



University of Minho
School of Engineering

Vitor José Ribeiro Castro

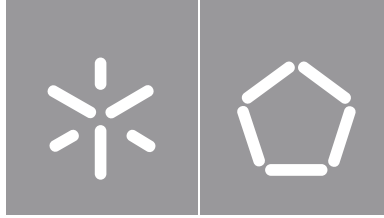
**Benchmarking Deep Learning for
Predicting Telecommunications
Recurring Problems**

**Benchmarking Deep Learning for
Predicting Telecommunications
Recurring Problems**

Vitor Castro

UMinho | 2020

October 2020



University of Minho

School of Engineering

Vitor José Ribeiro Castro

**Benchmarking Deep Learning for
Predicting Telecommunications
Recurring Problems**

Master Dissertation

Integrated Master's Degree in Informatics Engineering

Dissertation supervised by

Professor Doctor Victor Manuel Rodrigues Alves

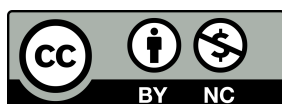
Doctor Carlos Miguel Silva Couto Pereira

COPYRIGHT AND CONDITIONS OF USE BY THIRD PARTIES

This is an academic work that can be used by third parties if internationally accepted rules and good practice with regard to copyright and related rights are respected.

Thus, the present work can be used under the terms of the license indicated below.

In case the user needs permission to be able to make use of the work in conditions not foreseen in the indicated licensing, he should contact the author through the *RepositóriUM* of the University of Minho.



Atribuição-NãoComercial

CC BY-NC

<https://creativecommons.org/licenses/by-nc/4.0/>

Não é o muito saber que sacia e satisfaz a alma, mas o sentir e saborear internamente as coisas.

Santo Inácio de Loyola

ACKNOWLEDGMENTS

AGRADECIMENTOS

This dissertation is the result of a continuous effort over five years. It results not only from a very large individual work, but also from an extremely important support from many other people. In a self-centered world like today's, it is important to remember that we alone are nothing. Thus, too briefly to represent my gratitude, I would like to express my deep thanks to the many parties involved in this journey:

To my maternal grandfather and grandmother, who contributed in large part to my personal growth. The availability, caring, accompaniment, affection and love with which they took care of me was, without a doubt, one of the basic elements to my success. They were present from childhood until the end of university, always trying to understand and advise me as they knew best. I love you.

To my parents, whom I greatly admire, for the simplicity with which they live life together and how they respect each other. They are examples of life, just like their grandparents, without whom my personal development would be impossible. They were also the ones who supported my education, always putting it ahead of everything: they let me be free in choosing my future, and free and responsible for the way I proceeded throughout these years. I love you.

To Marta, who is indescribable in the way she perceives what I feel and what I am; who teaches me so much on an emotional level, and who is a great example of what love for the other can do for a community; who is restless with the world as much or more than I am and without whom the months invested in this thesis would not have the same meaning. I love you.

To Diana, Sérgio and Marcos, for sharing not only knowledge but also life. They have provided me with great learning and are very responsible for the academic success and personal growth I feel. A tight hug.

To my friends, who have been with me since childhood, and who have understood and supported the doubts I have felt throughout my journey in school. Moreover, they shared beautiful moments of friendship and were in many ways anchors. A strong hug.

To CAB and all the companions, and to the Grupo de Jovens de São Sebastião and all those with whom I could have several interesting discussions there. They were very important in my growth and indispensable in the moments of greatest discouragement. A fraternal hug.

To Carlos, supervisor at NOS, who was an excellent colleague, became a friend, and showed

interest and availability throughout this project where I learned a lot. A deep thank you.

To Professor Victor Alves, a teacher who guided my master thesis with availability and kindness. This thanks goes out to all the teachers who are committed and passionate about teaching, with whom I had a great pleasure to share interesting thoughts. Thank you for your dedication and please never stop believing in the power of education.

To the rest of the family, thank you very much for the good examples that were, in several moments of my life, providing the necessary stability for my growth. Thank you.

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

Benchmarking Deep Learning for Predicting Telecommunications Recurring Problems

Nowadays, companies live in a scenario of strong competitiveness. The telecommunications market is not an exception and it is possible to offer a differentiation from competition through better service quality, differentiated support and even better value proposals. With the evolution of technologies, companies have more data about their customers and the usage profile of each one of them. With this information it is possible to establish a better relationship with the customer through a more efficient support service.

The evolution of artificial intelligence and computational power, combined with existing data, allows for several comparisons between different machine learning algorithms. In this dissertation, a prediction model capable of predicting recurrences of contacts with the customer service is proposed. The aim is to predict whether a particular problem reported by the customer will repeat and require a new contact, so that it is possible to correct those problems in advance, making the user experience more pleasant and fluid. In order to achieve the best possible model, different classical machine learning approaches were tested, along with several deep neural network architectures. In recent years, deep neural networks have shown interesting results in several non-tabular applications, therefore being interesting to test them in tabular applications like the one present in this work. TabNet, developed by Google, is a deep neural network adjusted to perform the better in tabular datasets, and was also tested, as it has shown better performance than several neural networks or decision-tree bases algorithms.

The used data were collected by various internal systems, the most important of which being the one related to customer support calls. The customer service, due to its size and complexity, has a system that monitors all calls and their motivations, as well as the parties involved (both operator and customer) and other additional data such as time spent and the call outcome. Data from other systems is related to billing, service usage and customer profile, and is added to help to understand the context of the call.

The model that shown the best results was CatBoost, a decision trees based algorithm, showing an AUC_ROC of 79%, with a Recall of 61% and a Precision of 62%, allowing the identification of about 8,6% of the 3.9 million calls made to the support service as recurrences even before they occur, about 340k cases. In an ideal scenario, all these calls would be avoided, allowing a substantial cost reduction for the company, as well as a consequent increase in customer satisfaction in relation to the service.

The CatBoost model showed better training times and less memory needs, while achieving a

better performance than the different architectures of deep neural networks proposed. Only TabNet was able to achieve a similar performance, while maintaining a higher training time. However, in futures uses, where the CatBoost model achieves a plateau and is not benefiting for the increasing data, it could be useful to use TabNet as the model in production. TabNet has the advantage of being a neural network and, for that reason, being more capable of breaking the plateau that classical models often achieve.

Keywords: Customer, Telecommunications, Data Mining, Quality of Service, Artificial Intelligence.

RESUMO

Benchmarking de Deep Learning na Previsão de Problemas Recorrentes de Telecomunicações

Atualmente, as empresas vivem num cenário de forte competitividade. O mercado das telecomunicações não é uma exceção e é possível oferecer uma diferenciação da competição através de melhor qualidade de serviço, suporte diferenciado e até melhores propostas de valor. Com a evolução das tecnologias, as empresas possuem também cada vez mais dados acerca dos seus clientes e sobre o perfil de uso de cada um deles. Com esta informação é possível estabelecer uma melhor relação com o cliente através de um suporte mais eficiente.

A evolução da inteligência artificial e do poder computacional, aliada aos dados existentes, permitem fazer várias comparações entre diferentes algoritmos de *machine learning*. Nesta dissertação, é proposto um modelo de previsão capaz de prever reincidências de contactos com o serviço de apoio ao cliente. O objetivo é, então, prever se um determinado problema reportado pelo cliente se vai tornar recorrente e exigir um novo contacto, para que seja possível proceder à correção antecipada desses problemas, tornando a experiência de utilizador mais agradável e fluida. A fim de alcançar o melhor modelo possível, foram testadas diferentes abordagens clássicas de *machine learning*, juntamente com várias arquiteturas de *deep neural networks*. Nos últimos anos, as *deep neural networks* mostraram resultados interessantes em várias aplicações não tabulares, pelo que é interessante testá-las em aplicações tabulares como a presente neste trabalho. O TabNet, desenvolvido pela Google, é uma *deep neural network* ajustada para ter um melhor desempenho em conjuntos de dados tabulares, e também foi testada, uma vez que mostrou um melhor desempenho do que várias redes neurais e algoritmos baseados em árvores de decisão.

Os dados usados são recolhidos por diversos sistemas internos, sendo que os de maior importância são os dados relativos a chamadas para o apoio ao cliente. O serviço de apoio ao cliente, devido à sua dimensão e complexidade, possui um sistema que monitoriza todas as chamadas e as suas motivações, bem como os intervenientes e outros dados acessórios como tempo dispensado e soluções encontradas. Os dados provenientes de outros sistemas estão relacionados com a faturação, uso e perfil do cliente, com vista a fornecer um contexto para a situação.

O modelo que obteve o melhor resultado foi o CatBoost, baseado em árvores de decisão, com um ROC_AUC de 79%, com uma Recall de 61% e uma Precision de 62%, permitindo identificar cerca de 8,6% das 3,9 milhões de chamadas feitas ao serviço de suporte como reincidências mesmo antes de elas ocorrerem, ou seja, 334 mil casos. Num cenário ideal, todas essas chamadas seriam evitadas, possibilitando uma redução de custos substancial para a empresa, bem como um consequente aumento na satisfação do cliente em relação ao serviço.

O CatBoost foi também o modelo que mostrou melhores tempos de treino e menor exigência de memória, conseguindo ao mesmo tempo um melhor desempenho do que as diferentes arquiteturas de *deep neural networks* propostas. Apenas o TabNet conseguiu um desempenho semelhante, apesar de manter um tempo de treino superior. Contudo, em utilizações futuras, onde o modelo CatBoost atinge um patamar de performance e já não beneficie com o aumento de dados, poderá ser útil utilizar o TabNet como modelo em produção. O TabNet tem a vantagem de ser uma rede neural e, por essa razão, ser mais capaz de quebrar o patamar de performance que os modelos clássicos frequentemente alcançam e não conseguem quebrar.

Palavras-chave: Cliente, Telecomunicações, Mineração de Dados, Qualidade de Serviço, Inteligência Artificial.

TABLE OF CONTENTS

Acknowledgments	iv
Abstract	vii
Resumo	ix
List of Figures	xii
List of Tables	xiv
List of Abbreviations and Acronyms	xv
1 Introduction	1
1.1 Context and Motivation	2
1.2 Objectives	4
1.3 Contributions	5
1.4 Investigation Methodology	5
1.5 Dissertation Structure	6
2 Concepts	7
2.1 Domain Knowledge	8
2.2 Artificial Intelligence	10
2.3 Data Preprocessing	23
2.4 Sampling Techniques	26
2.5 Model Training	30
2.6 Model Evaluation	31
2.7 Hyperparameter Optimization	34
2.8 Related Work	35
3 Recurrences Dataset	38
3.1 Materials	39
3.2 Data Understanding and Processing	42
4 Modeling and Evaluation	59
4.1 Modeling	60
4.2 Evaluation	66
4.3 Business Evaluation	70
5 Conclusions	72
5.1 Conclusions	73
5.2 Future Work	74
References	75

LIST OF FIGURES

- 1.1 Design Research Process schema. 6
- 2.1 Linear Regression illustration. 12
- 2.2 Decision Tree Regression adaptation with different depths. 13
- 2.3 A representation of the division of different points of clusters. 14
- 2.4 An illustration of the data points and sample points to predict. 15
- 2.5 Bias-variance problem illustration. 15
- 2.6 A boosting ensemble example. 17
- 2.7 LightGBM works leaf-wise and vertically. 17
- 2.8 General tree based algorithms work level-wise and horizontally. 17
- 2.9 Other boosting tree algorithms versus CatBoost symmetric trees implementation. 18
- 2.10 Comparison between an artificial and a biological neuron. 19
- 2.11 Fully connected Artificial Neural Network. 20
- 2.12 Comparison between a Simple Neural Network and a Deep Neural Network. 22
- 2.13 Impact on balancing ratio in a linear SVC predictor. 27
- 2.14 Scatter Plot of Imbalanced Binary Classification Problem. 28
- 2.15 Scatter Plot of Imbalanced Binary Classification Problem Transformed by SMOTE. 28
- 2.16 Application of Tomek links on data. 29
- 2.17 Data before Cluster Centroids application. 29
- 2.18 Data after Cluster Centroids application. 29
- 2.19 Activation functions. 30
- 2.20 Confusion Matrix. 32
- 2.21 AUC-ROC Curve representation. 34
- 3.1 Predictive model usage in business. 43
- 3.2 Data sources that make up the dataset. 43
- 3.3 Amount of data entries covered by the Top N most frequent typologies. 44
- 3.4 Recurrence volume of call with the same problem in short time. 45
- 3.5 Recurrence volume of call with the same problem between 30 minutes to 1 hour. 45
- 3.6 Typifications and Categories relation with recurrence. 47
- 3.7 Same typology problem occurrences within 180 days target relation. 48
- 3.8 Teams responsible for ticket openings. 49
- 3.9 Time spent in a customer support call importance. 49
- 3.10 Retention and account age relation with recurrence. 50
- 3.11 Maintenance’s relation with recurrences. 51

3.12	Contracted services and technology relation with recurrences.	52
3.13	Contracted services and technology relation with recurrences.	53
4.1	Precision and Recall Scores as a function of the decision threshold from CatBoost. . .	70

