundary that it is wanted to be identified, and the blue line shows the actual results obtained. The left side of the figure shows the result of classification after applying a general machine learning algorithm without undersampling, and

the right section shows the result after applying undersampling. By applying undersampling on the majority class, some informative majority class information is removed. It caused the blue decision boundary to be slanted, inducing some majority class instances to be classified as minority class instances wrongly (Kaur & Gosain 2018).

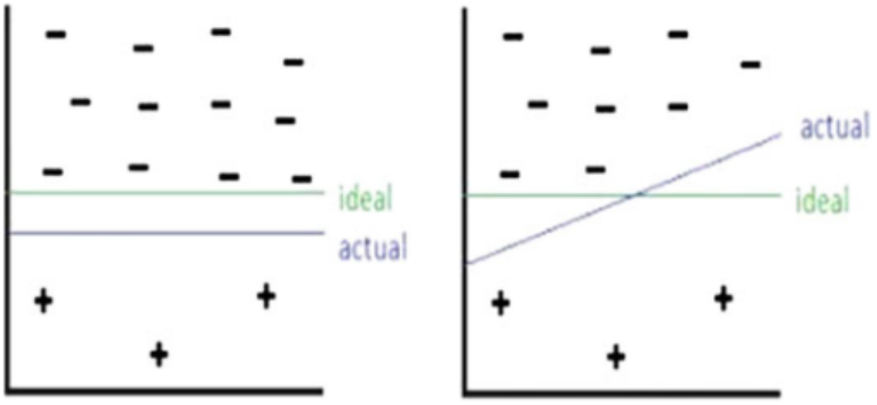


Figure 11: Classification with and without using undersampling (reproduced from Kaur & Gosain (2018)).

On the other hand, ROS is a simple approach that randomly duplicates examples in the minority class to balance the dataset before applying a classification technique. This technique's main advantage is that there is no loss of any vital information from the dataset. Unlike RUS, the original dataset is retained, although new information is added to balance the data. As already mentioned, possible limitations include this technique taking more time to execute than the undersampling approach as there is an increase in the number of instances. It may also cause the problem of overfitting as it replicates the same instances.

Both techniques include a sampling strategy, that is, sampling information to resample the dataset. In RUS's case, the defined float (e.g., 0.4) corresponds to the desired ratio of the number of samples in the minority class over the number of samples in the majority class after resampling. Therefore, the ratio is expressed as $\alpha_{us} = N_m/N_{rM}$ where N_m is the number of samples in the minority class and N_{rM} is the number of samples in the majority class after resampling (Lemaitre et al. 2017*b*). For Random Oversampling, the chosen float number corresponds to the desired ratio of the number of samples to be in the majority class after resampling. Therefore, the ratio is expressed as $\alpha_{os} = N_{rm}/N_M$ where N_{rm} is the number of samples in the minority class after resampling and N_M is the number of samples in the majority class (Lemaitre et al. 2017*a*).This sampling strategy means that the number of samples for each class in the final dataset does not necessarily need to be equal. In some cases, it is reasonable to leave some imbalance in the dataset to mirror the difference in the classes, so ratios like 40%-60% are sufficient to balance the data set and improve the classification task.

Class Weights

Another method used to fight class imbalance is class weights. Many of the predictive models for classification can pay more attention to the underrepresented class by passing weights for each class. Without resampling the data, this method allows the classifier to learn equally from all classes. This can be achieved by setting the class_weight hyperparameter to 'balanced' in many classifiers, and the algorithm automatically finds the weights for each class. The class_weight parameter can also include the definition of explicit weights for each class, chosen by the data scientist, as shown below. This forces the algorithm to treat every instance of class 1 as 20 instances of class 0, meaning that it is assigned to the loss function a higher value to these instances.

$$class\_weight = \{0 : 1, 1 : 20, 2 : 5\}$$

Being a hyperparameter of models, instead of being defined manually, it can be included in a Gridsearch or Randomsearch with different combinations of weights. There can be other combinations of weights that can perform equally well or maybe better. These optimization methods are detailed in section 2.4.4.

2.6 Data Preprocessing

Data gathered in data sets can present multiple forms and come from many different sources. Data directly extracted from relational databases or obtained from the real world is entirely raw: it has not been transformed, cleansed, or changed at all. Therefore, it may contain errors due to wrong data entry procedures or missing data or inconsistencies due to ill-handled merging data processes. Three elements define data quality: accuracy, completeness, and consistency (García et al. 2015).

Low-quality data will lead to low-quality mining results. *"How can the data be preprocessed in order to help improve the quality of the data and, consequently, of the mining results? How can the data be preprocessed so as to improve the efficiency and ease of the mining process?"* There are several data preprocessing techniques. Data cleaning can be applied to remove noise and correct inconsistencies in data. Data integration merges data from multiple sources into a coherent data store such as a data warehouse. Data reduction can reduce the data size by, for instance, aggregating, eliminating redundant features, or clustering. Data transformations (e.g., normalization) may be applied, where data are scaled to fall within a smaller range like 0.0 to 1.0. This can improve the accuracy and efficiency of mining algorithms involving distance measurements. These techniques are not mutually exclusive; they may work together. For example, data cleaning can involve transfor-

mations to correct wrong data, such as transforming all entries for a date field to a common format (Han et al. 2011).

The next sections explain, with more detail, the standard tasks of data preprocessing used in this master thesis.

2.6.1 Data Integration

Data integration is one of the first tasks of data processing. Often, the data needed for a machine learning project is found in multiple sources, which is why data integration plays a key role. This task includes gathering data from several sources in a single dataset for further cleaning and processing. If the integration process is not properly performed, redundancies and anomalies will appear, resulting in a reduction in the accuracy and speed of the subsequent DM process.

2.6.2 Data Cleaning

Real-world data tend to be incomplete, noisy, and inconsistent. Data cleaning (or data cleansing) routines attempt to fill in missing values, smooth out noise while identifying outliers, and correct inconsistencies in the data (Han et al. 2011). A proper data cleaning can make or break a project, and data scientists usually spend most of their time on this step. In this section, many basic methods for incomplete, noisy, or inconsistent data will be analyzed in order to facilitate the analytical process, uncovering reliable answers, and provide more accurate results.

A missing value can occur for many reasons: data not available or not applicable, a certain event did not happen, or a person responsible for manually inserting the data missed filling in. In these cases, either the missing value is ignored, removed, or there is an action to fill or give meaning to it. According to Han et al. (2011), some common practices are:

- Using a global constant to fill in the missing value: all missing attribute values are replaced by the same constant such as a label like “Unknown” or “-1”.
- Using a measure of central tendency for the attribute: the mean or median are used to fill in the missing value.
- Using the most probable value to fill in the missing value: this may be determined with regression, inference-based tools using a Bayesian formalism or decision tree induction.

It is important to note that, in some cases, a missing value may not imply an error in the data. For example, when applying for a credit card, candidates may be asked to supply their driver’s license number. Candidates who do not have a driver’s license may naturally leave this field blank.

When it comes to noisy or inconsistent data, there are more advanced methods and some simple ones. Noise is a random error or variance in a measured variable, and inconsistent data exists once completely different and conflicting versions of identical information seem in numerous places. Some examples are duplicates, redundancy, or absurd values. If a record is duplicated, the most common action is the removal of one of them. In the case of redundancy, these cases can be properly identified and corrected. For absurd values (or noise), besides the simple removal, techniques like outlier analysis or regression are frequently used.

2.6.3 Data Transformation

Data transformation usually combines the original raw attributes using different mathematical formulas originated in business models or pure mathematical formulas García et al. (2015). This section presents methods of data transformation. In this preprocessing step, the data are transformed or consolidated so that the resulting mining process may be more efficient, and the patterns found may be easier to understand Han et al. (2011). Data discretization, a form of data transformation, is also discussed.

Encoding and Normalization

In general, learning algorithms benefit from normalization and encoding of the data set. Before beginning to understand these data transformation operations, first, it is necessary to clarify the existing data types, since different operations are applied according to each type of data. Therefore, Han et al. (2011) organizes attributes into nominal, binary, ordinal, and numeric types:

- Nominal: means “relating to names”. The values of a nominal attribute are symbols or names of things. Each value represents some type of category, code, or state, and so nominal attributes are also referred to as categorical. The values do not have any meaningful order.
- Binary: is a nominal attribute with only two categories or states - 0 or 1, where 0 typically means that the attribute is absent, and 1 means that it is present. Binary attributes are referred to as Boolean if the two states correspond to true and false.
- Ordinal: is an attribute with possible values that have a meaningful order or ranking among them, but the magnitude between successive values is not known.
- Numerical: is quantitative; that is, it is a measurable quantity, represented in integer or real values. Numeric attributes can be interval-scaled or ratio-scaled.
  - Interval-Scaled: is measured on a scale of equal-size units. The values of interval-scaled attributes have order and can be positive, 0, or negative. Thus, in addition

to providing a ranking of values, such attributes allow to compare and quantify the difference between values.

- Ratio-Scaled: is a numeric attribute with an inherent zero-point. Besides, the values are ordered, and it is possible to compute the difference between values, as well as the mean, median, and mode.

Once the types are not mutually exclusive, classification algorithms developed from the field of machine learning often talk of attributes as being either discrete or continuous. A discrete attribute has a finite or countably infinite set of values, which may or may not be represented as integers. If an attribute is not discrete, it is continuous.

With this, two common data transformation steps are encoding and normalization. The next sections detail and organize these concepts.

Normalization

Concerning the numerical variables, the measurement unit used can affect the data analysis. For example, changing measurement units from meters to inches for height, or from kilograms to pounds for weight, may lead to very different results. In general, expressing an attribute in smaller units will lead to a larger range for that attribute, and thus tend to give such an attribute greater effect or "weight". To help avoid dependence on the choice of measurement units, the data should be normalized or standardized. This involves transforming the data to fall within a smaller or common range such as [−1,1] or [0.0, 1.0]. (The terms standardize and normalize are used interchangeably in data preprocessing, although, in statistics, the latter term also has other connotations.) Normalizing the data attempts to give all attributes an equal weight and fall between a defined interval. Normalization is particularly useful for classification algorithms involving neural networks or distance measurements such as nearest-neighbor classification and clustering. Standardization typically means rescaling data to have a mean of 0 and a standard deviation of 1 (unit variance).

With this, a common method used for normalization is Min-Max. Min-max normalization performs a linear transformation on the original data. Suppose that min_A and max_A are the minimum and maximum values of an attribute, A. Min-max normalization maps a value, v_i , of A to v_i' in the range [new_min_A,new_max_A] by computing

$$v_i' = \frac{v_i - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Min-max normalization preserves the relationships among the original data values. It will encounter an "out-of-bounds" error if a future input case for normalization falls outside of the original

data range for A. Since all the values fall between an interval, this method guarantees the same scale, but it does not deal well with outliers. The MinMaxScaler function from Scikit-Learn (2020*a*) transforms features by scaling each feature to a given range. This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. Figure 12 shows the effect of this scaler in a given dataset.

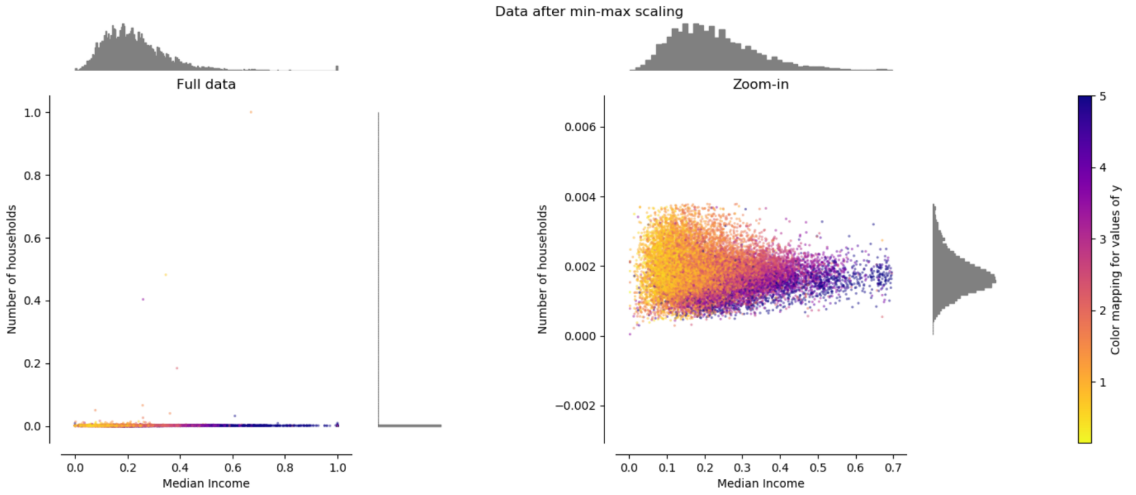


Figure 12: MinMaxScaler rescales the data set such that all feature values are between 0 and 1 as shown in the right panel. However, this scaling compress all inliers in the narrow range from 0 to 0.005, for the transformed number of households (reproduced from Scikit-Learn (2020*a*)).

In z-score normalization (or standardization, or zero-mean normalization), the values for an attribute, A, are normalized based on the mean (i.e., average) and standard deviation of A. A value, v_i, of A is normalized to v_i' by computing

$$v_i' = \frac{v_i - \overline{A}}{\sigma_A}$$

where $\overline{A}$ and σ_A are the mean and standard deviation, respectively, of attribute A. $\overline{A} = \frac{1}{n}(v_1 + v_2 + ... + v_n)$ and σ_A is computed as the square root of the variance of A. This method of normalization is useful when the actual minimum and maximum of attribute A are unknown, or when there are outliers that dominate the min-max normalization. The StandardScaler method from Scikit-Learn (2020*a*) standardize features by removing the mean and scaling to unit variance. However, the outliers have an influence when computing the empirical mean and standard deviation, which shrinks the range of the feature values, as shown in the left figure below (Figure 13). In particular, because the outliers on each feature have different magnitudes, the spread of the transformed data on each feature is very different: most of the data lie in the [-2, 4] range for the transformed median income feature while the same data is squeezed in the smaller [-0.2, 0.2] range for the transformed number of households. This scaler, therefore, cannot guarantee balanced feature scales in the presence of outliers.