LIST OF TABLES

- 3.1 Hardware in which Docker was running. 39
- 3.2 Examples of target problems count and distributions during the cleaning process. 55

- 4.1 Hyperparameters tested for each estimator. 61
- 4.2 Description of the different MLP structures used. 64
- 4.3 Hyperparameter configurations used for each different structure. 65
- 4.4 Hyperparameters tested for TabNet. 66
- 4.5 Classical models' performances comparison on the validation set. 67
- 4.6 Results obtained with the different configurations for the validation set. 68
- 4.7 Classical models' performances comparison on validation set. 69
- 4.8 Best model's performances comparison on validation set. 70
- 4.9 Results on the test set: default vs. custom threshold. 71

LIST OF ABBREVIATIONS AND ACRONYMS

A

AI Artificial Intelligence

ANN Artificial Neural Network

AUC Area under the ROC Curve

C

CE Cross-entropy

CRISP-DM Cross-industry Standard Process for Data Mining

CRM Customer Relationship Management

D

DCN Deep Convolutional Network

DFE Deep Feed Forward

DNN Deep Neural Network

DL Deep Learning

DT Decision Trees

DSR Design Science Research

F

FTTH Fiber-to-the-Home

H

HD High Definition

HFC Hybrid Fiber-Coaxial

I

IDE Integrated Development Environment

K

KNN K-Nearest Neighbors

L

LSTM Long/Short Term Memory

M

ML Machine Learning

O

OS Operative System

R

RNN Recurrent Neural Network

S

SMOTE Synthetic Minority Over-sampling Technique

W

WCE Weighted Cross-Entropy

1. INTRODUCTION

1.1 CONTEXT AND MOTIVATION

In a world increasingly connected, companies that are able to adapt to new technologies and business models have better chances of succeeding. In this digital era, companies are able to use their systems and data to take better decisions and not just only to provide with the services. The telecommunications market has been experiencing a continuous growth since the beginning of the century. In Portugal, the numbers observed in the last semester of 2018 proved once more this growth, which is very attractive to investors (ANACOM, 2019). The company that supported this dissertation is one of the biggest in the telecommunications and entertainment portuguese scene, offering fixed and mobile solutions like television, broadband, internet and mobile data to all market segments. This really high number of services and customers presents very difficult challenges to the companies, that have to keep all the services operational and all the customer satisfied.

Even though there is a constant investment in improving physical infrastructures and offered technology, today the differentiation sought by telecommunications companies is based on improving the quality of services provided. In order to improve these services, the optimization of operations is key, since it allows redirecting costs to investment in areas that directly affect the user (Pina, Torres, & Bachiller, 2014).

One way to optimize operations is to be able to predict the situations that may occur, taking a proactive attitude in their resolution. Another way is simply to know the levels of customer satisfaction and act accordingly, changing the customer support behavior in certain situations. Typically, these forecasting problems are solved based on statistical models. Nowadays, Artificial Intelligence (AI) has been gaining more and more expression in the industry, presenting significant improvements in the optimization of processes and decisions. This adoption is due to the accelerated growth in computational processing capacity and development in AI algorithms, having already strong application in the world retail (Schaverien, 2019). The telecommunications market is not indifferent to this progress, given its fast and dynamic nature, and is already actively seeking AI-based solutions (Nand Kumar, 2017).

In addition, the fact that companies are increasingly collecting data on services provided, equipment and clients, provides a solid basis for AI modeling. Machine Learning (ML), a subarea of AI, is the one that benefits the most from this increase in data and with which there have been better results, where computers are able to obtain better models the larger the set of data provided.

In fact, telecommunication providers want to avoid the loss of a customer, since acquiring a new one costs 5 to 10 times more than maintaining the current ones. This is a clear motivation to improve the customer support service, since it is the entity with the most interaction with customers themselves. The customer support service can improve by increasing the number of operators, thereby reducing waiting times and having more specializations to solve each type of problem. It

can also be improved in a smarter way, with lower costs than the alternatives that include increasing the number of operators, simply by providing the operators with tools to help the customer faster and more efficiently.

Nowadays, customer service operators have an interface for each one of the calls they attend giving information about the customer and the services he contracted. As an example, all the previous calls and motivations of them are registered and available for operator consultation. By having this and other information gathered in a single tool, operators can understand the context of the problem being reported. However, sometimes customers call for the same reason within a short timeframe and, although the customer can see that a similar problem has already happened, they have to repeat all the procedures in order to fix that same problem. This not only causes the company monetary losses because the operator is doing something that should've been already solved, but also because the time being spent in this customer could be spent with other customer if the problem was correctly fixed in the previous report. In fact, if sometimes a customer support operator has 100% of his time being used in customer calls, some other times it could just be waiting for a new customer to call. Hence, it would be beneficial if this time where operators are free from work could be used to solve problems that are solvable without customer intervention, before they become recurrent problems.

Using the most recent techniques in ML to predict if a customer would need to call again for the same motivation (a recurrence) is a great step to improve customer satisfaction and support centers efficiency. If a possible customer recurrence is predicted, a special care could be provided in the next call, with a direct contact with a specialized operator being made. Even more, the customer could have his problem solved beforehand because one of the free operators could be assigned with the task of checking the status of the service of a customer and perform any operations needed to get everything working perfectly again. Deep Learning (DL), a subfield of ML, has shown great capability to deal with very complex problems like the one being dealt with, by creating a structure that simulates the human brain. DL has been used and showed promising results in fields like computer vision and text recognition, but it is not often used in tabular data like the one available in this work. Recently, Google explored DL applications in tabular data with the novel TabNet, which showed promising results than the most common classical machine learning approaches used for tabular data. This is encouraging, as classical machine learning techniques often have a performance plateau that can be overcome by the DL applications, as they usually scale with the amount of data available. Although statistical approaches are possible, their flexibility and complexity are sometimes not attractive to companies, who aren't capable of trusting in techniques with a performance that could not have an economic impact big enough to get into production. Therefore, DL applications in industry can be of great value, contributing to a better economical performance.

1.2 OBJECTIVES

In this dissertation, the aim is to develop a model for predicting problem recurrence among telecommunication services customer, and validate the corresponding predictions against real cases in an experimental proof of concept. When customers have to interact with customer service their question should be solved quickly and effectively, meaning that they should not have to call again for the same motive. Predicting if they will need to call again, within a time period of 30 days, means that it is possible to predict a recurrence of a problem.

The telecommunications' companies have a lot of data about their customers and the services provided. Ranging from personal customer informations, to service usage profiles, passing by billing and customer support data, a lot of knowledge can be extracted. The most important data is the one coming from the customer support service since the problem that needs to be solved concerns to that. However, all the other data sources provide with data that hints about the recurrence probability when taken in consideration and shall not be disregarded. Therefore, gathering all the data from different sources is the first big challenge of the project.

The second challenge is to compare different machine learning approaches and identify the one that performs the best in the company's business model. In the first place, several classical machine learning models need to be selected accordingly to the data available and the following performance tests made. Secondly, several Deep Neural Networks (DNNs) should be designed and the performance tests should also be made to them. After those steps, the best performance models should be compared taking in consideration the prediction performance, the resources spent on training and the impacts of the amount of data fed into the predictions.

In result of the data preparation and the selection of the best model to predict customers' recurrences, a prediction for each new call ended should be done. If a probable recurrence is predicted for a certain customer, the company should register this for a later customer's service status investigation.

This work aims to develop a model for predicting problem recurrence and its validation in an experimental proof of concept. It is intended that the model developed will be able to predict when a user of a service of a telecommunications company has, after any adverse event, a recurrence of the same type of problem, in a time to be determined. Cases concerning the last three years of events, collected daily by customer services, will be used.

1.3 CONTRIBUTIONS

This study performed an intensive exploration of DNNs application in the present data. The width and depth of the neural networks showed significant importance in the models' performance. This work has been peer-reviewed and accepted at:

- (Castro, Pereira, & Alves, 2020) V. Castro, C. Pereira, and V. Alves. Predicting Recurring Telecommunications Customer Support Problems using Deep Learning. In *Intelligent Data Engineering and Automated Learning - IDEAL 2020*.

1.4 INVESTIGATION METHODOLOGY

Prior to writing this dissertation, a thorough research was required in order to better comprehend possible approaches to solve the problem in hands. To facilitate this process, Design Science Research (DSR) was used. DSR is a research strategy which is considered to be very effective in the computer science field. The DSR methodology (Figure 1.1), is a rigorous scientific research methodology that well defines all the six main steps of research, which consist of:

- Identification of the problem and motivation – what need to be solved and the scientific value of it;
- Definition of the objectives to be attained – what are the solution objectives according to the identified problem and the needs of the research problem;
- Development of the solution – design the features and details of the solution and develop the respective product to solve the problem identified;
- Demonstration – all the experiments and simulation evolved in the process of test and demonstration, which lead to the evaluation of the product;
- Evaluation – compare the defined objectives for the solution to have with the practical observed results. Verify if the solution worked out for the problem identified;
- Communication – present the audience with the solution developed and the need for it. Demonstrate the importance of the problem and consequent solution.

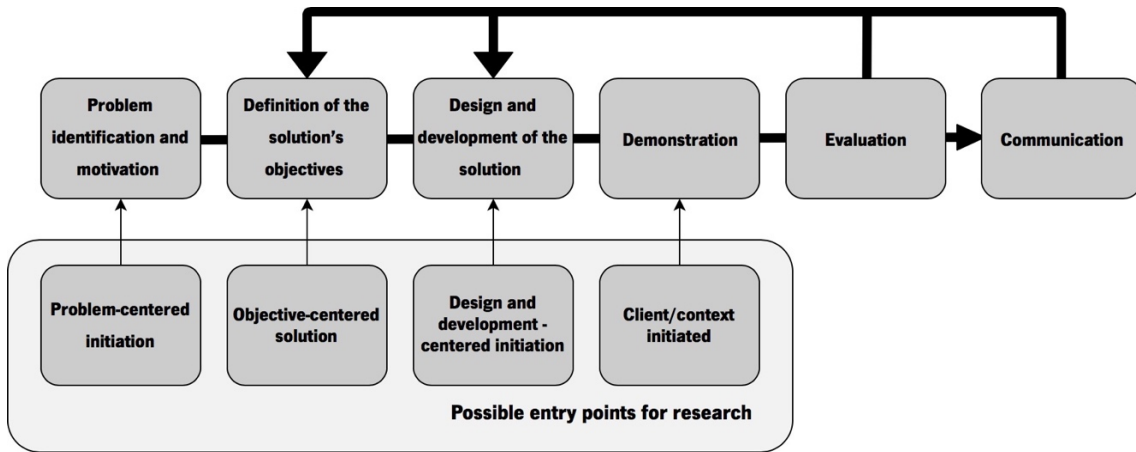


Figure 1.1: Design Research Process schema.

1.5 DISSERTATION STRUCTURE

This dissertation is structured in 5 sections. In section 1, an Introduction is given to bring the reader into contact with the problem and to specify the objectives of this dissertation. The section 2, Concepts, describes in more detail the problem as well as the technologies used throughout this work to overcome it. A brief description of the available related work is also present. The following section, Predicting Recurrences, describes the available data and the methods adopted to produce a quality dataset and the models needed. In section 4, Evaluation, the models are compared and an evaluation of the selected model is done. Finally, this dissertation ends with Conclusions, summarizing everything that has been done up to this point and the next steps.

2. CONCEPTS

2.1 DOMAIN KNOWLEDGE

2.1.1 NOS

NOS was born in 2013 as a result of a merge between ZON and Optimus, two of the largest telecommunications companies in Portugal at the time. ZON was born in 1993, when it was named TV Cabo. At the time there was a growing demand for entertainment and telecommunications services and TV Cabo became the largest television distributor in the country. It was also the first company in Portugal to offer broadband. Since it bord, ZON reached more than three million homes and became the second largest Internet and landline provider of the country. Optimus was created in 1998 and, at the end of the year, had already a market share of 18%. In 2004, Optimus introduced the first plan with free SMS and, in 2008, introduced free calls between users in the same sub-net, being recognized on both occasions with international awards (*ComparaJá - NOS*, n.d.). In 2014, after the merge of ZON and Optimus, the brand NOS was launched. Since that time, NOS was responsible for more innovations like being the pioneer in the country in offering Ultra-HD content. Currently, NOS is the leader operator in subscription television, next generation broadband and movie distribution.

2.1.2 Market

In the telecommunications market, costumers usually subscribe to packages and stay associated to them in a period that can range from one to two years. The packages vary accordingly to the services subscribed, being the most common 1P, 2P, 3P, 4P and 5P. The 1P packages include one of the services that can be television, internet, mobile phone or landline. The 2P packages are usually composed by landline and fixed internet services. The 3P packages comprise three services that are usually television, landline and fixed internet. In some variations, a mobile phone can be associated with the subscription instead of one of the most common services. The 4P and 5P are usually compositions of the 3P package with either one mobile number and one mobile internet or just two mobile numbers. These packages are standard across all the national operators and all the others are variations of the base ones.

2.1.3 Technologies

When trying to differentiate between each other, companies often have to balance cost and quality for the infrastructure they provide. With home installations there are, at the moment, two different common approaches: get the fiber cable into the house or get it near the house and connect the remaining with coaxial cable. Fiber cable is a newer technology that has a higher capacity of

conducting information in a high quality ambient. This higher capacity comes at a cost because the sheer price of the cable is higher and the installation process is more time consuming, where often coaxial cables have to be removed and fiber cables used as a replacement. The costs associated with FTTH (Fibre-to-the-Home), that is, bring fiber optic cables directly to the home of subscriber consumers, led to the use of an intermediate solution, in which fibre optic cables, instead of being installed inside consumers' homes, are placed at points close to them. Therefore, HFC (Hybrid Fibre-Coaxial) is called a hybrid structure that incorporates fiber optics and coaxial cable into the same distribution network. The final path of the connection, closer to the customers home, is connected with coaxial cables. However, due to market pressure and the need to improve service quality, NOS is trying to improve the existent connections from HFC to FTTH, as the direct competitors are using this handicap to negotiate with their customers.

In some less developed areas, DTH (Direct to Home) also exists. Instead of getting the services to customers using a cable approach, DTH is a satellite alternative. It was a common approach a few years ago but with the cost-reduction of fiber, along with DTH being more problematic, it has been less used in new applications.

2.1.4 Customer Support

NOS, like all telecommunications operators, has a customer support service, available on a daily basis, in order to clarify doubts and to provide support for technical difficulties or breakdowns from subscribers. When trying to establish contact with NOS, customers usually call to numbers that are answered by call center operators. Customers usually want and need a fast resolution for their problem and are already annoyed by the situations. Having this in consideration, predicting how the problem is affecting the client, the recurrence of the problem and the level of annoyance it is causing is a valuable information. When trying to solve problems to customers the best-case scenario is to be redirected to the most capable call center operator. This is, however, not possible, for two main reasons, being the first one the lack of availability and the second one the cost associated with having specialized operators. The goal is, therefore, predict the level of annoyance of a customer and, with this information, treat differently the customer, with several levels of problem severity. This enables the company to provide adequate, personalized, fast and comfortable assistance to the most severe cases of annoyance and to reduce the costs associated with problems that could be easily solved by non-specialized operators.

2.2 ARTIFICIAL INTELLIGENCE

AI is a branch of computer science that tries to gather science and engineering in order to build instruments to support and replicate the human intelligence. The definition of AI has evolved in the course of time, due to technology and scientific developments (Kok, Boers, Kusters, Putten, & Poel, 2010). Nevertheless, AI focuses on solving problems that, if accomplished by humans, would be considered a sign of intelligence (Moursund, 2006). The solutions spread across several areas, ranging from autopilots for airplanes until shop suggestions. It all started as a fantasy or a mere possibility of machines working for humans in an intelligent way. Philosophers saw intelligent machines as a way to help them define what a human really is. Descartes, for example, had an idea of a mechanical man, a machine that acted like a man, more as a metaphor than a possibility. As the years passed by, the consequent evolutions in technology and computer science paved the way to a real production and application of AI. Through science fiction, the idea of intelligent nonhuman and robotic intelligent entities spread into people's mind. In the 1980s, chess-playing machines started arising in popularity, just like "The Turk", that fooled people into thinking that the machine was playing by himself. Finally, in 1997, the Deep Blue program defeated the world chess champion, Gary Kasparov, which showed the real potential of AI to the community and led to a huge growth of the field (Buchanan, 2005). In the early twentieth century, several inventions were made by major academic laboratories and companies, proving the benefits of this application in real life.

2.2.1 AI in Telecommunication Providers

Telecommunications' companies have lots of data from a variety of sources. Being capable of using that data to hint about the market and their customers can help on the decision making. Good business decisions can result in an increase in profits and competitiveness, while bad ones can result in permanent loss to the companies.

In recent years, the interest in machine learning has seen a huge growth, and the applications across industries have also increased. Some companies are leaving the statistical models they had by a better performing and flexible solution provided by machine learning, while others are starting to understand the advantages of his usage. The telecommunication industry in no exception, and by identifying customer groups it is possible to create offers more personalized and that fulfill the needs of those customers. By doing so, the customer satisfaction is increased and the consequent probability of churn diminishes.

Churn is one of the biggest problems telecommunication's companies face nowadays, as it is very costly to have a customer doing a new service contract. Customer have lots of motivations to change from a company to another, ranging from service quality and billing problems to service accessibility and better priced offers.

However, the data is often collected in a manner that is not usable in the models. Sometimes it is collected for a certain purpose and so it comes already digested, other times it is collected with no specific purpose in mind and its correctness is undervalued. Moreover, data is not always collected in the same time-frame, causing possible errors when gathering all the data in the same dataset. The problem to be solved is also very unbalanced, which does not help when trying to predict recurrences. For this reason, it is very important to take a deep dive in the data and have it completely understood. Data mining and data visualization greatly help on this subject.

2.2.2 Machine Learning

In the past couple of decades companies are collecting more and more data from their customers or operations. ML refers to the automated detection of meaningful patterns in data. Learning means create knowledge from experience, where experience is given by the data available. Data is, therefore, a key factor for ML applications because they need to pass by a training phase, where they get all the knowledge. As expected, and contrary to the common approach when programming, not everything can be controlled and the output produced cannot be explicitly understood. This happens because ML models can identify patterns where humans cannot. The use of ML makes particular sense when we try to replicate tasks performed by animals or humans, in a routinely but not defined way, just like driving. Tasks beyond human capabilities or that are just very hard to accomplish in a serviceable time window are also a great use for ML applications. Usually, things that require high computational power fit under this category, like astronomical data, weather forecast, genomic analysis and many others. With the increasingly amount of data available and acting like a knowledge base, systems that use ML tend to increase their performance. The adaptability of ML models also contributes to the use of it in real world applications, as environments constantly change and adapting systems can be expensive (Shalev-Shwartz & Ben-David, 2013).

Supervised Learning

Learning can occur in a supervised way, where the training data has information about the correct answer the model should output. With this approach it is expected that the learner should compare the new situations with the previously identified patterns, establishing positive relations that outcome a right solution. Commonly, this is used in scenarios where the goal is to predict some missing information (Shalev-Shwartz & Ben-David, 2013). The answers provided are called labels and a problem could be identified as a classification or regression problem according to them. In a classification problem, labels are finite and represent something discrete like YES/NO, colors (R/G/B) or even a mood (Happy/Sad//Frustrated). However, in regression problems, the number of possibilities is virtually infinite and the idea is to approximate a function that represents all the data points.

Unsupervised Learning

In the unsupervised learning approach, the goal is usually to create a summary or compressed version of the data available (Shalev-Shwartz & Ben-David, 2013). This learning, contrarily to the supervised, doesn't use labels as a way to get information about the problem. Instead, the learning idea is to identify similarities between the data entries. By finding this similarities and patters it is possible to create several clusters of data that can be considered similar.

2.2.3 Classical Supervised Classifiers

Logistic Regression

Logistic regression is used for classification tasks. It derives from Linear Regression by composition of the sigmoid function over the class of linear function. As it is a statistical model, the sigmoid function is applied over the probability of the class.

A linear regression is an algorithm capable of identifying a relation between one or more variables, by fitting a linear equation to the given data points. This linear equation forms a regression line, like the one seen in figure 2.1.

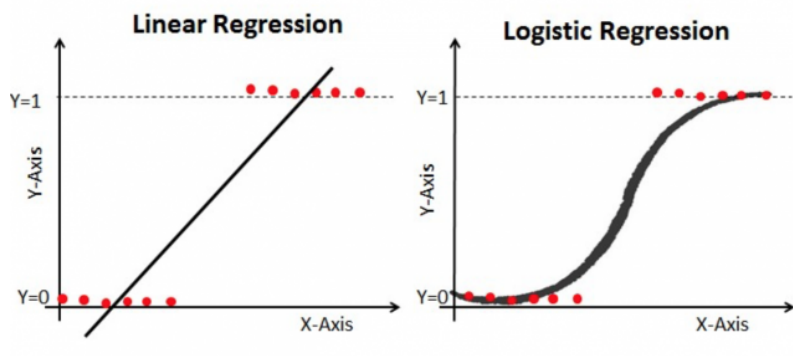


Figure 2.1: Linear Regression illustration. Extracted from (Ralabs, 2019).

In a linear regression there are several variables X that result in a outcome Y . If X is only a variable, the model is a simple linear regression, whereas in the case of several variables X , the model is a multiple linear regression.

Logistic regression works in a similar way but, instead of having just one outcome Y , it has 2 or more. As an example, there could be 3 different Y s, each one of them representing a possible outcome. The exit value for each one of the Y s would be corresponding to the probability of that outcome. All the Y values would sum up as 1, or 100% probability of one of the outcomes to happen.

Decision Trees

Decision Trees (DT) classifier works by breaking down a dataset into smaller subsets based on different criteria. When trying to predict an instance X it travels from the root node of the tree up to the leaf, after clearing all the conditions created when training, taking in consideration the best and most significant features for the model at each node. The number of nodes is given by the depth we want to achieve, according to the problem, resources and data available (Fig. 2.2). They are good predictors for both binary and multi class problems.

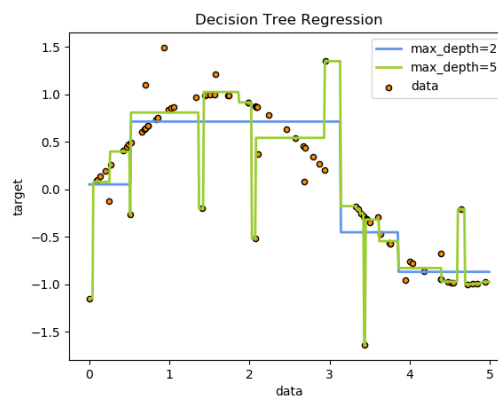


Figure 2.2: Decision Tree Regression adaptation with different depths. Extracted from (Pedregosa et al., n.d.).

This is a non-parametric supervised method. When in train, a series of trees are built and in each tree exists a node holding a classification rule. In each side of the node there could be another two tree that have other classification rules themselves. When a data entry is evaluated it follows a path in the tree built in the training phase. At each node, the classification rule is compared with the information that the data entry holds. In DT, the data is divided in heterogeneous groups using techniques to gain information like Gini, Chi-square and Entropy.

Support Vector Machines

In Support Vector Machines the work is done by separating as much as possible the different clusters of data identified (Fig. 2.3). The main problem for the algorithm is to predict from which cluster a data point belongs when it is equally distant from two or more classes. The separation and increase of distances between the points of the classes is, therefore, the main challenge and is countered by different kernels and parameters that could be applied.

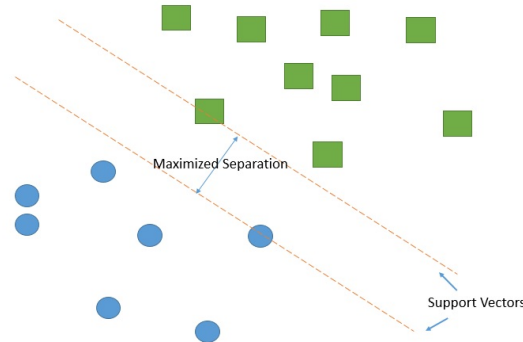


Figure 2.3: A representation of the division of different points of clusters. Extracted from (Nelson, 2020).

Naïve Bayes

The Naïve Bayes classifier is a clear demonstration of how generalization can simplify the learning process. The generalization is done by assuming that each one of the features is independent of each other, decreasing the number of examples needed to learn about all the features. It is especially efficient in situations where the amount of data available is not big enough for other algorithms. Another benefit is the high scalability of this classifier due to the linear time taken to evaluate, instead of the more common iterative approximation that other classifiers perform.

The classifier is based in the Bayes theorem, that describes the probability of an event A to happen, knowing that a event B already happened, equal to the probability of a event B happen, knowing that A happened, multiplied by the probability of the event A happening, all that divided by the probability of the event B happening (Eq. 2.1).

$$P(A|B) = \frac{P(A|B) * P(A)}{P(B)} \quad (2.1)$$

The name "Naïve" comes from the supposition that the variables are all independent. By considering them like that, the probability of B only needs to be extended to all the variables of a data entry.

K-Nearest Neighbors

The K-Nearest Neighbors (KNN) classifier operates by checking the distance from the test sample to the known training examples. In the training phase, it groups data into classes according to the patterns identified, balancing the importance of each feature accordingly. In the test phase, it aligns the example provided and measures the data point created to the center of the other data points, classifying the new entry as the one closest to the data points previously known (Fig. 2.4).