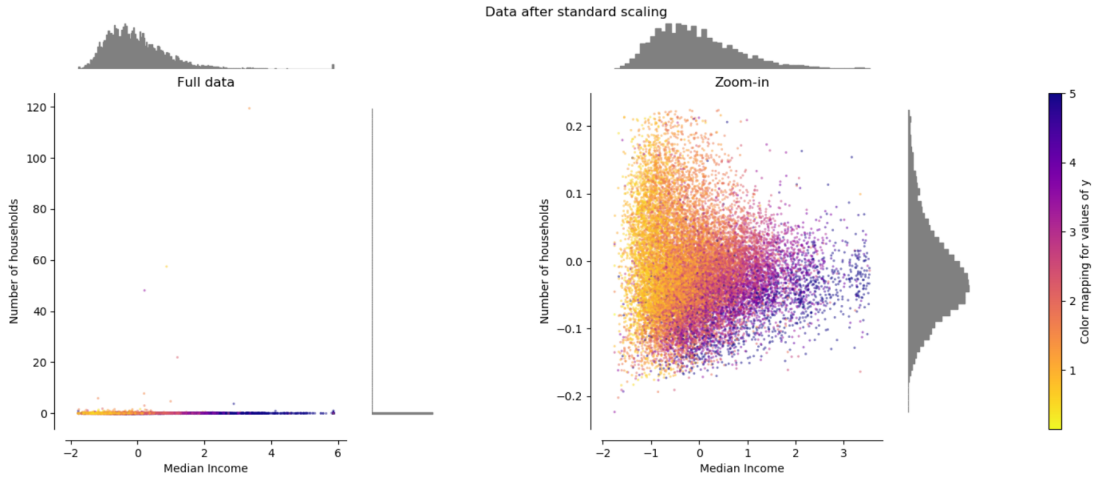


Figure 13: StandardScaler removes the mean and scales the data to unit variance. However, this scaler does not guarantees the absense of outliers (reproduced from Scikit-Learn (2020*a*)).

When outliers are a problem, Scikit-Learn (2020*a*) provides RobustScaler as a solution. This scaler removes the median and scales the data according to the quantile range (defaults to IQR: Interquartile Range). The IQR is the range between the 1st quartile (25th quantile) and the 3rd quartile (75th quantile). Unlike the previous scalers, the centering and scaling statistics of this scaler based on percentiles are not influenced by a few numbers of very large marginal outliers. Consequently, the resulting range of the transformed feature values is larger than for the previous scalers and, more importantly, are approximately similar: for both features most of the transformed values lie in a [-2, 3] range as seen in the zoomed-in figure (Figure 14).

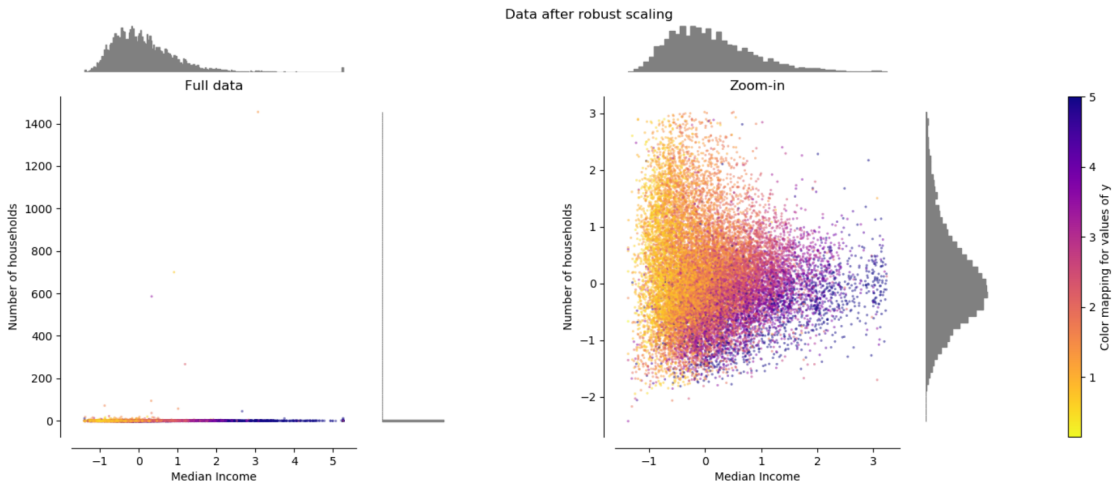


Figure 14: RoustScaler scales features using statistics that are robust to outliers, namely, interquartile range (reproduced from Scikit-Learn (2020*a*)). It's important to note that the outliers themselves are still present in the transformed data. If a separate outlier clipping is desirable, a non-linear transformation is required.

Encoding

The categories of a categorical variable are usually not numeric. For example, eye color can be "black", "blue", or "brown". Thus, an encoding method is needed to turn these non-numeric categories into numbers. This section goes through some encoding methods that are necessary for the machine to interpret the data (more specifically, categorical data) and learn from it.

One hot encoding is the most common approach, and it works very well unless the categorical variable takes on a large number of values, e.g., for variables taking more than 15 different values. As explained by Zheng & Casari (2018), this method uses a group of bits, and each bit represents a possible category. If the variable cannot belong to multiple categories at once, then only one bit in the group can be "on". This is s implemented in scikitlearn as OneHotEncoder (Scikit-Learn 2020*b*). Each of the bits is a feature, thus, a categorical variable with k possible categories is encoded as a feature vector of length k. Figure 15 shows the result of one hot encoding for a possible variable "City" with three different values.

	e_1	e_2	e_3
San Francisco	1	0	0
New York	0	1	0
Seattle	0	0	1

Figure 15: One-hot encoding of a category of three cities (reproduced from Zheng & Casari (2018)).

As already mentioned, this type of encoding breaks down when the number of categories becomes very large. Different strategies are needed to handle extremely large categorical variables. Target encoding, or mean encoding, allows retaining actual useful information about the categories like one-hot encoding while keeping the dimensionality of the data the same as the unencoded data. In general, target encoding is the average of the target value by category. The idea is to replace each level of the categorical feature with a number, and that number is calculated from the distribution of the target labels for that particular level of category. For example, the levels of a categorical feature can be "black", "blue", and "brown". Then replace "black" with some statistical aggregate (e.g., mean, median, or variance) of all the target labels where-ever the feature value is "black" in training data. Normally, random noise should be added to the target average in order to reduce overfitting.

Another simple way to encode data is integer encoding, where each unique label is mapped to an integer. This technique could be used to preserve order between categories (e.g., 0 for "cold", 1 for "warm" ...), or to transform a Boolean variable to the respective value (0 or 1).

Finally, a not so common encoding method is cyclic encoding. This encoding comes from the idea that some data is inherently cyclical. Time is a rich example of this: minutes, hours, seconds, day of the week, week of the month, month, season, and so on all follow cycles. Ecological features

like the tide, astrological features like a position in orbit, spatial features like rotation or longitude, visual features like color wheels are all naturally cyclical. According to London (2016), a way to let machine learning models know that a feature is cyclical is with a transformation into two dimensions: a sine transform and cosine transform of the desired time feature. With this, every new feature (e.g. seconds) is described with two new features. As shown in Figure 16, the distance between two points corresponds to the difference in time as expected from a 24-hour cycle: 12 pm is close do 1 am, December is close to January. A simple translation of these variables into text would not be able to demonstrate the proximity between the times at the ends of the month or hour scales, for example. Furthermore, failing to analyze the cyclical nature of time could prevent the model from detecting specific patterns relating, for example, to parts of the day.

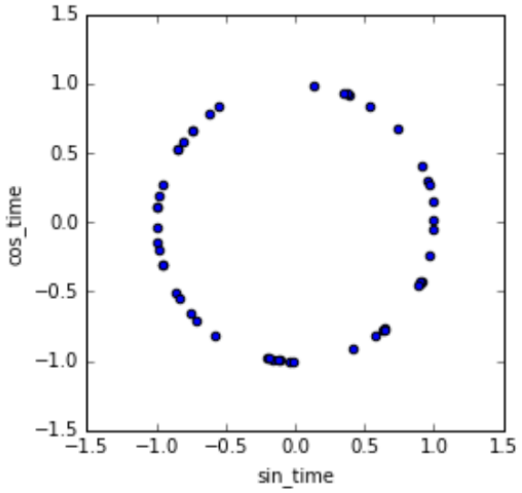


Figure 16: Two-feature transformation in 2D as a 24-hour clock (reproduced from London (2016)).

Binning

Data binning (also called Discrete binning or bucketing) is a data preprocessing technique used to reduce the effects of minor observation errors, turning continuous variables into categorical values. This is a top-down splitting technique based on a specified number of bins. Attribute values can be discretized by applying equal-width or equal-frequency binning, as shown in Figure 17, and then replacing each bin value by the bin mean or median (Han et al. 2011). Another option is replacing a bin by a value representative of that interval.

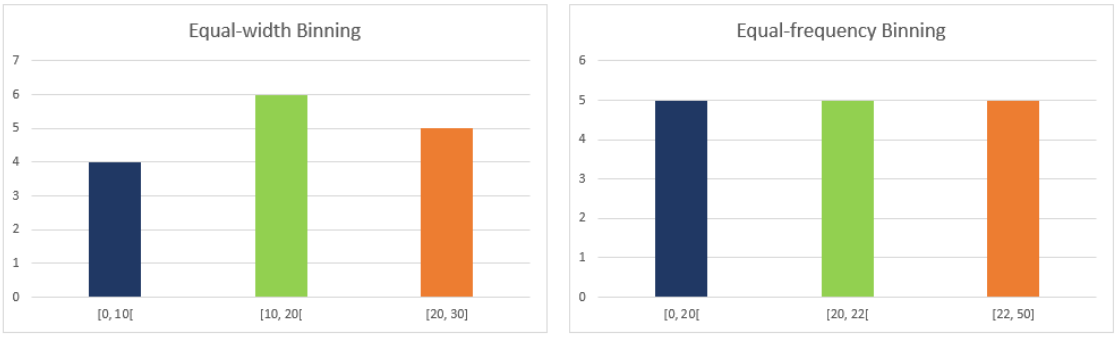


Figure 17: Equal-width or equal-frequency binning.

Although the typical binning is a technique for continuous numerical variables, it can be used for categorical values with a large number of categories occurring with little frequency. These categories can be grouped in a bin with a global category like "Others", maintaining an equal-frequency binning among categories.

2.7 Related Work

This section presents studies related to customer behavior and prediction of problems arising from installations problems, or articles that use similar techniques to those that will be used in this work, whether in telecommunications area or any other. It is structured with the resource to the guidelines for performing systematic literature reviews, by Kitchenham & Charters (2007), although quite simplified.

Thus, subsection 2.7.1 defines the search strategy, in order to achieve relevant information, and subsection 2.7.2 presents the inclusion and exclusion criteria, to set the boundaries for the research. For all the selected articles, the subsection 2.7.3 shows the results of the research, mentioning relevant parts of some articles and a balance justifying why it is useful or not for the present dissertation.

2.7.1 Search Strategy

Three electronic information databases (*Google Scholar*, *ScienceDirect* and *ACM Digital Library*) where used to identify articles published on the subject. Unpublished research was not considered. The search strategy included the following terms:

("Predict" AND "Problem" AND "Installation")
OR
("Telecommunications" AND "Installation")
OR

("Customer" AND "Predict" AND "Telecommunications" AND "Complaints")
OR
("Prediction" AND "Failure" AND "Causes")

Then, all these combinations were searched with the term "Machine Learning". These combinations were used to determine the prediction of problems arising from installation processes or relevant problem-solving techniques, not focusing specifically on the telecommunications area. This sequence of terms aimed to provide a wide range of results, once the main goal is to provide a global review of problems factors.

2.7.2 Inclusion and Exclusion Criteria

After identifying suitable articles, it was defined an inclusion and exclusion criteria to identify the primary studies that provide direct evidence about the research objective. With this, the studies were reviewed against the following inclusion criteria: a) written in English; b) studies from any geographical location; c) any studies using machine learning models and techniques, specifically, supervised and unsupervised learning. Articles were excluded when a) studies were lacking information on how the outcome was reached or the final measures were achieved; b) studies not include minimum outcomes or adequate evaluation metrics; c) studies were duplicated; d) only a slide version or an abstract is documented; e) studies were not published; f) studies used a sample minor than 100 records - studies that use larger sample sizes have a better statistical power because it enhances the possibility of generalization of results.

2.7.3 Literature Review

Customer behavior is not a new concern in the telecommunications industry. In fact, due to the highly competitive environment, these companies have client satisfaction as one of their main goals, performing several predictive studies in this area. Acknowledging this, there are several articles in the telecommunications area, involving churn prediction, intelligent call routing, fraud prediction, network optimization, among others. However, and regardless of the industry, no study was found that focused specifically on the installation process, or customer behavior after having problems with the installation process. There was a need for thinking more broadly about what this work might need and use, and map the possible areas that are linked to the question *"What clients are more likely to contact client support with problems or questions arising from the installation process?"* . It was necessary to sort out what is most relevant and related, and the literature found is brought together in a way that supports the research. This work is novel not on the methods or techniques it is going to use, but on the topic itself.

With this in mind, this section goes through articles with common problems in the telecom area that may be related (even if indirectly) to the theme of this thesis, and studies that use techniques similar to those that will be used in this work.

Churn Prediction

Because customer behavior based on previous factors is the main idea, some articles related to churn prediction in telecommunications were analyzed and found relevant. Choi (2018), in his master thesis entitled "Predicting Customer Complaints in Mobile Telecom Industry Using Machine Learning Algorithms", focused on customer retention and the complaints being the key factor around this idea. In this particular work, machine learning models were implemented to predict customer complaints of a Korean mobile telecom company, and a database of 10000 Korean mobile market subscribers was used. Variables like gender, age, device manufacturer, service quality and complaint status were considered, and four ML models were used: ANN (Artificial Neural Networks), SVM (Support Vector Machines), kNN (k-nearest neighbor) and DT (Decision Trees). The study showed an outperformance of ANN algorithm with 87.3% prediction, and the use of segment-prediction model resulted in a better accuracy (segments of customer group were examined by age, gender and device manufacturer). Although the purpose is not the same, the techniques used in Choi's master's thesis are quite pertinent and relevant to the present work.

Another study on churn prediction, by Ahmad, Jafar & Aljoumaa (2019), had the same concern as Chiyoung Choi's master thesis but used slightly different techniques. In their article, "Customer churn prediction in telecom using machine learning in big data platform", it was developed a churn prediction model which assists telecom operators to predict customers who are more likely to churn. They have used ML techniques on a big data platform, proposing AUC as the standard measure adopted, achieving 93.3% for AUC. They also used SNA (social network analysis) features, which enhanced the performance of the model from 84% to 93.3%, against AUC standard. The ML models used were decision trees, random forests, gradient boosted machine tree (GBM) and extreme gradient boosting (XGBoost), having better results by applying XGBoost, a well-known algorithm for its great performance when compared to similar ones. Besides this, because of the imbalanced data on the target (only 5% of the entries represented customers' churn), the authors of the article concluded that it was better to use undersampling or tree algorithms not affected by this problem. Abdelrahim Kasem Ahmad, Assef Jafar and Kadan Aljoumaa's study proved to be very useful because of the imbalance problems it faced, and the use of techniques and models similar to those that will be implemented in the present work.

Failure Prediction

This work does not aim to predict when specific equipment is going to fail but to analyze the factors (client profiling, equipment details, technician service) that can lead to failure and, consequently, customer complaints. With this, some failure prediction studies were found relevant to this thesis, such as Hamerly & Elkan (2001) work "Bayesian approaches to failure prediction for disk drives". Like telecommunications equipment, hard disk drive failures are rare but are often costly. The ability to predict failures is important to consumers, drive manufacturers, and computer system manufacturers alike. In Hamerly & Elkan (2001) paper, they investigate the abilities of two Bayesian methods to predict disk drive failures based on measurements of internal drive conditions. First, the problem is viewed from an anomaly detection stance. It is introduced a mixture model of Naïve Bayes sub-models (i.e. clusters) that are trained using expectation-maximization. The second method is a Naïve Bayes classifier, a supervised learning approach. Both methods are tested on real-world data concerning 1936 drives. The predictive accuracy of both algorithms is far higher than the accuracy of thresholding methods used in the disk drive industry today. What is particular in this study is the combination of methods to predict failure. The mixture model of Naïve Bayes sub-models with expectation-maximization showed better results combining only the three best attributes. The supervised Naïve Bayes classifier performed better using all attributes than using only three. Both failure prediction methods presented in this study performed better than the current industry standard methods, well enough to be useful in practice. The thresholding methods that are standard today in the disk drive industry have been estimated to yield a drive-level true positive rate of 0.04 to 0.11 at a false positive rate of 0.002. At the same false positive rate, the first approach used by Hamerly & Elkan (2001) (mixture model with expectation-maximization) achieves a true positive rate of 0.30, about three times better.

One last study was taken into consideration for this thesis, although the theme is slightly different. "Predicting the Reasons of Customer Complaints: A First Step Toward Anticipating Quality Issues of In Vitro Diagnostics Assays with Machine Learning", by Aris-Brosou, Kim, Li & Liu (2018), also concerns about customer complaints, but the study aimed to help prevent customer complaints by predicting them based on the QC (quality control) data collected by in vitro diagnostic systems. Instead of customer behavior, their work focuses on failure prediction. Traditionally, and as mentioned above, failure prediction in industrial applications aims at predicting when a particular system is likely to fail. Here, it was addressed a different question, one not directly related to the timing of failure, but one that focused on which type of failure can be predicted based on customer complaints. As for the methods, the authors combined QC data from five selected in vitro diagnostic assays, with the corresponding database of customer complaints over 90 days. A subset of these data over the last 45 days was also analyzed to assess how the length of the training period affects predictions. They defined a set of features used to train two classifiers, one based on decision trees (CART) and the other based on adaptive boosting, and assessed model performance by cross-

validation. The cross-validations showed classification error rates close to zero for some assays with adaptive boosting when predicting the potential cause of customer complaints. The performance was improved by shortening the training period when the volume of complaints increased. Denoising filters that reduced the number of categories to predict further improved performance, as their application simplified the prediction problem. These are relevant DM techniques for the present work, that can enhance a better performance for the models to be used.

All the research done had a great weight in this dissertation, not only to find the "gap" in current knowledge but also because the combination of the methods used on all selected studies allows producing a more reliable and consistent study, aiming for better results.

3. PREDICTING INSTALLATION PROBLEMS

This chapter outlines in depth the work methodology used and all the phases until reaching the results. The work went through analyzing the data and treating it to obtain a clean dataset with new underlying features. The modeling of the problem took into account the substantial imbalance of the data, and several ML models were executed, followed by the optimization phase and the respective evaluation.

3.1 Methodology

A methodology refers to the overarching strategy and rationale of a research project. It involves studying the methods used in a particular field and the theories or principles behind them, in order to develop an approach that matches the work objectives. To have a better response to the present case study, the CRISP-DM methodology was used, which describes approaches commonly used by specialists to tackle problems of this nature (Saltz et al. 2017). CRISP-DM (CRoss-Industry Standard Process for Data Mining) was conceived in late 1996, and it is a comprehensive data mining methodology and process model that provides anyone — from novices to data mining experts — with a complete blueprint for conducting a data mining project. CRISP-DM breaks down the life cycle of a data mining project into six phases: business understanding, data understanding, data preparation, modeling, evaluation, and deployment (Shearer 2000). These steps are not necessarily sequential, existing a cyclical nature between all of them (except the last one), as illustrated in Figure 18. The task of deploying the solution into production, the last phase of CRISP-DM, is not presented in this study and is the next step after checking whether the results are proved to be sustainable.

Figure 18: Process diagram showing the relationship between the different phases of CRISP-DM (reproduced from Wikipedia (2019*a*)).

The phases of the data mining process are detailed below:

1. Business Understanding: focuses on understanding the project objective from a business perspective, defining a preliminary plan to achieve the goals.
2. Data Understanding: data collection and the start of activities to familiarity with the data, identifying problems or interesting sets.
3. Data Preparation: construction of the final dataset from the initial raw data. It usually occurs several times in the process.
4. Modeling: various modeling techniques are applied, and their parameters are calibrated to optimal values. Thus, it is common to return to Data Preparation during this phase.
5. Evaluation: before proceeding to final deployment of the model built by the data analyst, it is essential to more thoroughly evaluate the model and review the model's construction to be sure it accurately achieves the business objectives.
6. Deployment: the knowledge gained by the model is organized and presented in a way that the customer can use.

One of the advantages of this methodology is that it is independent of the industry because it is a generic process model that can be specialized according to the needs of any particular company. It is also independent of the tools used in the process, and it is closely related to KDD process models.

3.2 Business Understanding

The goal of the work presented in this paper is to reduce customers' problems related to installing (or upgrading) new services. An installation process is defined as the establishment of a service, such as the internet or telephone, in which specialized telecommunications knowledge is required for the person performing the craft, and requiring a technician to visit the customer's home. Knowing in advance who will face an installation problem and being aware of the customer's probable future inconveniences allows the company to provide personalized services (suitable assistance, equipment, or technician), culminating in better customer experience and higher installation service efficiency. It also allows having a proactive action and dialogue after the installation with the client and before the complaint call occurs, responding efficiently to the situation. All these actions ultimately lead to a decrease in time/equipment spent (number of trips to the customer's home or expensive equipment replacement).

The data used in this study were retrieved from a telecommunications company in Portugal. The DM aim is to develop accurate models that can support the decision-making process by predicting

if a particular customer will have a problem that requires a maintenance intervention in a period of time following the initial installation. For the prediction to be the most accurate as possible, models use as input values the installation records from the clients who had and clients who had not have technical problems within that period since the installation.

3.2.1 Tools and Technologies

Next are presented the tools used to code and to implement approaches throughout this thesis, as well as some relevant libraries.

- **Python**: is an interpreted, object-oriented and high-level programming language. Its simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance (Python 2020). Python is widely considered as the preferred language for teaching and learning machine learning because it is open source, capable of interacting with almost all the third-party languages and platforms and data handling capacity is great (School 2017).
  - SciKit-Learn: is a simple Python library for data analysis and mining-related tasks.
  - Pandas: referred to as Python Data Analysis Library, is another Python library for availing high-performance data structures and analysis tools.
  - Seaborn: is a Python library, designed to visualize statistical models. It has the potential to deliver precise graphs like heat maps.
  - Numpy: is a package that supports multidimensional arrays and arrays, having a wide collection of mathematical functions to work with these structures.
- **JupyterLab**: JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: itis possible to configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: writes plugins that add new components and integrates with existing ones (Jupyter 2019). Its popularity among data scientists relies on the "all in one place" concept, and on the easy to share and convert.
- **Visual Studio Code**: Visual Studio Code is a lightweight but powerful source code editor which runs on every desktop and is available for Windows, macOS and Linux (Code 2020). It supports working with Jupyter Notebooks natively, as well as through Python code files. It is possible to manage source control, open multiple files, and leverage productivity features like IntelliSense, Git integration, and multi-file management, offering a brand-new way for data scientists and developers to experiment and work with data efficiently (LinkedIn 2018).

3.3 Data Understanding

The second phase of CRISP-DM usually includes getting familiar with the data, to identify quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses about hidden information. With this, in this section, the data from the initial individual datasets is analyzed. These datasets subsequently underwent an integration and cleaning process, explained in the next section.

For this dissertation, the data was provided in comma-separated values (CSV) files containing miscellaneous information, and all the records were previously anonymized and filtered in order to comply with privacy regulations. The client support dataset ranges from 2018 to 2019 and includes information not only on each call made from or to the telecom company, but also the complaints arriving to support via other channels. In this process, in each record, it is registered the categories of the problem (technical or not) being reported, temporal data such as call start and ending (if applicable) or client's contract periods, the respective customer key and his/her service keys, the kind of network used by the customer and other service details, the front team contacted with the problem, and, lastly, some metrics like if the complaint was solved in the contact or the recurrence of the problem. Figure 19 shows the number of problems that arrived at customer support in 2018, regardless of the input channel (telephone, online, ...), by the top 12 most recurring problems. A large number of problems related to technical service failures are notorious, which reinforces the need to inspect the origins of the problems.

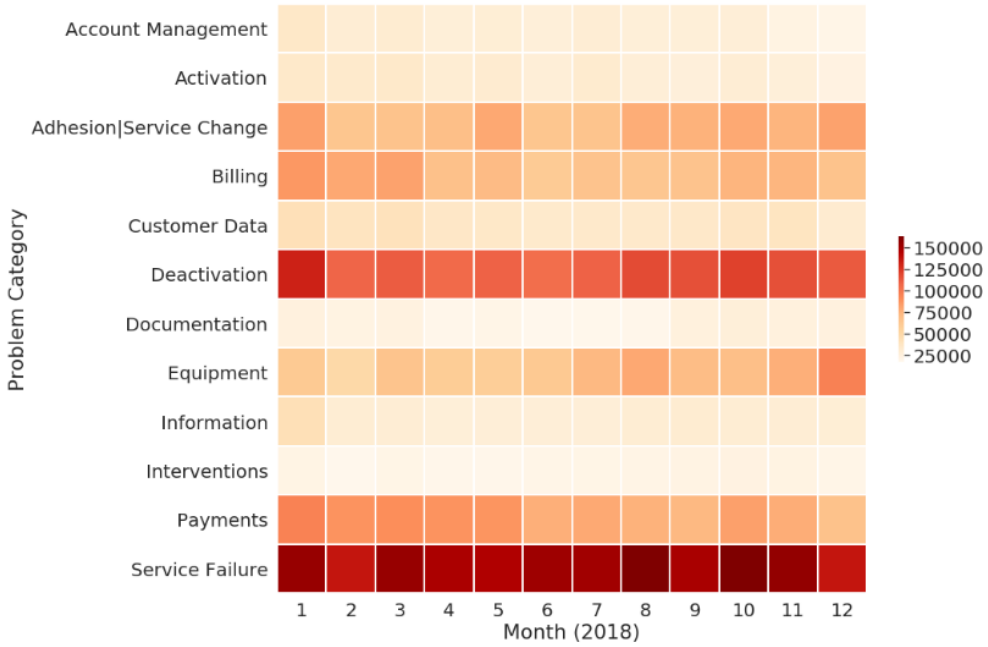


Figure 19: Number of problems arriving to customer support monthly, by top categories, referring to 2018.

Figure 20 presents the number of the clients' problems that were solved right in the first contact by the assistant, according to the category of the problem: technical, non-technical or not applicable. It is noticeable that most problems are technical, and there is a marked difference between solving different types of problems on the first contact.

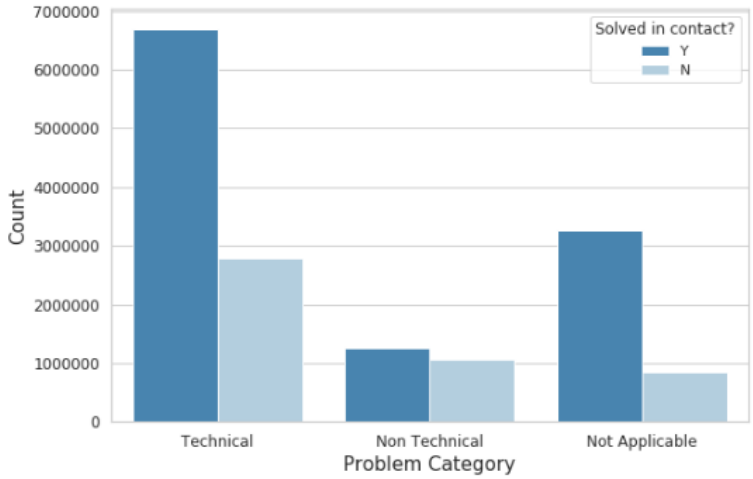


Figure 20: Number of problems solved in the first contact to customer support, by category.