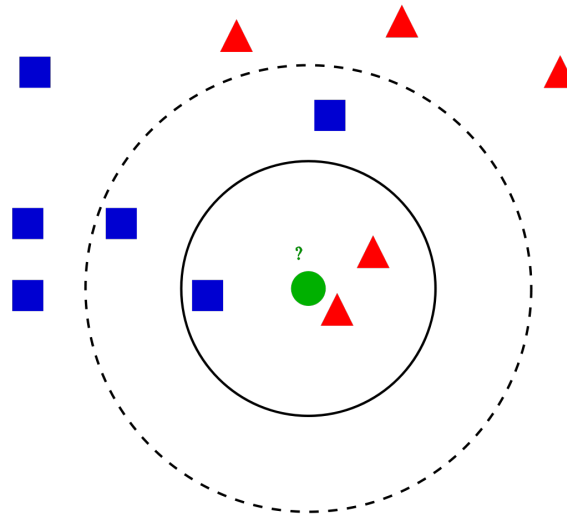


Figure 2.4: An illustration of the data points and sample points to predict. Extracted from (Srivastava, 2018).

2.2.4 Ensembles

Ensembles is a machine learning technique where several base models are combined to produce the optimal model. A common problem in machine learning is having too much noise, bias and variance (Fig. 2.5).

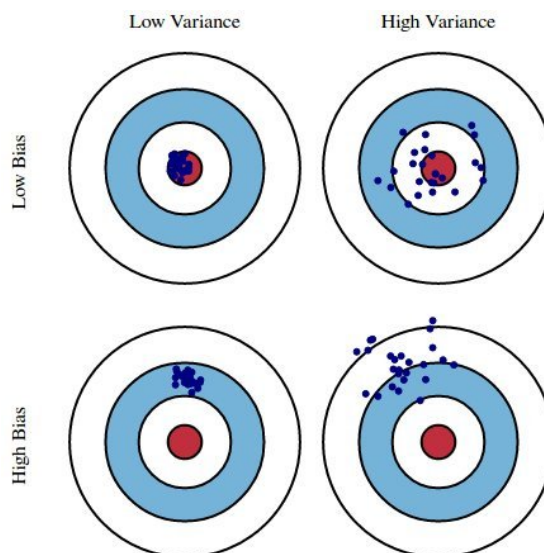


Figure 2.5: Bias-variance problem illustration. Extracted from (Mayo, n.d.).

In general, ensemble techniques produce more accurate predictions, are more stable and robust because their individual models are simple and can capture both linear and non-linear relationships in the data. However, there is a reduction in model interpret-ability because several models contribute to the prediction result. Computation and design time can also be costlier because several models can take too long produce an output. Ensembles could be of Bagging or Boosting type.

Bagging

In Bagging, each one of the models that make up the solution is exposed to a different subset of data. The output of the individual models is, therefore, independent of the others even at the data level. The data being divided and independent makes overfitting a less notorious problem because the variance error is, overall, spread across the models. This helps to reduce the variance, thereby overcoming problems related with overfitting.

Random Forests are an example of the application of the bagging technique. It's an algorithm based on DT that splits the data entries features in subsets and builds several trees, having each one of them the corresponding subset of features. When trying to make a prediction, all the DT have their outcome. The resulting outcomes are weighted and the final decision comes as a result of all the previous ones.

In this model it is important to consider the amount of trees and how big a leaf should be considered before becoming another tree. Less trees and bigger leaves get faster results. However, if those values are too large or too small they can reduce the performance of the model.

Boosting

Boosting, contrarily to Bagging, makes the models dependent on others. It is an iterative technique that increases the weight of an observation based on the last observation done, in order to classify it correctly afterwards (Fig. 2.6). The first algorithm is trained on the complete data and the subsequent algorithms are trained by fitting the residuals of the first algorithm and giving enough weight to those poorly predicted observations. The weak learners, the individual algorithms, are good specifically in predicting what they were trained for and that is why the model they can boost the ensemble performance.

The most well-known boosting technique, Gradient Boosting, relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction 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. The base idea is to set target outcomes for the next model, in order to minimize the error. If a change in the prediction causes a big drop in the error, then it has great value in the next model.

LightGBM is a gradient boosting framework that uses tree based learning algorithms. By default it uses decision trees but also supports random forests, Dropouts meet Multiple Additive Regression Trees (DART), and Gradient Based One-Side Sampling (Goss) (*Welcome to LightGBM's documentation! – LightGBM 3.0.0.99 documentation, n.d.*).

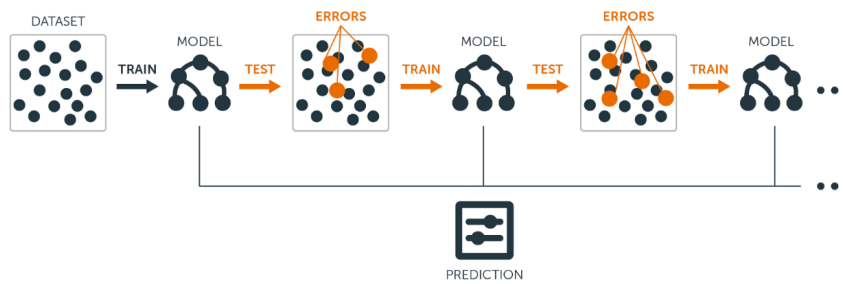


Figure 2.6: A boosting ensemble example, training the subsequent models with the errors from previous iterations. Extracted from (Ramzai, 2019).

The main difference over other algorithms is that it grows the trees vertically instead of horizontally and does it leaf-wise (Fig. 2.7) instead of level-wise (Fig. 2.8). It grows the leaf with higher delta loss, decreasing the max the loss at each level. This solution allows for a higher speed in the algorithm, making it handle larger data sizes and lowering the memory needed to run. Those characteristics are of high importance in today's age, where data is becoming more abundant and complex.

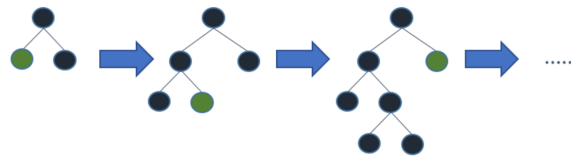


Figure 2.7: LightGBM works leaf-wise and vertically. Extracted from (Mandot, 2017).



Figure 2.8: General tree based algorithms work level-wise and horizontally. Extracted from (Mandot, 2017).

CatBoost is also an algorithm for gradient boosting but only on decision trees. CatBoost implements symmetric trees, which is also a better solution to general decision trees tree growth which can be unbalanced (Fig. 2.9). Having the trees symmetric results in a more constant time to prediction.

Another major advantage of this algorithm is that it can reproduce a time sequence in the training examples. This is important in situations where a row of the training data is not independent of the previous events (e.g. the same customer's previous calls).

XGBoost is an algorithm based on DT where the boosting is applied to the gradient. It is similar but not as fast as the LightGBM, offering, however, more consistent results.

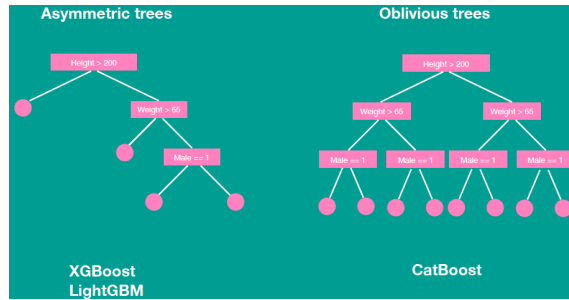


Figure 2.9: Other boosting tree algorithms versus CatBoost symmetric trees implementation. Extracted from (Chepenko, 2019).

Stacking

In stacking, models are not simply trained with a subset of the features or being improved over-time. Instead, from the various models available the best one is chosen for each one of the possible labels. As an example, if a classification task has labels A, B and C, there will be 3 classifiers chosen with each one of them being the better predicting label A, B or C. There are two components that make this possible:

- **Base model:** individually trained on the data and the ones that give the probability values for each label;
- **Meta model:** responsible for selecting the best base model for each one of the classes.

Voting

In voting, every one of the models outputs is prediction and the final prediction is a balance of the votes. This balance could be done in two different ways:

- **Hard-voting:** where each one of the votes has the value 1 or 0 and the result is the mean of the votes;
- **Soft-voting:** where the votes consist in probabilities (i.e. confidence in prediction) and the prediction will be according to the mean of the predictions.

2.2.5 Artificial Neural Networks

AI can be defined as the capacity of a computer have to simulate the human intelligent processes. AI is, therefore, a branch of computer science which tries to create systems capable of learning new concepts and tasks. These systems, ultimately, solve the problems they may face accordingly to

the behavior acquired previously in the most intelligent way (Hunt & Hunt, 1986). One of the several approaches to AI is ML, which consists of a computer being trained with large amounts of data using specific algorithms, in order to allow the computer to gain the ability to adapt to new situations having in consideration previous patterns. Artificial Neural Networks (ANNs) are one of the several algorithms used in ML. An ANN is a system capable of processing data in a large and interconnected structure, similarly to the human brain cortex. Having this human brain approach to problems, by generating connections between the structures, computers are provided with a toolset that make them capable of doing similar things to humans and that normal computers do in a poor way (Uhrig, 1995). Having this structures (Fig. 2.10) that provide similarity to human brain, ANNs show a big capability to modify their own internal structures in order to better perform in relation to an objective function. They are able to solve complex nonlinear problems because they can act over knowledge instead of computing all the possible options or simply follow rules.

In figure 2.11 it's possible to observe the propagation and back-propagation steps. This is the most common method to train the neural networks, with the propagation phase being responsible for passing the inputs of the new data entry to the network, which generates an output. That output is then compared to the true output and the error is calculated. The back-propagation phase uses the calculated error to adjust the neurons' weights from the last layer to the first, adjusting the future outcomes.

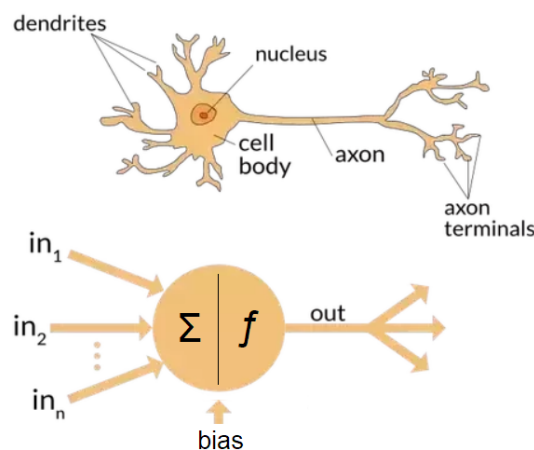


Figure 2.10: Comparison between an artificial and a biological neuron. Extracted from (Vazquez, 2018).

The internal structures are composed by nodes, which have one or more connections to other nodes, and are activated (like in the human brain) accordingly to a certain activation function. Therefore, it's worth to better understand this basic structure of an ANN.

Nodes Nodes are the basic structures of ANNs, acting like the human brain's neurons. They receive the data and produce an output, accordingly to the weight associated to the input, the bias and the activation function they have.

Each one of the nodes receives communications from the ambient, if it is part of the primary layer, or other nodes, if it is part of the subsequent layers. Each one of these nodes has his unique behavior accordingly to the function given at it and the data fed. This function, called activation function, acts like a switch between the input and output, that modulates the signal accordingly to the problem, similarly to the behavior of neurons on a human brain. Adding this behavior with the ability to give different weights to each one of the input connections, it is possible to build a complex system that replicates the human brain behind a complex mathematical equation composed by nodes functions and connections weights (Grossi & Buscema, 2007) (Fig. 2.11).

It is also possible to introduce a bias element in the node, just like our brains do. Bias allows neurons to favor a certain output, and is useful in real-world scenarios where some outputs are more important or less frequent and need an adjust.

Nodes are combined in order to perform a layer (Fig. 2.11). A neuron by itself wouldn't be able to communicate and generate the best outputs for every different situation, as the result of our brain's success is a composition of millions of neurons working together. A higher number of layers is used in complex problems, demanding for higher computational resources.

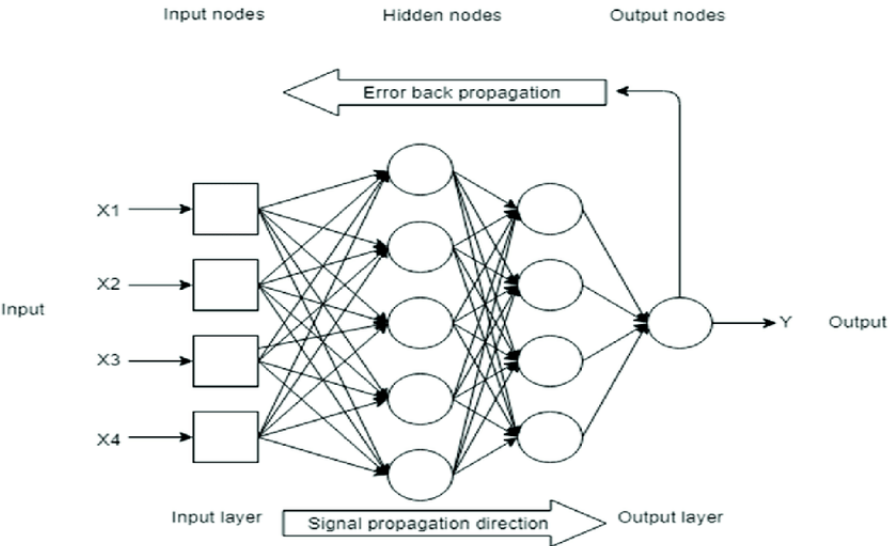


Figure 2.11: Fully connected Artificial Neural Network (Azlah et al., 2019).