The history of the equipment models' of a certain user, and the ones currently in use by the customers of the company are stored in a different dataset, along with the timestamps of beginning/ending and the type of the machinery (in each record).

The maintenance dataset ranges from 2018 to 2019 and stores all the records about the action of the technician at the customer's home. Every line has dates registered, such as the start and end of service, as well as the technician's identification key, the type of service performed (e.g., installation or maintenance) and the result of the service. Figure 21 illustrates the number of occurrences of each type of intervention at the customers' homes, according to the type of technology of the customer, being HFC (Hybrid fiber-coaxial) the most common technology. HFC is a term for a broadband network that combines optical fiber and coaxial cable, and it is globally employed since the early 1990s. The FTTH technology (Fiber to the Home) surpasses DTH (Direct to Home), remaining this last technology for remote locations.

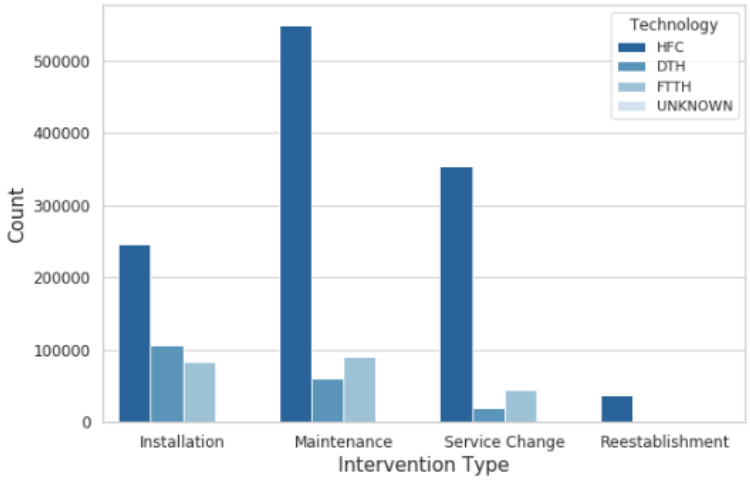


Figure 21: Number of interventions at the customers' home according to its type, and by technology.

More datasets include customer information, such as personal data, where a history of clients' data is recorded along with a timestamp. The approximate age of the customers is stored in a different dataset, alongside with his/her identification key.

The last dataset contains the information obtained from the utilization of services, like consumption, where each record synthesizes monthly averages and usages of a certain client. The dataset concerning the monthly consumption of customers was not used to analyze consumption, since installation takes place in a shorter time than necessary to record usages (30 days). Even so, this dataset provides relevant and detailed information about customers' packages. Figure 22 schematizes the average of the amount of the downloaded data, organized by month and regarding the year of 2018. It is visible that, on average, the number of downloads increase around the Christmas holidays, and has the lowest consumption in April.

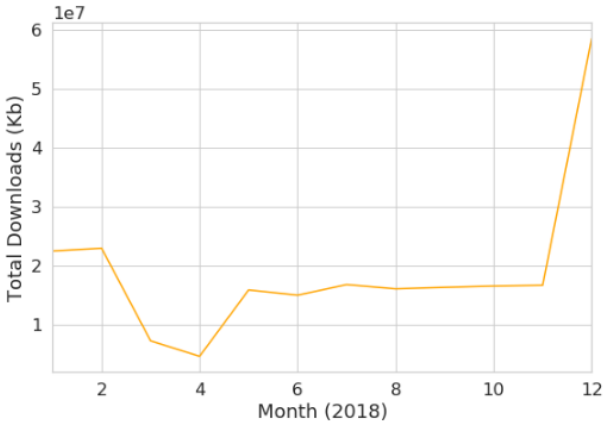


Figure 22: Average amount (Kb) of downloaded data per month, for the year 2018.

All these datasets, supplied by the telecommunications company, are summarized in Table 2. The first column indicates the thematic of the dataset, and the next one provides a brief description

of them. Then, column “Nulls” checks the presence of nulls in the file, and “Date Range” gives the date range of the records provided. Lastly, the number of files is indicated in the table, as well as the size of these files combined.

Table 2: Summarized datasets supplied by the telecommunications company.

Dataset	Description	Nulls	Date Range	Nº Files	Size (GB)
Customer Support	Records of the clients' problems and complaints	yes	2018-2019	4	20,17
Equipment	History of all the clients' equipment	yes	—	1	14,23
Maintenance and Installations	Records of the interventions at the clients' houses	yes	2018-2019	1	1,32
Customers' Personal Details	Personal data records	yes	—	1	1,16
Customers' Age	Records of the estimated age of the clients	no	—	1	0,12
Customers' Consumption and Service Details	Details about the services and monthly consumption	yes	2018-2019	2	21,00

Since the main goal is to investigate which installations have proved to be problematic, the main dataset was the one with the reports about the installations and maintenance at the customers' home. Then, this information was complemented with data from the other datasets mentioned above, such as personal details, types of equipment in use and service details, and the customer's previous calls to the customer support with the respective complaint or request. The data integration process is illustrated in image 23.

3.4 Data Preparation

After analyzing the data from the initial individual datasets, this section describes the data preparation techniques performed. Data preparation is the method for selecting, cleaning, and processing of raw data that precedes processing. It usually includes reformatting and correcting data, the mixing of data sets to enrich data, among others. Since it was dealt with various types of algorithms, and not all of them accept, for example, null values, it was performed a complete cleaning of the data, described throughout the next sections.

3.4.1 Integration

Because the main goal is to investigate which installations have proved to be problematic, the main dataset was the one with the reports about the installations and maintenance at the customers' home. Then, this information was complemented with data from the datasets mentioned previously, such as personal details of the client, types of equipment in use and service details, and the customer's previous call to the customer support with the respective complaint or request. The data integration process is illustrated in Figure 23.

In order to achieve a single dataset, and starting with the maintenance and installation dataset, the starting point was linking to it the information related to the problem/complaint that reached customer support. This was possible due to the customer service ID, a unique key for each service, which is the common point between the two datasets. The merge asof proved to be useful in joining the two datasets, since it performs an operation similar to a left join, in addition to taking the date into account. Thus, through the service ID and the dates with a defined maximum tolerance of 30 days, these datasets were joined forming a timeline from the complaint to the intervention at the client's home to solve the problem. It is important to note that, since the customer support data contains information such as payment calls, contract changes, among others, it was necessary to make a filter in this dataset in order to extract only data related to technical problems that could imply maintenance. If this filter were not performed, if a customer had contacted customer support with an installation problem and, after a few hours, made a call about portability, the dataset for installations and maintenance would join the last call in error.

Then, the customer's personal information and approximate age were added to this dataset. Both sets of data had in common with the main dataset the customer ID, unique for each customer of the company. For the age, a simple merge was used, which is identical to a left join. For the remaining customer data, since it is updated over time, the merge asof operation was used, which combines the information by ID and by the closest date.

For the information of the equipment used by each customer, useful to detect possible problematic machines, the merge asof was used again by the customer service ID (connection point between the main dataset and the equipment). The date was necessary since, in this dataset, a history of the clients' equipment is maintained.

Finally, the service details dataset (eg., internet speed) was added to the main dataset, and the merge asof was used by the customer's service ID. It should be noted that the consumption present in this last dataset was not used, since its registered monthly, and an installation problem occurs in less than one month, as will be explained later. Even so, this dataset provides relevant and detailed information about customers' packages.

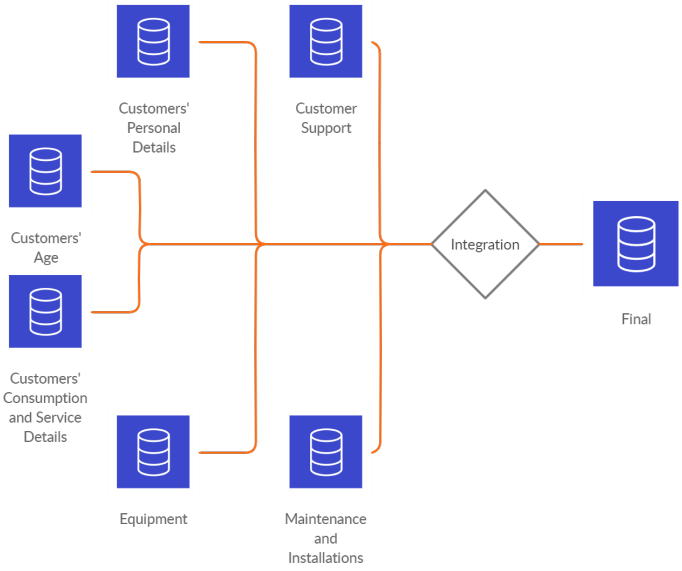


Figure 23: Data integration schema.

3.4.2 Selected Data

All the variables and rows available underwent a selection process as they would be useful in predicting problems arising from the installation process. Data selection is a vital preprocessing task in data mining because not all attributes are relevant to a given problem. Nowadays, in addition to models having more efficient resource management and greater capacity for processing large amounts of data, it is possible to study what is the best or most valuable combination of variables. Thus, with regard to columns, the selection process was based on the elimination of columns that proved to be unnecessary to the naked eye. As for the rows, some were removed, such as observations referring to businesses instead of customers or observations whose installation had not been carried out due to the customer's absence were eliminated, since they were not part of the study's object. This filter and others are explained in detail in the next section.

3.4.3 Cleaning Report

In addition to the fact that data in the real world is not perfect, it does not always meet the business and data mining objectives. This makes it necessary, in addition to the typical data cleaning, to filter the rows since not all are suitable for the problem of this thesis. With this, the first step was to filter the dataset only by the maintenance or installations that had been carried out. In order to analyze problems arising from installation processes, it is necessary that an installation has occurred and has not been canceled due to lack of customer availability, for example. Besides, the dataset was filtered by the residential segment (only private customers, excepting companies) and by the nonexistent PBX system. PBX stands for Private Branch Exchange, which is a private telephone

network used within a company or organization. To make sure that there were no rows associated with opened processes and that only private customers were analyzed, variables from the customer support dataset were used, and the complaints data was filtered by private business and closed processes.

Since the company is in the process of data migration and transformation, where the typification of problems that reach customer support is undergoing a new categorization process, it was also necessary to filter the records according to the old system or the new system. If the categories of problems belonging to both systems were maintained, an inconsistency problem would appear, where the same problem could have two different categories. In order to standardize the categories, and taking into account that the new system is still very recent and has fewer occurrences, the lines related to the new system were removed, leaving only those records that were categorized according to the old nomenclature.

Some cases of aggregation were performed, where data is gathered in groups in order to generalize an attribute and possibly improve the performance of the posterior model. In this way, the models of the equipment were added and renamed, as well as the counties were grouped into districts. The typification of the problems was also grouped and renamed.

For all the attributes in the dataset, to ensure that there was no incomplete or inconsistent information, data with noise or null values were removed. More specifically, duplicated rows, as well as the ones with noise or redundancy (e.g., cable and HFC), were eliminated from the dataset. The same goes for the variables that were dropped out if redundant, deprecated, or wrongly calculated. Concerning null values, special care was necessary to identify the respective meanings since not all blank records refer to unknown values, and some should be replaced by an identifying string (Han et al. 2011). The true nulls were replaced by the average/mode or changed to an 'unknown' string for categorical variables or number as -1 for numerical variables. Besides, variables with a large percentage of null values were also eliminated. In order to provide the models with the right and unified data types, it was also performed a data type conversion, especially in the Boolean values and dates. Finally, several categorical variables underwent a frequency binning process, explained in section 2.6.3, since they had very dispersed values, and that much detail would not be needed in their analysis and processing.

3.4.4 Feature Engineering

The task of feature engineering is crucial to accurately represent the underlying structure of the data and create the best model. With this in mind, it was made an effort to analyze all the features and to derive new ones.

Considering the set of variables related to installations and maintenance, it was decided to create

some metrics that could characterize the work and performance of technicians at customers' home, in order to examine potential problematic professionals. Features were created, such as the duration of the service at the client's home and the amount of time that the technician needed under or above what was initially scheduled. Its performance is calculated through three factors: (1) if the technician needed more time at the client's home than the scheduled initially, (2) if that visit is a recurrence of the same problem, and (3) if it is not a visit by the same technician in less than 10 days (a typical situation when the technician did not perform the service properly). Features were also built that could indicate problematic zones or teams, such as the number of installations or maintenance by county, district or delegation. For the client, some variables were created in order to express his pain or discomfort, such as the number of past interventions or the number of calls to customer support taking into account multiple intervals and categories (technical or non-technical problems, and interval time equal to the last month or since forever). For customer service technology, and since there are cases of customers already belonging to the company where there is a technology change at the installation, a technology change detection variable was created in order to investigate whether this change would be problematic. It was also decided to extract cyclical variables to represent time. As it is going to be explained in the next section, the use of sine and cosine in temporal representation allows the extraction of features such as month or hour, and to distance them correctly (e.g., December is close to January, or 12 pm is close to 1 am). A simple translation of these variables into text would not be able to demonstrate the proximity between the times at the ends of the month or hour scales, for example. Furthermore, failing to analyze the cyclical nature of time could prevent the model from detecting specific patterns relating, for example, to parts of the day.

Bridging to the variables related to customer support, it was decided to detect possible failures in service or poor problem handling. Thus, features related to the assistant were calculated, such as the percentage of problems solved right in the contact or the duration of the ticket (or call), and the respective average duration of the assistant's tickets. Since this data contains some temporal information on the customer's contract, some variables were built regarding the time the customer is loyal to the company. Again, in order to discover possible pain of the customer, a re-incident variable was calculated that indicates whether that problem being reported to customer support is repeated.

As for the client's personal information, features like the county were aggregated to create the district, as a higher level of abstraction could result in more accurate results.

It should be noted that all variables were calculated with information to date, or, in other words, data from future records was not fetched to calculate a feature. This avoids the phenomenon of data leakage, which, in this case, consists of using information from future cases to forecast current data. For example, using future data to calculate an average would be the same as using the statistics for a certain football player from next week to calculate their performance in today's game. All the

features created are detailed in the list below, divided by thematic groups of variables.

- **Installation and Maintenance**:
  - Technician performance
  - Intervention duration at home (min)
  - Time under/over the scheduled (min)
  - Average time of the technicians at houses
  - Number of past interventions by delegation, district and town hall
  - Number of past interventions by client
  - Technology change (before and after installation)
  - Day, part of the day, weekday, week and month
  - Verification if the intervention occurred in a weekend or not
- **Customer Support**:
  - Ticket duration
  - Mean of past ticked duration by assistant
  - Percentage of problems resolved right in the contact by the assistant
  - Number of calls of a client to customer support (until the date and last month)
  - Number of calls of a client with technical problems to customer support (until the date and last month)
  - Number of calls of a client with non-technical problems to customer support (until the date and last month)
  - Re-incidence to the category of the problem
  - Duration since the beginning of contract/service (days)
  - Duration since the customer is loyal to the company
- **Clients' Personal Information**:
  - District
- **Others**:
  - Time between the call with the complaint and the action at home
  - Verification if occurred a week change between the complaint call and the trip to the client's house

3.4.5 Target Building

Often, the target variable is not explicit or incorporated in a dataset, as is common in an academic environment. Thus, it is necessary to build it to meet the business and data mining objectives. As each line represents an installation or maintenance, it was necessary to identify, in the case of installations, those that had originated problems or not. This means that this target variable was built verifying if, for the same customer service, there was an installation or service upgrade followed by a (correction) maintenance after a certain period of time. As is deductible, after calculating the target, all lines related to maintenance have been eliminated, leaving only the lines related to the installation of new services or upgrading services already existing at the customers' house that went wrong or not. The following graphic (Figure 24) depicts the distribution of possible target variables according to four different scenarios: the customer has a maintenance intervention until 7 days, 15 days, 30 days, and 60 days after installation. Given the proportion of the data shown for each period, and because the month period is commonly used in the company, it was decided that a problem arising from an installation process would be treated as such if it happened within 30 days and the target was build considering this. The figure below also shows signs of an imbalanced dataset.

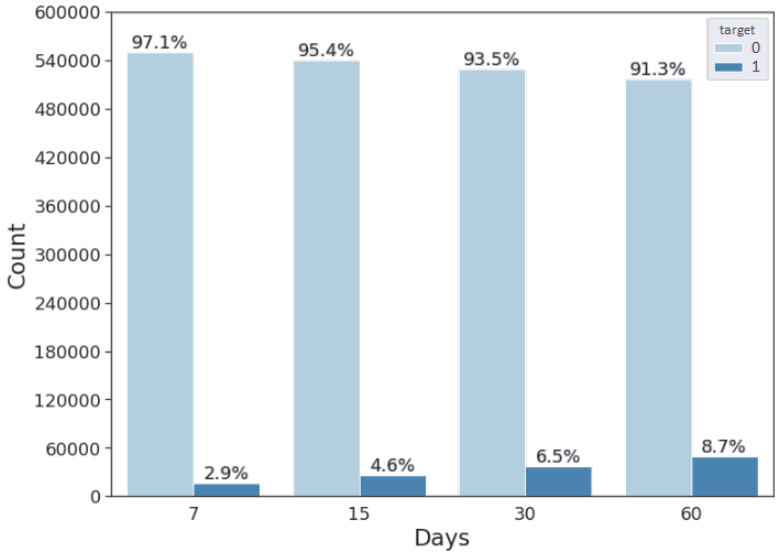


Figure 24: Target distributions - choice of target variable for maintenance occurring up to 30 days after the initial installation.

The task of building the target was one of the last because, after it, the maintenance lines disappeared, and these lines were useful to extract features, such as the number of past interventions by client.

3.4.6 Encoding and Normalization

Feature normalization and standardization is an important issue. Multiple variables can have varying degrees of magnitude, range, and units. Outliers are very common in features, and they are defined as an observation point that is distant from other observations. An outlier may indicate an experimental error, or it may be due to variability in the measurement. All of these phenomenons are a significant obstacle as a few machine learning algorithms are highly sensitive to these features. The figures below demonstrate the different possible ranges among variables and some outliers. For example, Figure 25a shows the huge range of the feature that represents the number of (past) interventions by district. The minimum is 0, the maximum is 308761, and the values in the interquartile range (25th to the 75th percentile) are between 17567 and 131441. The few visible outliers are not an input error, but a representation of the variability of the feature. Comparing this variable with the one in the boxplot in Figure 25b, the difference between the intervals that the variables present is visible. Once again, the visible outliers are not input errors.

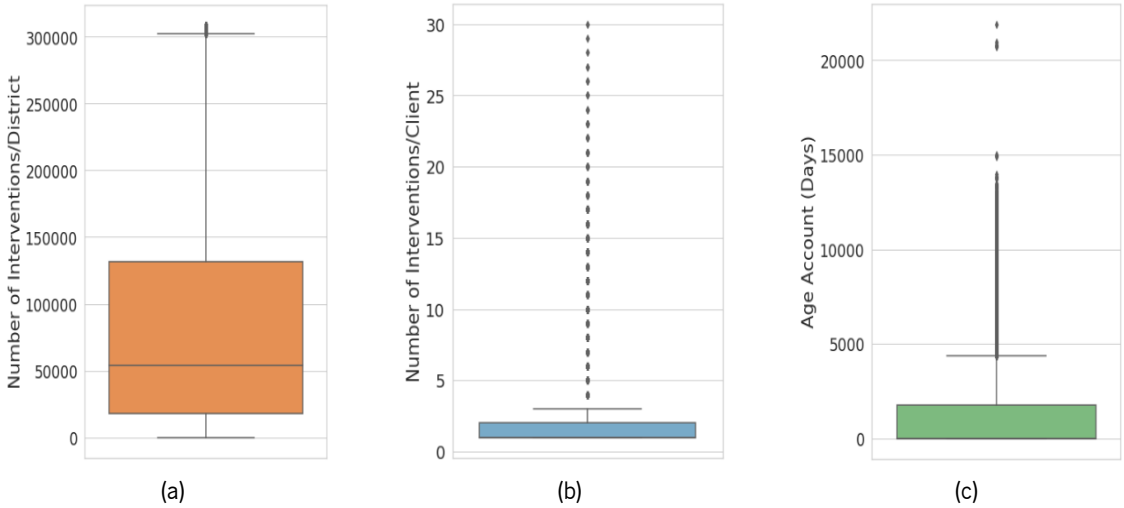


Figure 25: Range and outliers of features: (a) feature representing the number of interventions by district, (b) the number of interventions by client, and (c) the age account, in days.

By contrast, the distribution of the variable "Age Account" (Figure 25c) that represents how long a certain customer is faithful to the company, shows signs of input errors or anomalies. For example, a value of around 20000 days is approximately equal to 57 years. Since the telecom company has 26 years of existence, most of the outliers in the boxplot exemplify impossible values.

So, to ensure data normalization, all the numeric features were changed to a common scale without distorting differences in the ranges of values. StandardScaler and RobustScaler were the two scalers used for this task: StandardScaler removes the mean and scales the data to unit variance, while the centering and scaling statistics of RobustScaler are based on percentiles and therefore are not influenced by a few numbers of very large marginal outliers (Scikit-Learn 2020*a*). Experimenting

with multiple scaling methods can increase the score on classification tasks, and scaling methods affect differently on different classifiers. Section 3.5 shows the usage of different scalers, algorithms and sampling techniques on the chosen scoring metric for this work, and section 3.6 presents the results on these combinations.

Since most of the machine learning algorithms can only read numerical values, the encoding of categorical variables into numerical values is essential. For the categorical features, one-hot encoding and target encoding was performed, depending on the cardinality of the feature. If the cardinality was high, and to avoid the curse of dimensionality, the target encoding proved useful. Since this method is prone to overfitting, random noise was added to the target average. Boolean variables were also encoded in the respective values 0 or 1.

For the model to be able to see the inherently cyclical nature of time, it was also used cyclical encoding with two dimensions: cos and sin. This allows to extract interesting features like part of the day or month and to distance nearby time positions (e.g., December is close to January, or 12 pm is close to 1 am).

3.4.7 Final Dataset Summary

After all the data selection, cleanings, and transformations mentioned above, the dataset that initially consisted of 1787349 entries was reduced to 1415407 and, subsequently, shrank to 566244 observations with the construction of the target. Each of the entries is described by a total of 72 features (not considering one-hot encoding modifications), which are the consequence of a choice and metamorphosis of 156 original ones joined together. The final dataset, summarized in Table 3, refers to approximately 500 days, comprising the period between May 21, 2018, and September 30, 2019.

Table 3: Dataset dimensions after each phase of data preparation.

	Original Size	**Data Preparation**	**Target Building**
Entries	1787349	1415407	566244
Features	156	71	72

In the following tables is presented some statistical information that may help to comprehend how the data of the final dataset is distributed. Table 4 shows statistical measures related to some numerical variables, described by the minimum (min), maximum (max), average (avg), and standard deviation (std dev). Variables #ots_client and #calls represent the previous number of maintenance/installations and the previous number of calls of the client for the customer support, respectively. Variable #ots_county is the number of known services performed at the town hall of that client, and tech_performance measures the work quality of the technicians based on how often they have done something wrong. The last three variables in the table represent how long the

customer is in the company, the duration since the beginning of the present contract, and, finally, the clients' internet speed, respectively.

Table 4: Statistics measures of some business relevant numerical variables.

Variable	Min	Max	Avg	Std Dev
#ots_client	0.0	25.0	1.5	1.0
#calls	0.0	72.0	0.3	2.7
#ots_county	0.0	85292.0	10963.6	14559.8
tech_performance	0.6	1.0	0.9	0.2
dur_process_home (min)	0.0	7550.0	79.7	107.4
age	18.0	72.0	30.7	26.3
account_age (days)	0.0	13974.0	543.4	1177.9
begi_contract (months)	0.0	61.0	2.2	7.8
internet_speed (mbps)	6.0	200.0	132.6	98.5

The distribution of occurrences for some categorical variables is presented in Table 5. The customer's effort reflected on the number of calls made to the customer support in the previous month is represented in variable client_pain (month). The type_ot is the type of service performed at the customer's home, and the bundle is the number of services of the customer (e.g., 3P includes landline, TV, and internet). The last two attributes describe the technology of the service (fiber, cable, or satellite) and whether the installation included a technology change. The range of the variables do not represent the totality of the company's customers, but the distribution of the customers covered in this study.

Table 5: Statistics measures of some business relevant categorical variables.

Variable	Range	Percentage (%)
client_pain (month)	Yes	10,0
	No	90,0
type_ot	Installation	49,6
	Service up/downgrade	45,7
	Reestablishment	4,7
bundle	0P	1,0
	1P	1,3
	2P	1,0
	3P	6,4
	4P	15,2
	5P	4,1
	Unknown	71,0
cel_technology	HFC	70,0
	DTH	15,1
	FTTH	14,7
	Unknown	0,2
change_technology	Yes	0,7
	No	24,2
	Unknown	75,1

3.5 Modeling

Before building the model, it was necessary to generate a mechanism to test the model's quality and validity. In this way, the dataset was divided into training, validation, and testing with a ratio of 70%-15%-15%, respectively. Training and validation are the data samples to train the models and to evaluate the model fit while tuning model hyperparameters. The test dataset is a sample of data used to provide an unbiased assessment of the final model in the training data. It is important to note that all the rows in the dataset have a temporal order and that the division in training, validation and testing took this into account to avoid training the model with data from the future (data leakage phenomenon). Table 6 shows the train, validation, and test split with the respective ratios and date ranges that each set involves.

Table 6: Train, validation, and test split.

Dataset	Ratio (%)	Date Range
Train	70	21/05/2018 to 02/05/2019
Validation	15	03/05/2019 to 22/07/2019
Test	15	23/07/2019 to 30/09/2019

The classification algorithms considered were: Gaussian Naïve Bayes (GNB), Logistic Regression (LR), Linear Support Vector Classifier (LinearSVC), Random Forest (RF), eXtreme Gradient Boosting (XGBoost) and Light Gradient Boosting Machine (LightGBM). The algorithm selection was based in three aspects: easy understanding of the techniques, engine efficiency, and the fact of being possible training the models with large datasets and high dimensional data. With this, LR and NB are two of the most uncomplicated algorithms to understand. Concerning the efficiency of the engine, all algorithms comply with this parameter, except for RF because of the computational complexity (Herrera Cordova et al. 2019). The last principle, concerning large datasets and high dimensional data, is supported by all algorithms excluding LinearSVC.

The next step was to build and optimize the model. There is a variety of possible scalers to use and balancing methods to try with each algorithm mentioned above. The scalers are helpful with the outliers and to ensure a common scale without distorting differences in the many different ranges of values. The ones experimented in this work were RobustScaler and StandardScaler. Since the dataset shows signs of imbalance (Section 3.4.5), many balancing options were considered, such as Random Oversampling, Random Undersampling, both random over and undersampling, oversampling with SMOTE, and class weights. As explained in Section 2.5.2, random oversampling means to randomly replicate examples from the minority class in the training dataset and random undersampling deletes examples from the majority class. The first one can lead to overfitting, and the last is prone to losing information invaluable to a model. To avoid the random sampling techniques inconveniences, SMOTE was also considered. It synthesizes new minority instances between existing (real) minority instances, increasing the size of the training data set and the variety, with

the inconvenience of falsifying reality. Lastly, to combat the dataset imbalance, class weights were also studied. Combining the six classification algorithms with this variety of scalers and sampling methods, the number of possible combinations to test is high.