There are also several parameters that can be adjusted within the ANNs, that are important and make for a better learning process. Those are:

- Number of Hidden Layers - the sheer number of layers a ANN uses to comprehend the data and create relations between the inputs received. For a faster and better generalizing network, the fewer layers the better. Increasing the number the layers, however, increases the classifying power of the neural network for more complex problems;

- Learning Rate - determines the amount of correction term that is applied to adjust the neuron weights. When a ANN is being trained, the neurons weights are constantly adjusted accordingly to the result of the previous prediction. The greater the learning rate, the faster the training time. However, this could lead to less optimal results, whereas lower learning rates take more time but can be more accurate. However, small learning rates also have the disadvantage of possibly being stuck on a local minima;
- Momentum - determines how much of the previous correction is remembered and carried on in the next correction. With a larger momentum, the current correction has more emphasis. Momentum is important because it allows for a smoothing between the current and the previous corrections, so that a single iteration doesn't ruin the learning process in an undesirable correction;
- Activation Function - the function through which the weighed sum of the node is passed, in order to have a significant output, namely as a vector of probability or a 0–1 output;
- Minibatch size - the number of samples a model trains with at each step. When feeding the ANNs with big amounts of data, it could take long for them to train if they are fed with one sample at a time. The typical size of the minibatch is 32 or higher, but with the consideration that a value could be too big, making the model over generalized;
- Epochs - the number of times the ANN is fed with the entire dataset. A low epoch number can lead to underfitting, while a big epoch number can lead to overfitting. This happens because the bigger the number of times the model is trained on training data, the more it gets adjusted to it;
- Dropout - removing some of the nodes, making the model less heavy and lowering the information propagated with redundancy. This also helps the model to generalize better.

Deep Neural Networks

Computers can perform logical operations faster than humans. DNNs are an evolution of ANNs, where several more layers are added, in order to process the information received in a more human way. ANNs and the shallow common ML models (like Support Vector Machines, Random Forest) often fail to recreate the abstract capacity of human brain, like the associations and deconstruction of an image or sound. Through life, humans are exposed to an immense amount of information and always learn something from it. Experiences make people learn what to expect from certain input, just like a DL model does. Children are in the early stage of life and, therefore, are exposed to a more intense process of learning and adaptation. They learn what to say in certain situations, how to demonstrate their feelings, how to recognize animals and objects, always with the help of

their parents. Parents act like the feedback system of DL models, as they help to get the correct output value and provide children with the best data as possible.

The shallow ANN have only 3 layers (input, hidden and output), whereas DNN have multiple hidden layers (Fig. 2.12). Through the use of multiple hidden layers it is possible to get a better approach to the way human brains work. As an example, humans can smell food (receive the input), have the desire to it something (hidden layer for thought), then think about the diet they are committed to (hidden layer for memory), and decide to ignore the diet just for once (hidden layer for decision making), with the outcome of really eating something. In an ANN all this reasoning would not be possible since there would be not enough hidden layers. The disadvantages inherent to this approach are the increased difficulty (i.e. time) to train and the possibility that the neural network becomes too intelligent for the problem itself. When the DNN are deepest then the problem difficulty it's frequent to occur overfitting.

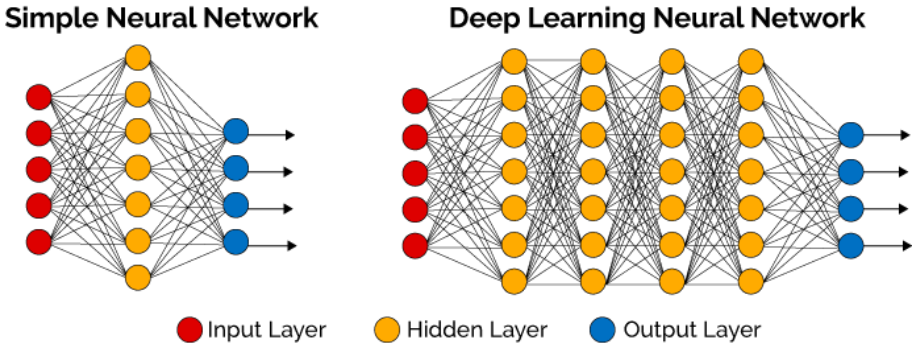


Figure 2.12: Comparison between a Simple Neural Network and a Deep Neural Network. Extracted from (Vazquez, 2017).

Common DNN architectures There are several architectures of DNNs, which are often used in different applications. The most common applications of DNNs are related with image, text, and sound.

The Deep Feed Forward (DFF) is the most common structure, where the first layer receives the inputs, passes them to the middle (hidden) layers, and produces the output. It only differs from the typical feed forward network because it adds multiple hidden layers. They nodes are often fully connected with the nodes of the previous and next layer.

The Recurrent Neural Network (RNN) introduces a different type of cell, the recurrent cell. This cell is mainly used in problems where the context is important, because they pass their output to themselves with a delay, i.e. they see their output from previous iterations. Applications for these types of networks are commonly found in problems involving texts.

Long/Short Term Memory (LSTM) networks are also commonly used in text classification. They introduce a new type of cell that resembles memory. These cells have a series of logical gates that

allow for previous information (memory) to pass on in a future time, while also erasing previous memories from time to time.

Deep Convolutional Networks (DCNs) are the most well-known networks to process images nowadays. They feature convolution cells that simplify the data, keeping only the features that are relevant during this compression. In image recognition, this could be a layer of gradients, another of lines, a third of shapes, and so on. By the end of these layers there is a DFF network for final processing.

The most relevant architecture for this dissertation is DFF, because no time or memory factor is needed, and no image processing is to be made.

2.3 DATA PREPROCESSING

Preparing the data is very important to the success of the machine learning techniques. Data is gathered using several devices and passing through several phases before getting into the data warehouse, which increases the probability of errors, inconsistencies or noise. ML models frequently have their performance reduced by this problems, so reducing their impact is of great importance.

2.3.1 Data Visualization

Nowadays, data is more complex than ever. Understanding the meaning of each data field is very important to understand the problem being faced and contributes to an easier solution. Data visualization tools makes this easier than using tabular data. It's easier for humans to understand relations through visual application, like a graphic or a map, and even to see where patterns, tendencies or outliers are forming. Some of the most common graphics are:

- CountPlot - a graphic that uses bars to represent the corresponding count for each column value;
- BarPlot - a graphic similar to the CountPlot but using the count axis to represent another variable. Allows to put two variables against each other;
- DistPlot - a graphic that represents counts for variables contained in certain intervals;
- BoxPlot - a graphic that represents the lower and superior limit of the data, as well as the quartiles and median. It allows to better visualize outliers.

2.3.2 Data Cleaning

After understanding the state of the data, it is important to clean it and improve its quality. The problems targeted in this step are:

- Inconsistencies;
- Duplication;
- Empty data fields;
- Data fields with wrong information.

Inconsistencies are generated when joining data from different sources, as some sources contain informations related to their domain that, when used in order to make predictions on other domain, lose their meaning or compatibility.

Duplication can also occur due to merges and joins or because the information systems need to follow protocols whose transaction are recorded.

The lack of data present in certain records is also a big problem. In situations where data is missing it is very important to decide between removing the entire data entry or to fill it. Usually, when lots of missing fields are present the data entry is removed. When this happens, due to the characteristics of the problem, all the data entries related to the customer should also be deleted. If that operation was not performed, the models could see only parts of a problem that led to the recurrence, reducing the capability of identifying those situations. When filling the data entries with other calculated values it's probable that the data becomes deviated from the reality, which also causes troubles to the models.

Data fields with wrong information can occur when some field is filled with data that is calculated using considerations that are not considered true for the problem being solved. When this happens, the problem should be addressed similarly to an empty data field situation.

2.3.3 Data Integration

Usually, when facing a certain problem, data scientists have one data source in mind that could be leveraged to solve that problem. However, more data from different sources could be added to this base dataset, contributing for a better performing model. Joining data from several sources is, however, a difficult operation, from sources with different timestamps to different IDs. This step needs to be done by someone with a high level of business knowledge.

2.3.4 Data Transformation

Transforming the data is important because it introduces business knowledge into the dataset or makes the data have less noise. There are several procedures that count as data transformation steps (Han, Kamber, & Pei, 2012).

Smoothing

Used to reduce noise from data and includes the following techniques:

- Binning - Where data is grouped by ranges, thereby reducing the number of different values. The groups could have the same length or the same number of data points;
- Clustering - An automatic process where an algorithm gathers data points by clusters, comparing their similarities.

Feature Construction

New attributes are constructed and added using the given set of attributes. This helps to make clear for the model characteristics that seem to be important to the data scientist.

Aggregation

Where the available data is aggregated or summed. As an example, joining all the daily upload data consumption in a monthly feature.

Normalization and Standardization

Normalization happens where the attribute data is scaled so it fits in a certain range, usually of -1 to 1 or 0 to 1. This helps the models to consider every feature with the same importance. Standardization happens when the data is transformed in order to have an average of 0 and a standard deviation of 1.

When the data distribution is not gaussian or the data deviation is low, normalization is a good choice since it doesn't change the data but only the scale. Normalization, however, gives outliers a bigger importance with the scaling. Standardization reduces the outliers' values but can generate a significant distribution change in the data. Currently, some different version of normalization have been developed that try to tackle his lack of outliers' remotion by making use of the quartiles minimum and maximum values.

Discretization

Where the raw numeric values are grouped using labels to replace them. As an example, classifying every person with an age between 0 and 18 as youth.

Concept hierarchy generation for nominal data

The same as discretization but with things like street names grouped in cities.

2.3.5 Feature Selection

Although adding information to the model is important, sometimes reducing the number of features can contribute to the model's better performance. Columns with a lower importance can affect the model because:

- They introduce redundancy, increasing the overfit potential which is undesirable;
- Increase the training time since the model needs to perform calculations and take in consideration that variable;
- Reduces the model's performance because it introduces noise.

There are three possible methods to reduce the features' count. The filter methods use statistical metrics that observe the intrinsic properties of the features. Wrapper methods, however, use the performance retrieved from classifiers in order to calculate feature's importance. In embedded methods, the variable selection algorithm is integrated in the algorithms' learning phase.

2.4 SAMPLING TECHNIQUES

Sampling Techniques are often used in problems where there is an unbalancing in the data. Increasing or decreasing the frequency of a class in the data can improve the prediction capability of a model (Lemaitre, Nogueira, Oliveira, & Aridas, 2017a). Figure 2.13 shows the improvements of increasingly balancing of samples in each classes for a given problem.

The dataset used to produce this work was unbalanced, with a ratio of 1 positive for each 9 negative values. The most important class to predict was the positive one, that represented cases of possible recurrence. This motivated several tests with and without sampling techniques, in order to understand the sensibility of the models to them.

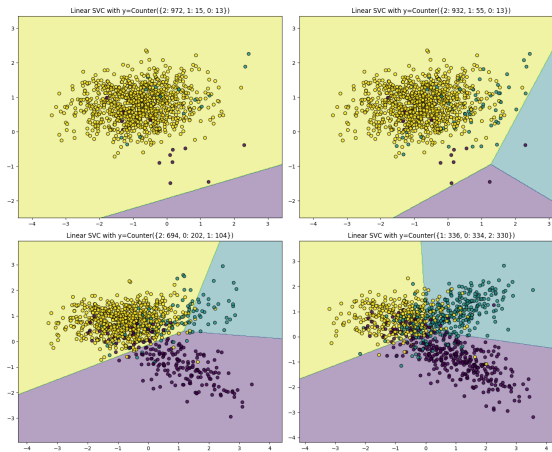


Figure 2.13: Impact on balancing ratio in a linear SVC predictor. Extracted from (Lemaitre et al., 2017b).

2.4.1 Oversampling

Random Oversampling

The Random Oversampling is the most common oversampling method. In this method, random entries are chosen to be duplicated in order to achieve the desired proportion of data. This technique is simple but often causes overfit and doesn't add any information to the model.

SMOTE

Synthetic Minority Over-sampling Technique (SMOTE) is an alternative to the Random Oversampling technique that, instead of duplicating existing entries, synthesizes new ones from the existing. This not only helps to prevent overfitting, since the data is not repeated, but also can be seen like a type of data augmentation (Brownlee, 2020a).

The way SMOTE works is by selecting entries that are in a close feature space and calculating a function that approximates them. By doing so, new points of the function can be calculated, giving new entries that will be used in the oversampling. To do so, a first random entry is chosen and then a number of k nearest neighbors of that class are found (usually $k=5$). From the found neighbors, one of them is chosen randomly and the function is calculated to approach the two selected entries in the feature space. In Figure 2.14 is represented an imbalanced situation that is after fixed using SMOTE, as seen in Figure 2.15.