In order to reduce the models to optimize, first, it was used a combination of the algorithms (GNB, LR, LinearSVC, RF, XGBoost, LightGBM) with different scalers (RobustScaler, StandardScaler) and balancing techniques (oversampling, undersampling, over + undersampling, balanced model class weight, and none). All the algorithms were tested with the default parameters at this first optimization phase, except for RF, XGBoost, and LightGBM's number of estimators, which was increased to 200 (the default 100 could inhibit the performance of the model since there is a large amount of data and features). All the 106 combinations evaluated (GBN does not support class weight) are summarized in Table 7.

Table 7: First phase of model optimization.

Scalers	**Algorithms**	**Balancing Techniques**
StandardScaler	GBN	Random Oversampling 35%
RobustScaler	LR	Random Oversampling 40%
	LinearSVC	Random Undersampling 35%
	RF	Random Undersampling 40%
	XGBoost	Oversampling w/ SMOTE 35%
	LightGBM	Oversampling w/ SMOTE 40%
		Random Over+Undersampling
		Balanced model class weights
		None
Total: 106 combinations		

This phase included a more in-depth study to analyze the impact of using different scalers and sampling methods in the different classification algorithms chosen. After this analysis (Section 3.6.1), it was decided that only the top 3 models were going to be optimized. The hyperparameter optimization phase, also known as tuning, is the process of finding the best machine learning model hyperparameters for a given dataset. Two conventional methods for hyperparameter tuning are GridSearchCV and RandomSearchCV, where a list of parameters is defined and evaluated (preferably on a validation set), winning the combination which yields the best performance. With large and high dimensional datasets, GridSearchCV would significantly slow down computation time and be very costly, making an inefficient method for this particular study due to the numerous combinations (Bergstra & Bengio 2012). Therefore, for the second optimization phase, RandomSearchCV was used for 100 iterations, repeating the process for each model around ten times. Once the best hyperparameters were found, the final model was tested on the test set, an untouched dataset to avoid a biased assessment.

As for the metrics to be used for this study, it was necessary to consider the imbalance of the dataset. Because most of the standard metrics that are widely used assume a balanced class distri-

bution, and because typically not all classes, and therefore, not all prediction errors, are equal for imbalanced classification, the metrics considered for evaluation were: AUPRC (Area Under Precision-Recall Curve), AUROC (Area Under the Receiver Operating Characteristic), Precision and Recall. The PR (precision-recall) curve can focus only on the class of interest, whereas the ROC curve covers both classes, and that is the main reason why it was decided that the main metric would be AUPRC. The well-known accuracy is not suitable because the impact of the least represented, but more crucial examples is reduced compared to that of the majority class (Branco et al. 2016). In this type of study, the results can be applied in a real context in the company. So, the trade-off between precision and recall is commonly analyzed, adapting the threshold as needed (Fawcett 2006): a higher recall or a greater precision. The classification model threshold, by default 0.5, can be lowered or increased in order to achieve the company's objectives. A higher recall makes it possible to avoid false negatives, which, in a business context, would mean picking up more possible cases of having an installation problem, requiring a second screening action on the part of the company to verify the true positives of this population. On the other hand, a greater precision means that it only predicts an installation problem if the results are confident, that is, the customers selected by the model for candidates for installation problems are almost certain. In this case, better precision is not intended, but a better recall. To increase recall, and keeping in mind a trade-off between both precision and recall, it is possible to change the threshold in the final model and choose which is the best value instead of the typical 0.5 (Fawcett 2006).

3.6 Evaluation

The focus of this section is to assess the degree to which the model meets the business objectives and seek to determine if there is some business reason why the model is deficient. The first two subsections show the results of the impact of the balancing and scaling methods, and the optimization of hyperparameters. The final results and best model is presented in Subsection 3.6.3.

3.6.1 Balancing/Scaling Study

To ensure that the best models were chosen in the first optimization phase, and to understand the impact of the different scalers, balancing methods, and classification algorithms on the data, it was carried out a comparative study of the 106 combinations in Table 7. As already mentioned, all the algorithms were tested with the default parameters, except for RF, XGBoost, and LightGBM's number of estimators, which were increased to 200 (the default 100 could inhibit the performance of the model since there is a large amount of data and features).

The first table (Table 8) shows, for each of the scalers used, the top five combinations obtained

according to the AUPRC metric (which coincides with the higher AUROC as well). It is visible that, regardless of the scaling method used, the benefited algorithms and the balancing techniques are the same, in addition to the fact that the AUPRC differences are not significant. Clearly, the Logistic Regression, Linear SVC, LightGBM, and Gaussian Naïve Bayes algorithms are the most favored by scalers, leaving out XGBoost and RF that do not make the top. Such a phenomenon makes sense since scalers are essential for machine learning algorithms that calculate distances between data. This is the case with LR and LinearSVC. If not scale, the feature with a higher value range starts dominating when calculating distances. As for balancing techniques, the models seem to enjoy more of class weights and random oversampling.

Table 8: Top 5 combinations obtained for each scaler, according to AUPRC metric.

Scaler	Algorithm	Balancing Technique	AUPRC	AUROC
Standard	LR	Balanced model class weights	0.0871	0.5802
	LinearSVC	Balanced model class weights	0.0870	0.5799
	LightGBM	Balanced model class weights	0.0859	0.5690
	GNB	Random Oversampling 35%	0.0811	0.5494
	GNB	Random Oversampling 40%	0.0808	0.5477
Robust	LR	Balanced model class weights	0.0870	0.5798
	LinearSVC	Balanced model class weights	0.0869	0.5796
	LightGBM	Balanced model class weights	0.0860	0.5700
	GNB	Random Oversampling 35%	0.0825	0.5581
	GNB	Random Oversampling 40%	0.0822	0.5574

Table 9 presents the top three models for each of the balancing techniques experimented. According to the AUPRC metric, the class weights parameter set to 'balanced' reached a higher overall score in the models. Within the same technique, for example, using oversampling with SMOTE with 35% or 40%, there is not a significant difference in score between the top models, and the percentage variations have not proved to be very useful. The use of no balancing technique ('None'), does not look beneficial for either the algorithms and scalers, and the AUROC metric is also slightly under average. The algorithms Gaussian Naïve Bayes, Logistic Regression, and LightGBM seem to be the ones that benefit most from sampling techniques. For the scalers, there is no apparent correlation between the balancing techniques and Robust or StandardScaler, once the AUPRC metric of the combinations is similar in each balancing technique.

Table 9: Top 3 combinations obtained for each balancing technique, according to AUPRC metric.

Balancing Technique	Algorithm	Scaler	AUPRC	AUROC
Random Oversampling 35%	GNB	Standard	0.0811	0.5494
	GNB	Robust	0.0782	0.5335
	LR	Standard	0.0767	0.5142
Random Oversampling 40%	GNB	Standard	0.0808	0.5477
	GNB	Robust	0.0781	0.5330
	XGBoost	Standard	0.0780	0.5166
Random Undersampling 35%	GNB	Standard	0.0795	0.5395
	LightGBM	Standard	0.0785	0.5190
	GNB	Robust	0.0785	0.5349
Random Undersampling 40%	LightGBM	Standard	0.0802	0.5243
	GNB	Standard	0.0796	0.5409
	GNB	Robust	0.0790	0.5374
Oversampling w/ SMOTE 35%	GNB	Robust	0.0825	0.5581
	GNB	Standard	0.0774	0.5322
	LR	Robust	0.0754	0.5089
Oversampling w/ SMOTE 40%	GNB	Robust	0.0822	0.5574
	GNB	Standard	0.0774	0.5323
	LR	Robust	0.0762	0.5140
Random Over+Undersampling	GNB	Standard	0.0798	0.5424
	GNB	Robust	0.0784	0.5341
	LightGBM	Standard	0.0732	0.5008
Balanced model class weights	LR	Standard	0.0871	0.5802
	LinearSVC	Standard	0.0870	0.5799
	LR	Robust	0.0870	0.5798
None	GNB	Standard	0.0800	0.5420
	LR	Standard	0.0800	0.5000
	GNB	Robust	0.0790	0.5350

The last table (Table 10) provides an overview by classification algorithm, where the top three combinations for each are presented by AUPRC metric. It is expected that the tree-based algorithms are the least favored by the scaling/balancing process, namely, RF, XGBoost, and LightGBM. This makes sense for two reasons: first, decision trees and ensemble methods do not require feature scaling to be performed as they are not sensitive to the variance in the data. The outliers do not impact the data classification as the data is split using scores, which are calculated using the homogeneity of the resultant data points. Second, decision trees are less sensitive to class imbalance problems, as they handle imbalanced data well, not being necessary any balancing methods. In reality, it is visible that the LightGBM algorithm achieves a higher overall AUPRC, even higher than XGBoost. This is because it produces much more complex trees by following a leaf-wise split approach rather than a level-wise approach, which is the main factor in achieving higher scores. Besides, LightGBM uses a novel technique of Gradient-based One-Side Sampling (GOSS) to filter out the data instances for finding a split value while XGBoost uses a pre-sorted algorithm

and histogram-based algorithm for computing the best split. This algorithm was not the one that presented the highest AUPRC in this phase, but, as shown after the tuning section (Section 3.6.3), it was the one that achieved the best performance.

Moving to the scalers, once again, it seems to be no particular connection between the Robust/StandardScaler and the algorithms, as the AUPRC metric remains almost the same. Concerning the balancing techniques, the synthetic methods like SMOTE did not bring any value compared to the traditional random over/undersampling.

Table 10: Top 3 combinations obtained for each algorithm, according to AUPRC metric.

Algorithm	Scaler	Balancing Technique	AUPRC	AUROC
GNB	Robust	Oversampling w/ SMOTE 35%	0.0825	0.5581
	Robust	Oversampling w/ SMOTE 40%	0.0822	0.5574
	Standard	Random Oversampling 35%	0.0811	0.5494
LR	Standard	Balanced model class weights	0.0871	0.5802
	Robust	Balanced model class weights	0.0870	0.5798
	Standard	None	0.0800	0.5000
LinearSVC	Standard	Balanced model class weights	0.0870	0.5799
	Robust	Balanced model class weights	0.0869	0.5796
	Robust	Random Undersampling 40%	0.0781	0.5178
RF	Standard	Random Undersampling 40%	0.0765	0.5139
	Robust	Random Undersampling 40%	0.0764	0.5130
	Standard	Random Undersampling 35%	0.0750	0.5076
XGBoost	Standard	Random Undersampling 40%	0.0780	0.5168
	Standard	Random Oversampling 40%	0.0780	0.5166
	Robust	Random Undersampling 40%	0.0776	0.5161
LightGBM	Robust	Balanced model class weights	0.0860	0.5700
	Standard	Balanced model class weights	0.0860	0.5690
	Standard	Random Undersampling 40%	0.0802	0.5243

In conclusion, the first optimization phase was intended to reduce the number of models to be optimized. Thus, algorithms were combined with different scalers and sampling methods in order to make a pre-selection. Even though that, with tuning, all the algorithms would benefit, a first screening phase was necessary because optimizing 106 models would not be reasonable. With this, completed the study by scaler, balancing technique, and algorithm, the top three models are summarized in Table 11, ranked by the highest AUPRC, which coincides with the higher AUROC as well. It is worth noting that, before LightGBM, the LR and LinearSVC algorithms appeared with a (small) larger AUPRC, using balanced model class weights and the RobustScaler. Even so, it was decided to opt for the next combination in the table, with the LightGBM algorithm, since it is the one that is one of the most favored with the tuning process (next optimization phase).

Table 11: First optimization phase - best models obtained.

Scaler	Algorithm	Balancing Technique	AUPRC	AUROC
Standard	LR	Balanced model class weights	0.0871	0.5802
Standard	LinearSVC	Balanced model class weights	0.0870	0.5799
Robust	LightGBM	Balanced model class weights	0.0860	0.5700

Hence, it was concluded that class weight would be used as an optimization parameter (instead of over/undersampling) and that each model would have specific weights to achieve the highest performance. For the LR and LinearSVC algorithms, the StandardScaler was chosen, while the LightGBM obtained better results with the RobustScaler. As seen in the previous tables, the use of scalers does not appear to alter AUPRC. Even so, the combinations were chosen where this metric was slightly higher. For the next phase, RandomSearchCV was used in the training and validation dataset. Each of the random searches was performed with the appropriate hyperparameters for each model and evaluation metric AUPRC.

3.6.2 Hyperparameter Optimization

After finding the best models (without tuning), it was decided that only the best three would be optimized. The hyperparameter optimization phase (or tuning) refers to the process of finding the best machine learning model hyperparameters automatically for a given dataset. The hyperparameters are all the parameters of a particular model which are not updated during the learning phase and are used to configure either the model or the algorithm used to lower the cost function. Two conventional methods for hyperparameter tuning are GridSearchCV and RandomSearchCV, where a list of parameters is defined and evaluated (preferably on a validation set), winning the combination which yields the best performance (on a certain metric). With large and high dimensional datasets, GridSearchCV would significantly slow down computation time and be very costly, making an inefficient method for this particular study due to the numerous combinations (Bergstra & Bengio 2012). For example, searching 10 different parameter values for each of 5 parameters will require 100,000 iterations, and a simple addition of one parameter to optimize would induce 1,000,000 rounds. The choice of random search may not be representative for all the range of the parameters, but as the sample or dataset size grows, the chances of this happening get smaller and smaller.

With this, the metric chosen to evaluate the models between rounds was *average_precision*, which coincides with the AUPRC. Since the data have a temporal order and is divided into training, validation and testing, cross-validation (the default evaluation method) could not be used. Instead, it was defined that the training would always be tested in the validation set. Table 12 presents a summary of the algorithms that have been optimized, along with the respective hyperparameters and ranges tested. The optimal parameters found are also detailed. These algorithms were tuned along with the respective scalers and balancing techniques, derived from the previous section. Once

the best balancing technique was balanced class weights (instead of over/undersampling), this was used as an optimization parameter (for target value 1) and that each model would have specific weights to achieve the highest performance.