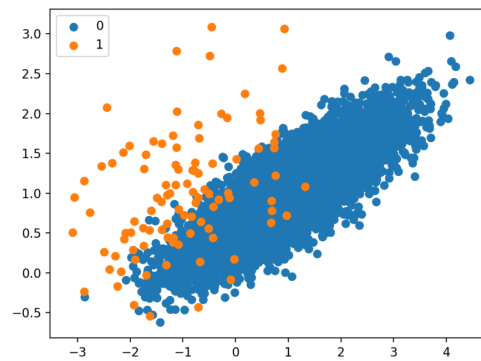


Figure 2.14: Scatter Plot of Imbalanced Binary Classification Problem. Extracted from (Brownlee, 2020b).

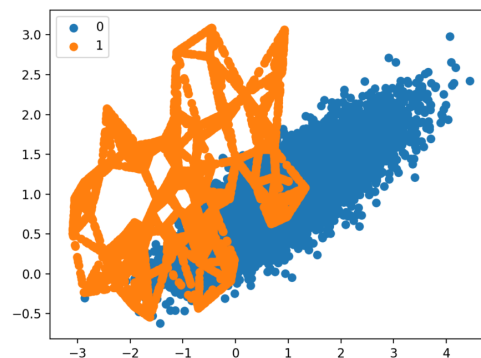


Figure 2.15: Scatter Plot of Imbalanced Binary Classification Problem Transformed by SMOTE. Extracted from (Brownlee, 2020b).

2.4.2 Undersampling

Random Oversampling

The Random Undersampling is the most common undersampling method. In this method, random entries are chosen to be deleted in order to achieve the desired proportion of data. This technique is simple but often causes lost of information, mainly in situations where data entries are not fully independent.

Tomek links

This technique is an alternative to random undersampling that tries to increase the space between the different classes. This increased space allows the models to better separate the classes, not having to considerate the sometimes noisy data in-between. The Figure 2.16 shows how this method

changes de data distribution.

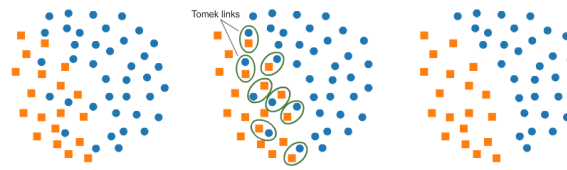


Figure 2.16: Application of Tomek links on data. Extracted from (Alencar, 2017).

Cluster Centroids

In this technique, centroids are generated using clustering methods on the data. The data is grouped by similarity so it helps on information preservation. Figures 2.17 and 2.18 show how data distribution is affected after the application of the undersampling technique.

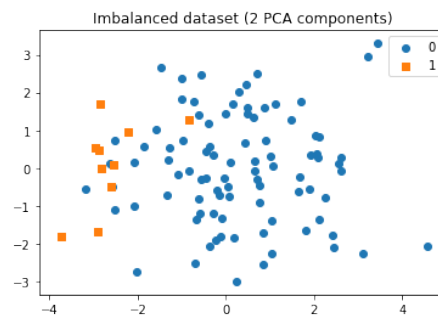


Figure 2.17: Data before Cluster Centroids application. Extracted from (Alencar, 2017).

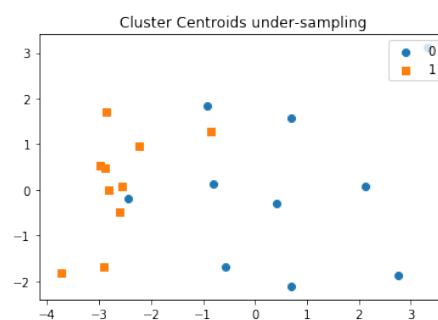


Figure 2.18: Data after Cluster Centroids application. Extracted from (Alencar, 2017).

2.5 MODEL TRAINING

2.5.1 Activation Functions

Nodes, the artificial structures that aim to recreate human neurons, act under a function. This function allows nodes to have a positive or negative impulse upon the next node. The function outputs a value according to the received inputs. Activation functions can be linear or nonlinear (Fig. 2.19). With linear functions the results are easier to compute and provide a faster model, while reducing his capability and leading to limitations. On the other end, nonlinear functions are more powerful and complex and can be one of the following: Softmax, Sigmoid and ReLU. Softmax is often used in multi-classification problems, normalizing the previous layers bringing the total of all to one. This leads to a result that specifies the probability for each one of the classes. In the end, only one label should have a value of 1, while the rest maintain a value of 0, indicating strong certainty in the prediction. Sigmoid is usually used in binary classification problems and aims to give values closest to 0 and 1. ReLU is becoming a very used and important function over the past years. It works well with large and consistent gradients and, for that reason, is being heavily used in computer vision applications. It blocks the negative values and gives a linear importance to the positive ones. This blocking action over negative values can, sometimes, be negative. Therefore, Leaky ReLU was invented and does not block the negative values, only attenuating them.

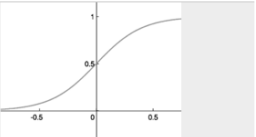
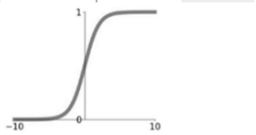
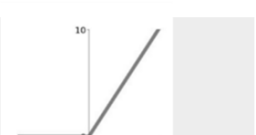
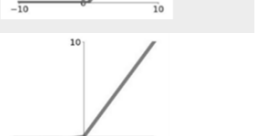
Activation Function	Equation	Graphical representation
Softmax	$f(x) = \frac{e^{x_n}}{\sum_{c=1}^n e^{x_c}}$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
ReLU	$f(x) = \max(0, x)$	
Leaky ReLU	$f(x) = \max(ax, x)$	

Figure 2.19: Activation functions.

2.5.2 Loss Functions and Optimizers

Other than activation functions, loss functions and optimizers are also very important to build a efficient brain-like system. The loss function acts like the dad or mom who corrects any incorrect behaviour of the sun they educate. In neural networks, this loss applies a penalty for the incorrect predictions the model outputs, giving the feedback needed. The system receives the feedback of the used loss function at any point and tries to adapt in order to reduce the loss. The adaptation occurs in the nodes and in the connection's functions and weights. Optimizers are responsible for this operation and can perform differentiation or statistical work in order to succeed. In the present work, optimizer is choice is not as important as loss function choice and, for that reason will not be described in such an extensive way.

Cross-Entropy

Cross-entropy (CE) is one of the most common loss functions. It is a positive function that tends to zero as the model becomes more capable of outputting the right values. For a multi-class problem, it is possible to introduce a factor in representation of the class frequency, in order to mitigate the class imbalance problems that often occur. This variation of CE is called Weighted Cross-Entropy (WCE).

2.6 MODEL EVALUATION

2.6.1 Confusion Matrix

A confusion matrix (Fig. 2.20) is a matrix that shows the predicted against the true values. Each row of the matrix represents the instances in a predicted class, for each one of the classes. The columns represent the actual class for each one of the classes.

2.6.2 Accuracy

Accuracy (Eq. 2.2) is a metric for evaluating classification models, as a fraction of predictions the model got right, including positives and negatives (Fielding & Fielding, 2010).

$$Accuracy = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad (2.2)$$

There is, however, a relevant point to be taken in consideration when using this metric: it only shows the true effectiveness of the model when there classes are balanced. This could be easily

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 2.20: Confusion Matrix.

explained by a situation when credit card fraud detection is imperative. In that case, the majority of the occurrences are negative, i.e. there is not fraud, while a few of them will be positive. The model could easily answer that every transaction was not fraudulent, therefore guessing on all the true negative values. As the majority of the values would be negative, the model would have a great performance by the metric by would not predict any fraud. In this situation, the cost of not predicting a true positive is very high since a customer's credit card would be used and the money available could be spent.

2.6.3 Precision

Precision (Eq. 2.3) is a metric to measure the proportion of positive identification actually correct. It takes only the positives in consideration (Google Developers, 2020a).

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad (2.3)$$

2.6.4 Recall

Recall (Eq. 2.4) is a similar metric with Precision. Instead of measuring the relation of how much of the positive identifications were actually correct, it tries to measure how many of the actual positives (and not the predicted) was correctly identified (Google Developers, 2020a).

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (2.4)$$

2.6.5 F1 Score

The F1 score (Eq. 2.5) relates precision and recall in a weighted average, reaching his best value at 1 and worst at 0. In multi-class problems, the F1 score is related to each one of the classes.

$$F1\ score = \frac{2 * precision * recall}{precision + recall} \quad (2.5)$$

A more general F score (Eq. 2.6), that uses a positive real factor β , where β is chosen such that recall is considered β times as important as precision, is:

$$F_{\beta} = (1 + \beta^2) * \frac{precision * recall}{(\beta^2 * precision) + recall} \quad (2.6)$$

The F1 score metric ($\beta = 1$), who gives the same importance to recall and precision, could be used in a unbalanced dataset, instead of accuracy. It is also possible to divide it in macro-F1 and micro-F1. The first one evaluates the performance considering that all the classes have the same weight. The last one performs the evaluation considering the classes' distribution, therefore being more used in unbalanced datasets.

2.6.6 ROC Curve

A ROC Curve (Fig. 2.21) is a graph showing the performance of a classification model at all classification thresholds. The curve plots against two axis: True Positive Rate and False Positive Rate. There is also the AUC, or Area under the ROC Curve, which represents the probability that the model ranks a random positive example higher than a random negative example. In conclusion, AUC-ROC is a performance measure that represents the degree of separability of classes, that is, how capable is the model of distinguish them (Google Developers, 2020b). The threshold is the probability value from which a sample is considered positive.

2.6.7 Cross-Validation

Cross-validation is a method to test models' performances on unseen data. A big mistake when measuring models' performances is testing in the same data in which the model was trained. Simply removing a part of the available data is not enough to have confidence in the performance because data could be badly separated, not representing real-life situations (Pedregosa et al., 2015). There are two big types of cross-validation:

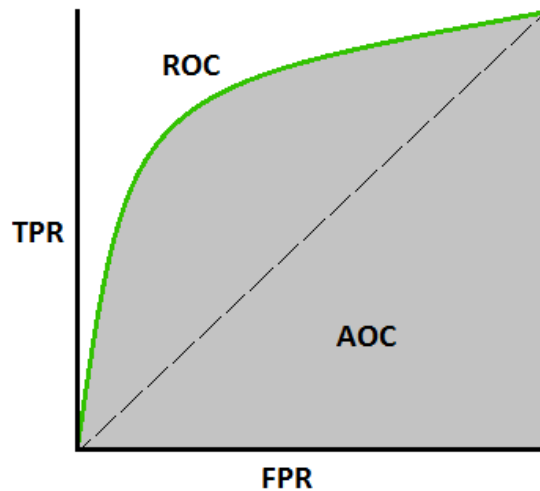


Figure 2.21: AUC-ROC Curve representation (Narkhede, 2018)

- K-Fold - the data is divided in k subsets and the validation method is repeated k iterations. In each iteration, one of the subsets is used as a test and the rest is used as training. This greatly reduce bias because the majority of the data is used as training;
- Stratified K-Fold - K-Fold does not consider the target class when making the separation through the subsets. Hence, there is a slight variation from the K-Fold implementation, where each one of the subsets has the same percentage of observations of each class.

2.7 HYPERPARAMETER OPTIMIZATION

In the training phase, ML models change their internal parameters to better adjust to the problem, in order to produce better predictions. However, models' operation and structure is affected by a group of parameters that could be defined by the data scientist, called hyperparameters. Adjusting this parameters can be very important to produce better predictions since every problem is different and finding the best combination of hyperparameters could result in significant improvements in the model's performance. There are several methods used to find the best combination.

2.7.1 Manual Search

The most basic search is the manual search. It consists in the usage of the data scientist's knowledge and expertise to build the model with the best parameters. Some iterations with different parameters are conducted, one by one, with the data scientist evaluating the model's performance at each step until it is acceptable.

2.7.2 Grid Search

Instead of having the programmer running the models and adjusting the parameters manually, a automatic search over a set of parameters can be done. The set of parameters is still defined by the data scientist but all the possible combinations between them are performed automatically in the grid search. Every combination is evaluated and, in the end, the model with the best performance is saved, as well as the best hyperparameters. There is, however, a problem with grid search: the hyperparameter domain is still adjusted by the data scientist, which can result in a lower performance if the scope is not right, and the method is computationally intensive. The data scientist cannot simply choose all the possible hyperparameters because there could be millions of different combinations, making the grid take a lot of time. It is better to use grid search when the data scientist is sure of the hyperparameters that could provide good results.

2.7.3 Random Search

When exploring through possible good hyperparameters, random search is a good alternative to grid search. Although grid search ensures that the best possible combinations is found, because all the combinations are tested, the random search is still capable of finding a similarly good result in less iterations. Instead of performing calculations through every possible combinations, it tests the hyperparameters randomly. With this method, the data scientist is able to explore much more hyperparameters and, after seeing the results obtained from the tests, he can choose a subset of hyperparameters that shown the best results.

2.7.4 Bayesian Search

In this search the hyperparameters are adjusted based in a regression. From a set of hyperparameters, the model tries to guess what would be the subset of parameters that could provide with a better performance, using the previous tests' results. The combinations could be similar to the previous ones but, sometimes, other parameters are randomly chosen. This helps the model to follow the best hyperparameters at each time but also to explore all the values' hyperspace.

2.8 RELATED WORK

Over the past years, an increasing number of studies have been conducted in order to better predict the customer 's behavior over telecommunication companies. Before the rise of ML, companies used all kinds of statistical models to predict situations that could affect their financial value. Whereas some of these models could achieve a good performance, they are not able to use all the

data available in today is systems. ML is, therefore, used to allow a better use of this available data and to improve the overall quality and velocity of the predictions. Special care was given to churn situations, therefore the majority of the articles tend to cover this problematic. Customer churn destroys profits and reduces the overall shareholder confidence. In the beginning of the decade, churn rates reached 25% in Europe and acquiring new customers cost five times more than maintaining the current ones (Yan, Miller, Mozer, & Wolniewicz, 2001). It is also expected that customers who switched from another company continue to do so in the present one, making this strategy a more risky one (Larivière & Van Den Poel, 2005).

In (Hung, Yen, & Wang, 2006) were used two approaches to predict churn that consisted of a decision tree and a back propagation neural network. In order to better classify the probability of churn, a K-means clustering method was used to divide customers in different churn sections, which replicate value-loyalty segments. The division took in account the billing amount (and, therefore, the raw current value of the customer), tenure months (to introduce the importance of loyalty in the model) and payment behaviors (to replicate credit risks). This division allowed to assess if the customer is behavior over the different clusters is different, which would lead to the creation of distinct models for each one of the clusters. The data elements fed to the models consists of demography, customer usage, billing and customer service interactions. The group measured the performance of different models and applications and concluded that clustering the customers in different segments contributed to a better performance. They found some difficulties due to the limited number of churners present in the dataset. They were also presented with a decrease in performance in the sixth month of validation. This could be due to several external factors like competition, natural disasters, and others. A method to counter this fluctuations in performance could be adjusting the data fed to train the model to a lower interval. The importance of using more or less historical data is approached in (Yan et al., 2001), where the use of more data in the same time window and the extension of this time window is debated. In the study, it was concluded that using only 50% of the data available in the defined time window did not contributed to a loss in performance. Sometimes, more data does not translate in more quality or performance for the model. When trying to get more historical data by increasing the time window, a new problem emerges: the customer may have a tenure of less than the time window. The work around found was to create several aggregations considering several time windows and from shifted moments in relation with the test data. When doing extractions from different moments and with different time windows the storage needs and the training difficulty increase. A solution to reduce the training difficulty is to produce several individual models for each one of the time windows and shifted time frames. Each one of the individual models would then be combined in a ensemble and retrieve a weighted prediction. The conclusion was that the ensemble method contributed to a better classification and should be taken in consideration in future or similar approaches, like the one being studied. A common problem when training models is including false predictors or information that is not available when the model is put to the test. In (Yan, Biosciences, Wolniewicz, & Computing, 2004), this situation is discussed and batch updates

are identified as the most probable cause for this types of error. The data sources provided to our study come from different systems and are built in different ways. Some are constantly updated, others perform batch operations overnight and others perform operation at the month end, turning this into a real problem that should be taken in consideration.

Probably, other studies using machine learning techniques have been conducted in telecommunication companies. However, this companies tend to keep them hidden from competitors, since the studies can reveal information about the business model or motivate them to explore similar situations.

In Neural Networks and other ML models it is not possible to introduce domain knowledge in the training phase. In (Yan et al., 2004), the solution for this problem is to modify the data fed to the model in order to represent this knowledge. This can be through clustering or techniques that modify the data but do not increase the feature cardinality or just by creating a new feature that could introduce the domains knowledge into the data. Feature extraction done with a good domain knowledge also contributes to a good model training, as the quality of the features extracted can impact a lot on the model is performance. Prior to the feature extraction phase, however, the features selected from the data available are also important. People tend to maintain all the possible attributes they have but some only add noise or complexity when training the model.

However, even though deep learning approaches show better performance in solving several problems, when compared to classical machine learning, there are not studies applying them to telecommunications or other related fields. This could be due to data being often in a tabular format, whereas deep learning techniques are more often used in image, sound or text contexts.

3. RECURRENCES DATASET

3.1 MATERIALS

The goal of this work is to develop a ML model to predict when a client will suffer a recurrent problem in a telecommunication service. To make this possible, the complete historic of problems and efforts done in his resolution is needed. Each client being different, according to factors like age, literacy, need for the service and other variables, acts in a different way when a problem occurs. There is also a big amount of different problems in each field of service of the telecommunication company, making harder the process of identification of the problem and troubleshooting. Technical diagnosis and technical reparations could also be needed and should therefore be considered when trying to predict if a recurrence will happen. The complete dataset will then summarize all the interactions between client and customer service, all the technical analysis and all the technical interventions. It will also compile information about the client profile and services usage. Python will be used as the main programming language to conduct the work as it has many libraries and frameworks that support ML development, with efficient and simple implementations found easily. Python was used along with other tools like Jupyter Lab, Keras, TensorFlow and Docker.

3.1.1 Working Environment

Docker

Docker makes possible to implement Operative System (OS) virtualization in a simple and self-contained environment. The containers docker creates act like a simple and stripped OS version, isolating all the main system in a kind of virtual machine. Even though it allows to make an isolated system, several containers could work simultaneously in the same system using the same physical resources like the CPU, GPU and RAM. Working with amounts of data like those available compromises the efficiency of the common laptops and workstations, making Docker a go-to solution, allowing corporate users to efficiently use company servers. To start a Docker, the software builds the software images available from the base environment and adds the specific changes required. Some changes are due to the necessity of the user and the work that must be done. In this specific case, modules like Keras, TensorFlow, Numpy and others will be needed.

Table 3.1: Hardware in which Docker was running.

Resource	Amount
CPU Cores	12
RAM	64 Gb
HDD	1 Tb

Jupyter Lab

In order to better organize and develop the code for the project, the Jupyter Lab IDE was used. Jupyter Lab is an evolution of the well-known Jupyter Notebook, a browser-based tool that supports workflows, code, data and visualization. The tool allows to split pieces of code, text and output in several different cells. This allows to run pieces of code independently, removing all the computational overhead that happens every time something fails.

Keras and Tensorflow

Keras is a high-level library that provides an easy and quick way to develop DNNs. Together with Tensorflow, who also allows to efficiently build powerful bases for the network, they allow for fast prototyping. Having high capability GPUs available also makes the use of this tools more important, because they can use the computational power available with their GPU support. In fact, although Keras running on top of TensorFlow, sometimes it is needed to use plain TensorFlow to better tune the network structures.

Python and Packages

Python was used as the base language for the project. It offers a wide range of packages that allow to read and transform data as needed. *pandas* is one of the main packages, used to read the data from the sources and to create dataframes with it. Dataframes are structures that facilitate an efficient data manipulation. *numpy* has a wide range of mathematical functions that allowed for transformations and calculations in the dataframe. *seaborn* allowed to get better data visualization, a very important measure to understand the data and what exactly is needed to do. In the end, *scikit-learn* was used to apply sampling techniques and to create and optimize classical machine learning models, as it offers a very good support for a wide variety of models.

3.1.2 Data Description

Customer Support Dataset

In the customer service dataset, each line represents an operator's interaction in the system, relative to a customer. It's possible to verify that each ticket has, associated to it, columns with information about the following subjects:

- Administrative: what are the systems that promoted the creation of the ticket and if the administrative team, responsible for internal problems, was needed;

- General customer data: identification key, type of contract, obligations, previous recurrences;
- Ticket related data: identification key, service in observation, motive for the call, responsible department, receiving call operator.

Customer's Personal Data Dataset

Each line describes the useful information that the company has for each customer. It consists of attributes such as the region of the country where the customer lives, the amount the customer pays monthly, gender, whether the electronic invoice is active or not, whether it meets the payment deadlines and also in which billing cycle it is inserted. There's also information about the calculated customer's age.

Customer's Service Usage Dataset

Each line of this dataset stores information about the usage of the contracted service of a customer, from mobile data to broadband call time used.

Technical Interventions Dataset

This set of data represents all the technical surveys that have been performed remotely on customers, as well as the on-site interventions needed.

Therefore, when a customer has a technical problem with their services, they try to solve the problem by phone call instead of immediately scheduling a personal intervention at the customer's home, which has an additional cost for the company. This dataset is, therefore, made up of attributes such as the client's equipment model, contracted tariff, problem and symptom described, some steps taken, which results from the screening, etc.

If there's a need for a specialized team to solve the problem on the customer's location, that intervention is also registered. In this case, information about the technician, the time spent and the tests performed are able to be used in the dataset.

3.2 DATA UNDERSTANDING AND PROCESSING

3.2.1 Business Understanding

Telecommunication companies have made a huge contribution to the increasing use of digital devices by providing the infrastructure for ordinary users. They end up with large data pool from which information can be extracted and converted into knowledge.

Customer support services have evolved significantly in recent years, and most simple problems can usually be solved with specific software without direct human interaction. Nevertheless, customers sometimes still have to contact a specialized operator directly. In order to provide the most cost-effective support service, customers usually first contact the general support operator, who is not specialized in any type of problem. As a rule, generalist operators are unable to solve technical problems and to schedule a call between technical support and the customer. Accordingly to the protocol, operators typically apply a number of automated processes that can solve the reported problem. Sometimes the problem seems to be solved, but the solution is temporary and the devices show the same symptoms later on. According to the data in our study, customers call the customer support again in about 10% of cases within a month recurring in the previously claimed problem. Operator time was then wasted and the solution was unsatisfactory, which added to the customer's annoyance. It is therefore important to avoid these situations, but also to predict whether customers will recur.

Nowadays, companies store a lot of information about their interactions with customers, including a detailed history of previous calls. This information can be used to determine how often customers return and why the complaints were motivated. The perfect scenario is that no customers complain because the contracted service is fully functional. Improving the customer support efficiency and the relation with the customers is of great importance since it promotes the decrease of churn, increasing company's profits. The goal of this work is to improve customer satisfaction by lowering the number of recurrent problems, i.e., problems that occur within a month for the same reasons, following the company's norms. A monthly time frame is useful in telecommunication companies because all the processes and payments are in that same time frame. This will be achieved by predicting when a problem will be recurrent for a customer after each interaction between him and the support service. With this information, companies can use automated processes that are typically used by a technical support agent and can sometimes solve the problems described.

The historical service data of the dataset being used has registered all the interactions between customers and support services operators, as well as the history of service usage of each individual customer. The goal is to use this data to train a deep learning model to predict when a customer's problem reported in a call will recur, with the complete process being presented in figure 3.1.

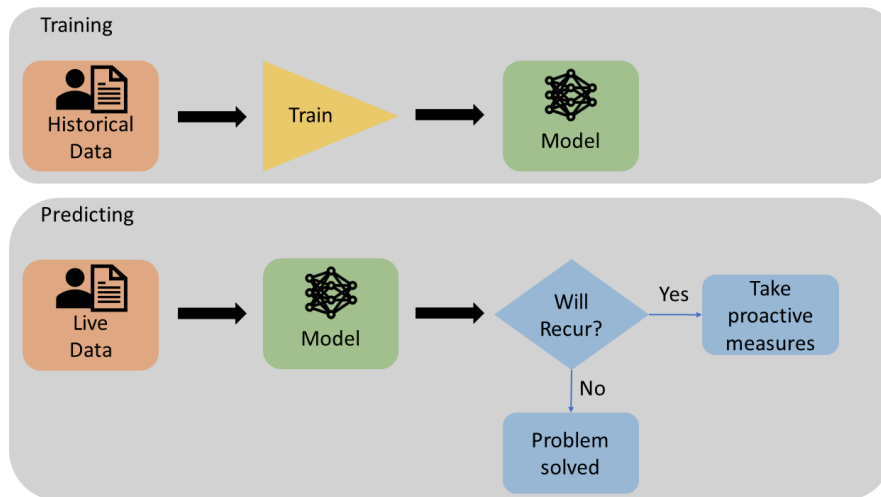


Figure 3.1: Predictive model usage in business.

3.2.2 Data Integration

The data was cleaned and transformed to achieve the best possible performance in the predictions. Since the data comes from different sources, the main data set must be selected, in which all other information is added. In this data set, for each entry that corresponds to an interaction between customer service and the customer, there is information about: the contracted service and technology, the data for the start and end of the interaction and the corresponding follow-up, the problem described and which of the support areas took care of the situation, as well as information about the start and the end of the expected term of the contract. Finally, to learn more about the customer, information about service usage of the last few months was collected. This data source contains information about which services have been contractually agreed and how often they are used (Fig. 3.2).