Table 12: Summary of the models used during the hyperparameter optimization phase, along with the hyperparameters and respective tested ranges.

Model	Hyperparameter	Range	Best Value
LR + StandardScaler	penalty	['l2']	'l2'
	C	[0.01, 0.1, 1, 10, 50]	10
	class_weight	[5, 6, 7, 8, 9, 10, 13, 15]	13
	solver	['newton-cg', 'lbfgs', 'liblinear', 'sag']	'sag'
LinearSVC + StandardScaler	C	[0.01, 0.1, 1, 10, 50]	0.01
	class_weight	[5, 6, 7, 8, 9, 10, 13, 15]	5
LightGBM + RobustScaler	min_data_in_leaf	sp_randint(5, 20)	16
	max_depth	sp_randint(10, 20)	15
	num_leaves	sp_randint(20, 100)	85
	n_estimators	sp_randint(150, 500)	238
	bagging_freq	[0, 2, 4, 6, 8]	4
	min_split_gain	[0.5, 0.7, 3, 5, 7, 9]	0.7
	reg_alpha	[0, 0.4, 0.8, 1.3, 2, 5]	0
	reg_lambda	[0, 0.4, 0.8, 1.3, 2, 5]	1.3
	class_weight	[5, 6, 7, 8, 9, 10, 13, 15]	8
	feature_fraction	[0.4, 0.6, 0.7, 0.8, 0.9, 1.0]	0.4
	bagging_fraction	[0.80, 0.85, 0.90, 0.95, 1.00]	0.95
	learning_rate	[1e-2, 1e-3, 3e-3, 6e-3, 1e-4]	1e-2
	min_child_weight	[1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 20, 30]	1

For each model, RandomSearchCV was used for 100 iterations, repeating the process around 10 times.

Starting with Logistic Regression, and in addition to class_weight, the penalty, C and solver parameters were optimized. The l2 penalty is supported by all solvers, and is a regularization used to regulate overfitting, just like the C value (in this case, the lower the C, the greater the regularization).

LinearSVC is an algorithm with few possible optimization parameters, so it was decided to iterate only through the values of C and class_weight.

As for LightGBM, there are a large number of parameters that, well combined, allow to deal with overfitting, obtaining better accuracy or achieving a faster speed. For combat overfitting, it is recommended to use a low num_leaves value and to achieve greater accuracy, a high value is recommended. As high accuracy is not the objective of this work, a maximum number of 100 leaves was defined, which is a relatively low value. Like num_leaves, the depth of the tree was tested to avoid growing a deep tree, and also the min_data_in_leaf was controlled to avoid overfitting. Following the same line of reasoning, and to test different complexities of the tree, for

the variable min_child_weight an interval was defined with equally small and large values. The min_child_weight variable is the minimum weight required in order to create a new node in the tree. A smaller min_child_weight allows the algorithm to create children that correspond to fewer samples, thus allowing for more complex trees, but again, more likely to overfit. The feature_fraction parameter, which controls the ratio of features of the model to be used, was studied as a way to control overfitting and to achieve a better execution speed. To also achieve this, bagging was used through bagging_freq and bagging_fraction. For the learning rate, which defines how quickly the model updates the concepts it has learned, a range diversified enough was created for the best ratio to be chosen. The well-known parameter n_estimators that defines the number of trees to build before taking the maximum voting or averages of predictions must be increased, taking into account the dimensionality of the dataset. Therefore, an interval between 150 and 500 estimators was defined. Finally, to control the regularization, the variables reg_alpha, reg_lambda and min_split_gain were used, testing different values.

The following section compares the performance of these three optimized models, verifying which one achieved the highest AUPRC.

3.6.3 Model Results

The first optimization phase was intended to reduce the number of models to be optimized, as mentioned in Section 3.6.1. Thus, algorithms were combined with different scalers and sampling methods in order to make a pre-selection. The top 3 models from this first analysis were tuned, and RandomSearchCV was used in the training and validation set. Each of the random searches was performed with the appropriate hyperparameters for each model and evaluation metric AUPRC. The achieved results are shown below in Table 13.

Table 13: Second optimization phase - best results achieved.

Scaler	Algorithm	Balancing Technique	AUPRC	AUROC
Standard	LR	Class Weight {1:13}	0.1102	0.6189
Standard	LinearSVC	Class Weight {1:5}	0.1076	0.6150
Robust	LightGBM	Class Weight {1:8}	**0.1108**	0.6206

The algorithm that presented the best results was LightGBM with RobustScaler and Class Weight as a balancing method. Contrary to what happened before, where the automatically balanced model class weights were found to be better, the model found a better result for a weight of 8 for the minority class. With this, it was achieved an AUPRC $\approx$ 0.11, which coincides with the higher AUROC $\approx$ 0.62. The improvement of the optimization phase was approximately 37% with respect to the AUPRC. Figure 26 shows the PR curve obtained for the best model. The plot of the precision-recall curve highlights that the model is barely above the no skill line for most recall values.

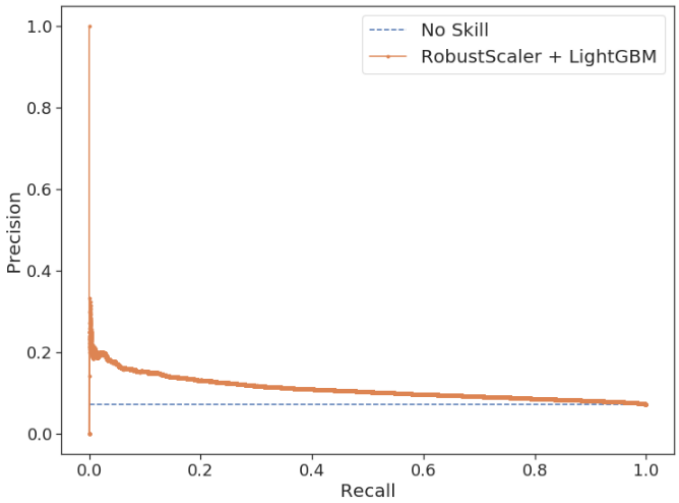


Figure 26: Comparison of the precision curve for different recall values between the best model and a no skill model.

The model threshold values in Figure 27 shows that the default value 0.5 needs to be lowered in order to maximize recall. The main goal of increasing recall is to avoid false negatives, meaning that the model chooses more cases of having installation problems, requiring a second screening action from the company to verify the true positives of this population. In this case, the selected threshold value that maximizes recall, and that keeps a good trade-off between precision and recall, is approximately 0.43.

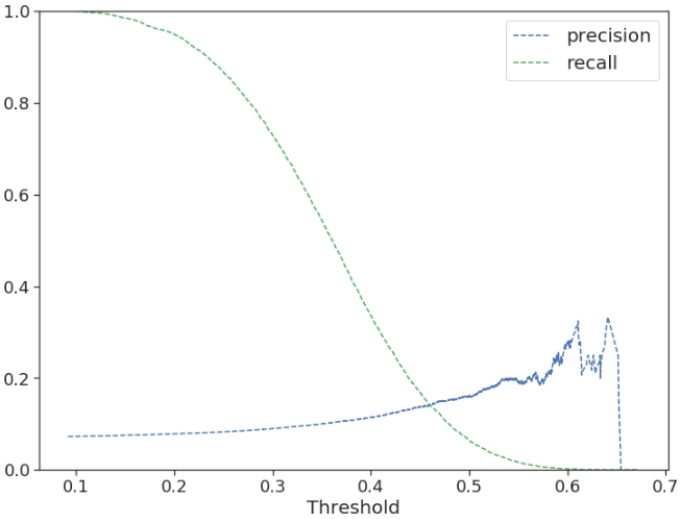


Figure 27: Precision and recall curves for different thresholds for the best model.

The top ten most relevant features from the model are presented in Figure 28. The variables are mainly related to the location of the customer and the technology installed, as well as features associated with the service time. These features can serve as guidelines for future implementations.

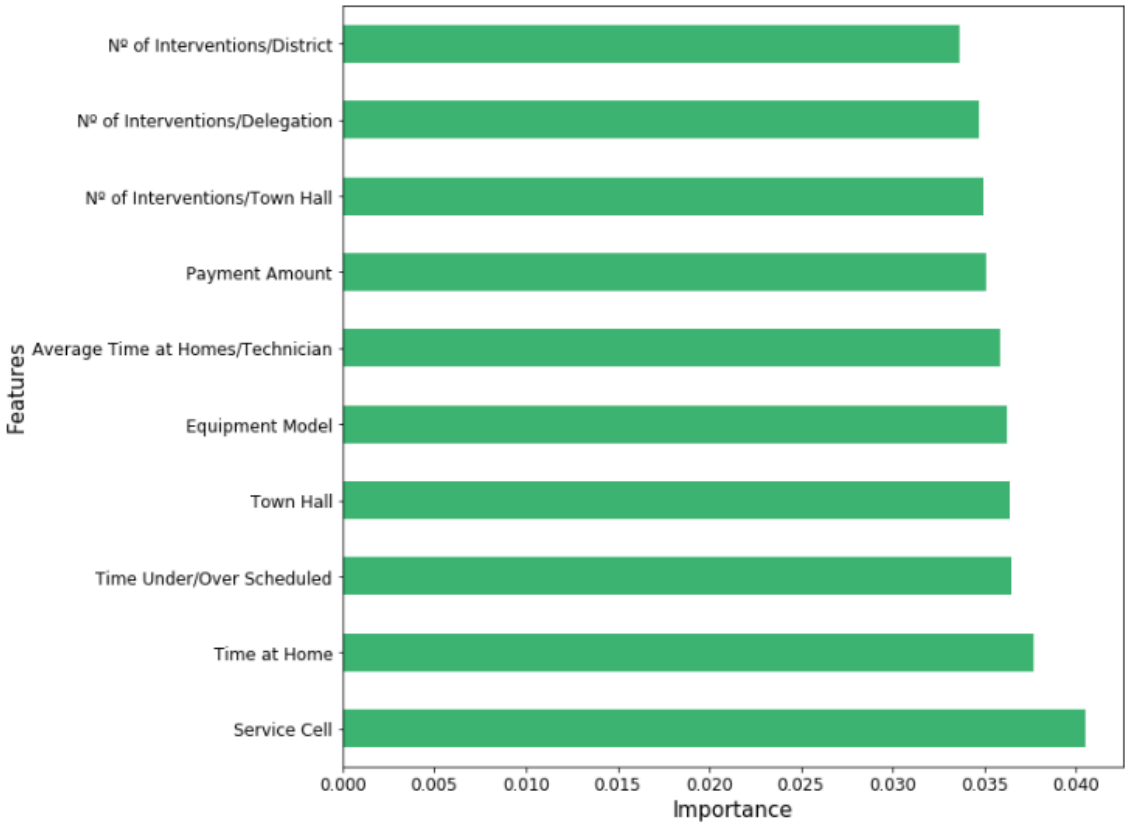


Figure 28: Input features ranked by their importance.

Finally, with the best hyperparameters and the optimal threshold found, the final model was tested on the test set, confirming the results achieved in the validation set, and obtaining the confusion matrix in Table 14. Even with the adjustment of the threshold, and taking into account the results obtained from AUPRC, a trade-off between precision and recall that tries to maximize the recall will not prevent the existence of false positives and negatives. The existence of a large number of these cases, and the low AUPRC value, indicate that further work is needed.

Table 14: Confusion matrix of the best model with respect to the test set.

		Predicted	
		Yes	No
Actual	Yes	685	4639
	No	10384	69229

3.7 Discussion

Several conclusions have already been drawn throughout the previous evaluation section (Section 3.6), where the benefit of data balancing/scaling and hyperparameter optimization was discussed. However, after obtaining the model results, some important points still need to be clarified. The first one is to understand the cause of the achieved results. In fact, several measures have been taken to combat data imbalance, and it has always been a concern throughout the process to ensure data quality and adequate metrics for evaluation. Yet, the distribution of examples across the known classes was proven to be severely biased. The lack of a large enough population of customers with problems resulting from the installation process meant that the model could not predict, with high performance, who would suffer from these technical obstacles. The related studies presented in Section 2.7 have demonstrated to be of great value, not only in the survey of ML concepts and models but also in the notion of imbalance and the use of techniques such as threshold-optimization.

The problems arising from installations, and the need for maintenance interventions after them, are, in fact, a problem with monetary consequences for the company and a handicap to higher customer satisfaction. Although not perfect, the current solution becomes feasible with the aim of the higher recall and the avoidance of false negatives, which means that the telecommunications company would have to perform a screening action with the clients to verify the true positives of this population. The benefits resulting from this solution are a more efficient service through greater attention and monitoring of customers. Nonetheless, the lack of excellent results also brings contributions and awareness to the company and indicates what must be done to solve this or similar problems. In this particular case, the solution may be to divide the problem into parts, or, more generally, collect more data characteristic of the customer and the installation moment, especially those variables that have shown more relevance in feature importance. If these data measures were taken, subsequently, this or other problems with a similar data mining goal will also be solved in the future.

This being said, it is just remaining to answer the main question of this investigation and assess whether the three objectives were accomplished. Starting with the three subsequent goals, the first one consisted of knowing in advance who would complain and be aware of the customer's probable inconveniences, having a proactive action and dialogue with the client before the complaint call occurs, responding efficiently to the situation. With this solution, this objective is achieved because it is possible to provide greater attention and monitor customers, which allows to resolve problems more quickly and act after installation and before the complaint occurs.

The second and third subsequent goals subsisted of knowing that a certain person will have an installation problem and applying a personalized service (suitable assistance, equipment, or technician), culminating in better customer experience and higher installation service efficiency. This also

increases profit for the company, since the number of problems also diminishes the time/equipment spent (number of trips to the customer's home or expensive equipment replacement). The cost reduction was not much achieved, since it would be necessary for the model to indicate, with greater confidence, who would be likely to have maintenance due to an installation problem. With this certainty, it would be possible not only to have a proactive action and dialogue with the client before the complaint call occurs but to act before the installation and according to each client's conditions, providing personalized service in general.

All the work carried out was more than satisfactory and thorough, and the company was given tools to improve customer satisfaction and be aware of its limits. Despite that, the model could not predict, with the wished high performance, who would suffer from these installation issues. Even so, it is considered that most of the objectives have been met and that useful information was obtained for the telecommunications company.

4. FINAL CONSIDERATIONS

This chapter discusses the conclusions drawn at the end of this study, as well as the recommendations for future work enhancements.

4.1 Conclusion

Improving customer experience is crucial in any industry, especially in telecommunications, where competition is a constant factor. Today, all telecommunications companies rely on the massive amount of data generated daily to get to know the customer or study their behavior, thus creating new effective strategies for their business. In this work, it was performed research in the process of installing new services, which can be an event that raises technical problems and, consequently, a possible degradation in the user experience.

This work's main objective was to create a predictive model that, through all the available data history, could predict with high performance which customers would contact the customer service with problems from the installation process. With this, the company can anticipate the cases that will be problematic and reduce the number of negative experiences. By the past cases, knowing that a certain person will have an installation problem, a personalized service can be applied (suitable assistance, equipment, or technician), culminating in better customer experience. Reducing the number of problems also diminishes the time/equipment spent (number of trips to the customer's home or expensive equipment replacement), resulting in increased profit for the company. The study was conducted using real data from a Portuguese telecommunications company, and the analysis was conducted resorting to the CRISP-DM methodology. The elaborated work went through analyzing the data in-depth and treating it to obtain a clean dataset with new underlying features. The modeling of the problem took into account the significant imbalance of the data, so a study of the impact of different balancing techniques and scalers was performed, and appropriate metrics biased towards the models' performance on these rare cases were used.

The task revealed to be of extreme difficulty, as shown in the rather small AUPRC and AUROC scores (0.11 and 0.62, respectively). The significant imbalance of the data proved problematic, and it was necessary to recur to precision-recall threshold optimization. The best trade-off between precision and recall was found with a threshold model of 0.43, in order to maximize the recall and not lose all precision. A higher recall makes it possible to avoid false negatives, which means that the telecommunications company would have to perform a screening action with the clients to verify the true positives of this population.

4.2 Future Work

The existence of a high number of false positives and negatives, as well as the low value of AUPRC, indicate that further work should be conducted. The exhaustive model experimentation performed suggests a need to obtain more data that could improve models' performance. The exploration of other machine learning models is also one of the tasks to be done, such as neural networks, which could lead to superior results. Besides, future research will focus on improving model tuning and determining if the predictions could be more detailed concerning the predicted time of occurrence and the model variables that have shown more importance. Lastly, it would be hard to carry out the last phase of CRISP-DM and deploy the solution found, since it would be necessary to make a balance between the results obtained (which were not brilliant) and the data processing time required for this solution.

REFERENCES

Ahmad, A. K., Jafar, A. & Aljoumaa, K. (2019), 'Customer churn prediction in telecom using machine learning in big data platform', *Journal of Big Data* **6**(1).

Aris-Brosou, S., Kim, J., Li, L. & Liu, H. (2018), 'Predicting the reasons of customer complaints: A first step toward anticipating quality issues of in vitro diagnostics assays with machine learning', *JMIR Med Inform* . [online] 6(2). doi: 10.2196/medinform.9960.

Bergstra, J. & Bengio, Y. (2012), 'Random search for hyper-parameter optimization', *Journal of Machine Learning Research* **13**(10), 281–305.

Branco, P., Torgo, L. & Ribeiro, R. P. (2016), 'A survey of predictive modeling on imbalanced domains', *ACM Comput. Surv.* **49**(2).

Breiman, L. (1997), '*Arcing the edge*'. Technical Report 486. Statistics Department, University of California, Berkeley. [online] Available at: `https://statistics.berkeley.edu/sites/default/files/tech-reports/486.pdf` [Accessed 16 May. 2020].

Brownlee, J. (2017), *'What is the Difference Between Test and Validation Datasets?'*. [online] Machine Learning Mastery. Available at: `https://machinelearningmastery.com/difference-test-validation-datasets/` [Accessed 15 Jun. 2020].

Chawla, N. V. (2010), *Data Mining for Imbalanced Datasets: An Overview*, Springer US, Boston, MA, pp. 875–886.

Chawla, N. V., Bowyer, K. W., Hall, L. O. & Kegelmeyer, W. P. (2002), 'Smote: Synthetic minority over-sampling technique', *J. Artif. Int. Res.* **16**(1), 321–357.

Chen, T. & Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '16, Association for Computing Machinery, New York, NY, USA, p. 785–794.

Choi, C. (2018), Predicting customer complaints in mobile telecom industry using machine learning algorithms, Master's thesis, Purdue University, Indiana. Available at: `https://search.proquest.com/docview/2054017508` [Accessed 02 Jan. 2020].

Claesen, M. & Moor, B. D. (2015), 'Hyperparameter search in machine learning', *ArXiv* **abs/1502.02127**.

Code, V. S. (2020), '*Getting Started*'. [online] Available at: `https://code.visualstudio.com/docs` [Accessed 27 Jan. 2020].

DECO (2019), '*Telecomunicações sempre no topo das reclamações*'. [online] Available at: https://www.deco.proteste.pt/institucionalemedia/imprensa/comunicados/2019/telecons-1705 [Accessed 29 Dec. 2019].

Fawcett, T. (2006), 'An introduction to roc analysis', *Pattern Recognition Letters* **27**(8), 861 – 874. ROC Analysis in Pattern Recognition.

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996), 'From data mining to knowledge discovery in databases', *AI Magazine* **17**(3), 37–54. doi: 10.1609/aimag.v17i3.1230.

Flach, P. (2012), *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*, Cambridge: Cambridge University Press.

García, S., Luengo, J. & Herrera, F. (2015), *Data Preprocessing in Data Mining*, Vol. 72 of *Intelligent Systems Reference Library*, Springer.

Hamerly, G. & Elkan, C. (2001), Bayesian approaches to failure prediction for disk drives, *in* 'Proceedings of the Eighteenth International Conference on Machine Learning', ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 202–209.

Han, J., Kamber, M. & Pei, J. (2011), *Data Mining: Concepts and Techniques*, 3 edn, Morgan Kaufmann.

Herrera Cordova, V., Khoshgoftaar, T., Villanustre, F. & Furht, B. (2019), 'Random forest implementation and optimization for big data analytics on lexisnexis's high performance computing cluster platform', *Journal of Big Data* **6**.

Hoare, J. (2020), '*Gradient Boosting Explained – The Coolest Kid on The Machine Learning Block.*'. [online] Displayr. Available at: https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/ [Accessed 20 May 2020].

Hosmer, D. & Lemeshow, S. (2000), *Applied Logistic Regression*, 2 edn, Wiley.

Jupyter (2019), '*Homepage*'. [online] Available at: https://jupyter.org/ [Accessed 27 Jan. 2020].

Kaur, P. & Gosain, A. (2018), Comparing the behavior of oversampling and undersampling approach of class imbalance learning by combining class imbalance problem with noise, *in* A. K. Saini, A. K. Nayak & R. K. Vyas, eds, 'ICT Based Innovations', Springer Singapore, Singapore, pp. 23–30.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q. & Liu, T.-Y. (2017), Lightgbm: A highly efficient gradient boosting decision tree, *in* I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, eds, 'Advances in Neural Information Processing Systems 30', Curran Associates, Inc., pp. 3146–3154.

Kitchenham, B. & Charters, S. (2007), Guidelines for performing systematic literature reviews in software engineering, Technical report, EBSE 2007-001, Keele University and Durham University.

Kohavi, R. & Provost, F. (1998), 'Glossary of terms', *Machine Learning* . [online] 30(2-3), 271-274. Available at: http://deeplearning.lipingyang.org/wp-content/uploads/2016/12/Glossary-of-Terms-Journal-of-Machine-Learning.pdf [Accessed 26 Dec. 2019].

Kundu, A. (2018), '*6 Key Tenets To Get Started with Enterprise AI*'. [image] Medium. Available at: https://medium.com/stretch-magazine/5-key-tenets-to-get-started-with-enterprise-ai-d880ae7fe543 [Accessed 26 Dec. 2019].

Lemaitre, G., Nogueira, F., Oliveira, D. & Aridas, C. (2017*a*), '*RandomOverSample*'. [online] Imbalanced-learn documentation. Available at: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.RandomOverSampler.html [Accessed 16 Jun. 2020].

Lemaitre, G., Nogueira, F., Oliveira, D. & Aridas, C. (2017*b*), '*RandomUnderSampler*'. [online] Imbalanced-learn documentation. Available at: https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.under_sampling.RandomUnderSampler.html/ [Accessed 16 Jun. 2020].

LinkedIn (2018), '*Jupyter Notebook in Visual Studio Code*'. [online] Available at: https://www.linkedin.com/pulse/jupyter-notebook-visual-studio-code-bikash-sundaray/?utm_campaign=News&utm_medium=Community&utm_source=DataCamp.com [Accessed 27 Jan. 2020].

London, I. (2016), '*Encoding cyclical continuous features - 24-hour time*'. [online] Github. Available at: https://ianlondon.github.io/blog/encoding-cyclical-features-24hour-time/ [Accessed 25 Jun. 2020].

Python (2020), '*What is Python? Executive Summary*'. [online] Available at: https://www.python.org/doc/essays/blurb/ [Accessed 27 Jan. 2020].

Rish, I. (2001), 'An empirical study of the naïve bayes classifier', *IJCAI 2001 Work Empir Methods Artif Intell* **3**.

Russell, S. J. & Norvig, P. (2010), *Artificial Intelligence: A Modern Approach*, 3 edn, Prentice Hall.

Saltz, J., Shamshurin, I. & Crowston, K. (2017), Comparing data science project management methodologies via a controlled experiment.

School, I. P. (2017), *'Why Python is the preferred language for Machine Learning?'*. [online] Available at: http://ivyproschool.com/blog/2017/08/21/why-python-is-the-preferred-language-for-machine-learning/ [Accessed 27 Jan. 2020].

Scikit-Learn (2020*a*), '*Compare the effect of different scalers on data with outliers*'. [online] Available at: https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html [Accessed 15 May 2020].

Scikit-Learn (2020*b*), '*OneHotEncoder*'. [online] Available at: http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html [Accessed 25 May 2020].

Shearer, C. (2000), 'The crisp-dm model: The new blueprint for data mining', *Journal of Data Warehousing* . [online] 5(4), pp. 13-22. Available at: https://mineracaodedados.files.wordpress.com/2012/04/the-crisp-dm-model-the-new-blueprint-for-data-mining-shearer-colin.pdf [Accessed 29 Dec. 2019].

Solutions, S. (2020), *'What is Logistic Regression?'*. [online] Available at: https://www.statisticssolutions.com/what-is-logistic-regression/ [Accessed 10 Jun. 2020].

Stanford (2019), '*Machine Learning*'. [online] Available at: http://mlclass.stanford.edu/ [Accessed 23 Dec. 2019].

Statistics How To (2019), '*Receiver Operating Characteristic (ROC) Curve: Definition, Example*'. [image] Available at: https://www.statisticshowto.datasciencecentral.com/receiver-operating-characteristic-roc-curve/ [Accessed 27 Dec. 2019].

Vidhya, A. (2017), *''Which algorithm takes the crown: Light GBM vs XGBOOST?'*. [online] Available at: https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/ [Accessed 10 Jun. 2020].

Weiss, G. (2008), Data mining in the telecommunications industry, *in* J. Wang, ed., 'Encyclopedia of Data Warehousing and Mining', 2 edn, Vol. 1, Information Science Publishing, pp. 486–491.

Wikipedia (2019*a*), '*Cross–industry standard process for data mining*'. [image] Available at: https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining#/media/File:CRISP-DM_Process_Diagram.png [Accessed 29 Dec. 2019].

Wikipedia (2019*b*), '*Machine Learning*'. [online] Available at: https://en.wikipedia.org/wiki/Machine_learning#Relation_to_data_mining [Accessed 26 Dec. 2019].

Witten, I. H., Frank, E. & Hall, M. A. (2011), *Data Mining: Practical Machine Learning Tools and Techniques*, 3 edn, Morgan Kaufmann.

Zheng, A. & Casari, A. (2018), *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*, 1st edn, O'Reilly Media, Inc.

APPENDICES

Appendix I - Publications

Anticipating Maintenance in Telecom Installation Processes

Authors: Diana Costa, Carlos Pereira, Hugo Peixoto, and José Machado

Title: Anticipating Maintenance in Telecom Installation Processes

Conference: 21st International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2020

Year of Publication: 2020 (reviewed and accepted for publication)

Abstract: Improving customer experience is crucial in any industry, especially in telecommunications, where competition is a constant factor. Today, all telecommunications companies rely on the massive amount of data generated daily to get to know the customer or study their behavior and thus create new effective strategies for their business. Within the most varied user experiences, the process of installing new services can be an event that raises doubts about their operation, degrade the user experience, or, in extreme cases, lead to maintenance interventions. Therefore, the use of advanced predictive models that can predict such occurrences become vital. With this, the company can anticipate the cases that will be problematic and reduce the number of negative experiences. The main objective of this work is to create a predictive model that, through all the available data history, can predict which customers will contact the customer service with problems derived from the installation process and have a following maintenance intervention. After analyzing an unbalanced dataset with approximately 560K entries from a Portuguese telecommunications company, and resorting to the CRISP-DM methodology for modeling, the best results were found with LightGBM which obtained an AUPRC of 0.11 and AUROC of 0.62. The best trade-off between precision and recall was found with a threshold model of 0.43 in order to maximize recall while still avoiding a large number of false negatives.