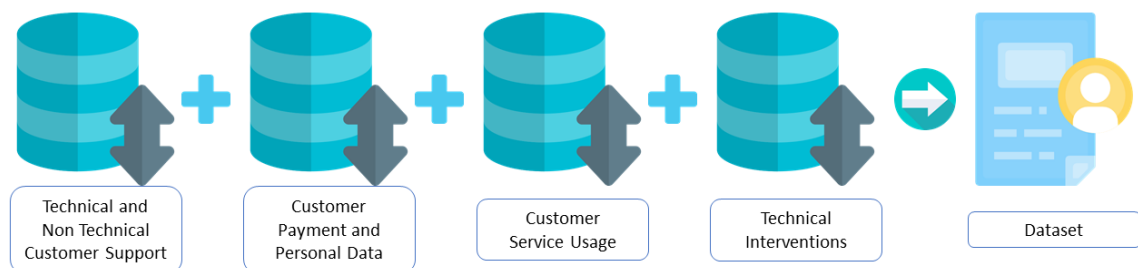


Figure 3.2: Data sources that make up the dataset.

Although not every entry corresponds to a call, the majority of them do. The ones that are not related to customer support call were removed from the dataset.

Target Variable Creation

When a customer calls to the customer support service, the problem reported by him is registered by the operator. The problem is typified in 4 levels of detail, hierarchically.

As the objective of the work is to predict if a customer will recur, i.e. if within a month a customer will need to contact the customer support for the same unsolved problem, the target should be created. In order to do so, firstly it's needed to identify what is the problem that the customer faced and if it could be considered the same as other problem with a different typification. In fact, an internet problem could have the first 3 levels with the same value (level 1 - Technical Problem, level 2 - Internet, level 3 - No connection) but the 4th level could have a different value (Lights blinking vs. No lights), although the customer is having the same problem: an non-working internet connection.

After taking a deep analysis, the first 2 levels of problem description were considered enough to represent the customer's problem. There are 148 distinct values for the 1th level of typification but the 25 most frequent are able to represent 88% of the dataset. For this reason, a manual mapping of the first 25 most frequent level 1 typifications were mapped with the corresponding level 2 typifications, whereas the rest was done automatically. This step was needed to ensure that different typifications about the same problem didn't exist among the most common problems.

After the join of the two levels of typification, the target problem has 1009 distinct values. With the 100 most frequent typifications (10%) is possible to represent 90% of the calls, like shown in figure 3.3.

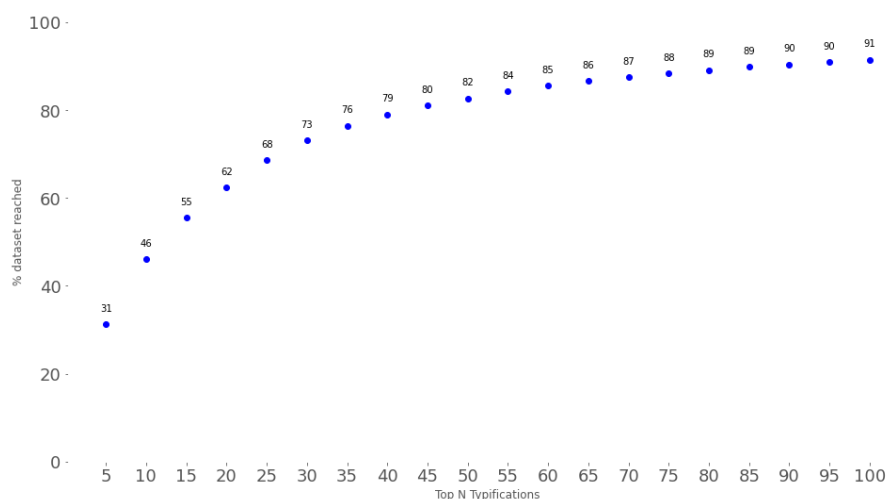
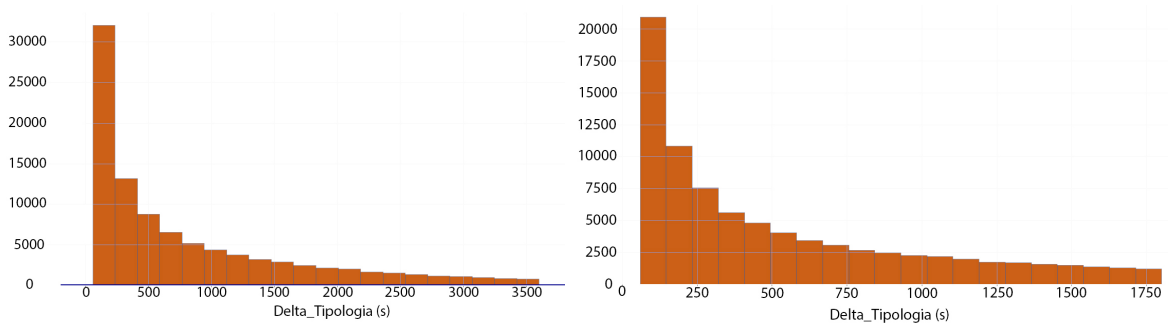


Figure 3.3: Amount of data entries covered by the Top N most frequent typologies.

With the problems correctly defined, each customer and his corresponding tickets were filtered and compared to all his other tickets. That allowed to know if a specific ticket had a similar one within a month, indicating a recurrence. When considering the time difference between tickets, it was possible to observe that some tickets reported the same problem in a small interval of time

between them. This occurred because sometimes needs to propagate information to other systems, generating duplicated entries. The operators can also do a mistake and submit the same problem twice in the system, generating more duplicated entries. Transmitting the call to another operator can also be a cause of this situation. In order to fix this problems, a deep analysis on the time between possible recurrence calls was conducted. In figure 3.4 it is possible to see that in the first 30 minutes the number of calls is quadratic, while past that time it gets linear. Figure 3.5 shows the volume of call after the first 30 minutes between a call, with a linear descend. This was the threshold considered necessary in order to avoid duplication, so all the entries that have the same problem (from the same customer) within 30 minutes were deleted.



(a) Between 1 minute and 1 hour.

(b) Between 1 minute and 30 minutes.

Figure 3.4: Recurrence volume of call with the same problem in short time.

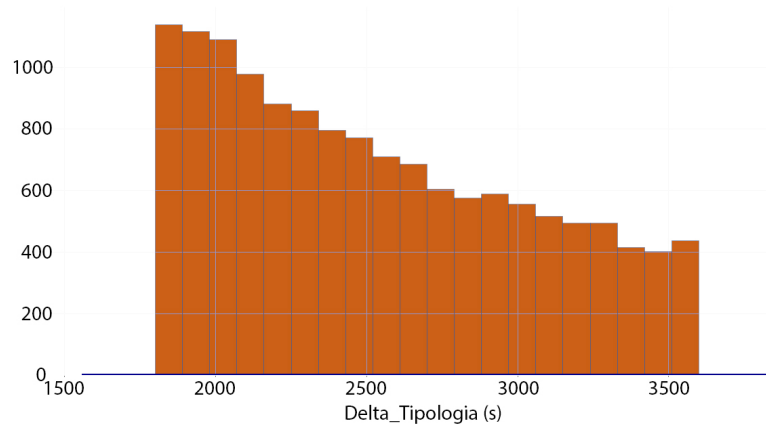


Figure 3.5: Recurrence volume of call with the same problem between 30 minutes to 1 hour.

In the end, there were about 3.4 million (90,9%) entries without recurrence and 340 thousand (9,1%) with recurrence.

Exploratory Data Analysis (EDA)

Problem category and typification After the join of the two levels of typification, the target problem has 1009 distinct values. With the 100 most frequent typifications (10%) is possible to

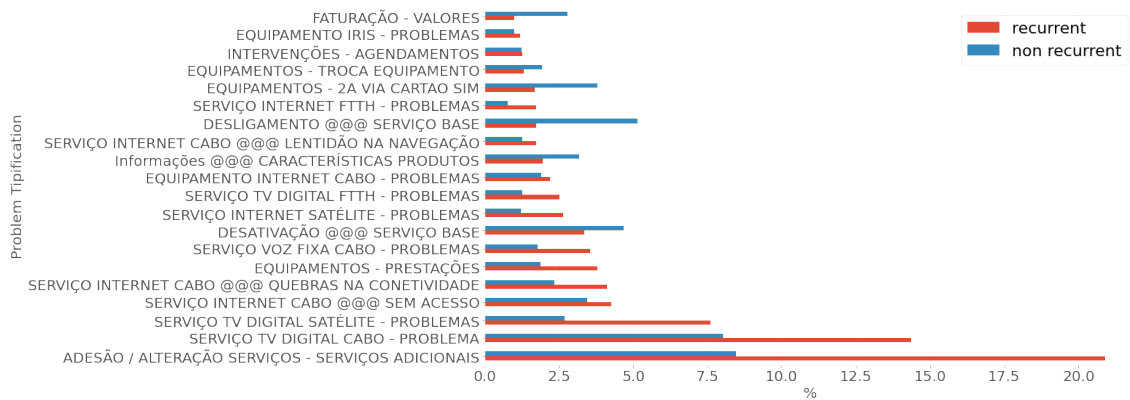
represent 90% of the calls, like shown in figure 3.3. It is also possible to understand that the 20 most frequent typifications have the highest percentage of dataset with the value decreasingly fast after those. Therefore, it is worth to analyze the 20 most frequent typifications (Fig. 3.6a).

As seen in figure 3.6a, different typifications show different probabilities of recurrence. When comparing the first typification of the graphic ("ADESÃO / ALTERAÇÃO SERVIÇOS - SERVIÇOS ADICIONAIS") with the eighth ("DESATIVAÇÃO @@@ SERVIÇO BASE"), this becomes even more clear. The first typification, related with customers calling to add or alter extra services to their base contract, shows a higher number of recurrences. This may be due to the fact that customers call several times before actually making a new addition to the service, either for knowing better the technical aspects of the service or to get price information. Changes to the service can also incur in this motivations for recurrence.

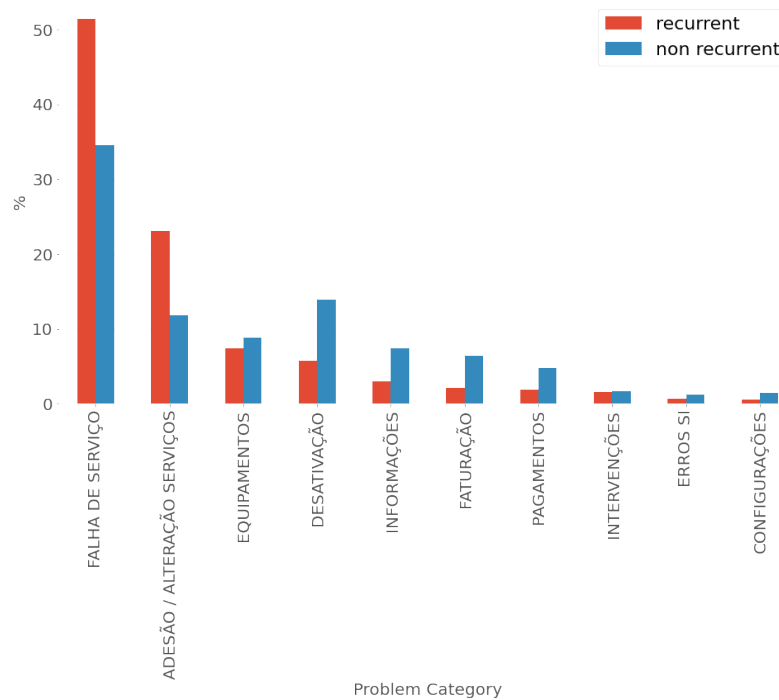
This typifications fit in higher-level groups called categories. There are more than 1000 different typifications but only about 100 categories. The 10 categories with higher frequency are represented in figure 3.6b. Like in figure 3.6a, it is possible to observe that some categories show a higher probability of recurrence.

Previous interactions with the customer support service Knowing of previous problems of the same typification can hint difficulties in solving some of them. More than that, knowing if any problem of the same typology has even occurred can help to understand if a recurrence will happen in case of a new problem report. From figure 3.7a it is possible to understand that problem who have happened already in the past 180 days tend to happen again and recur. This can be due to a problematic service installation, where the customer feels month after month problematic behaviors. On the other hand, a customer may also start feeling that his service is not working properly and his dissatisfaction makes him call more frequently and being more sensible to the service problems. That being said, customer with no incidents of the same typology in the past 180 days tend to not recur, whereas those who already had similar problems tend to recur when trying to solve a new one.

The number of previous incidents of the same typology in the past 180 days also helps to understand how interactions with customer service happens. As expected, the frequency of the number of previous incidents decreases the higher the number of incidents. Naturally, just one customer service interaction is needed to solve a certain problem and that is reflected in figure 3.7b. But more than this, it is possible to observe that as the number of previous incidents increases, the recurrence ratio decreases. This shows that the higher the number of contacts with the customer service, the lower the probability of recurrence. It is also possible do understand that the ratio of recurrence tends to decrease rapidly from a number of incidents between 1 and 4. Nevertheless, there's always a higher probability of recurrence on customer who have already have incidents than those who did not.



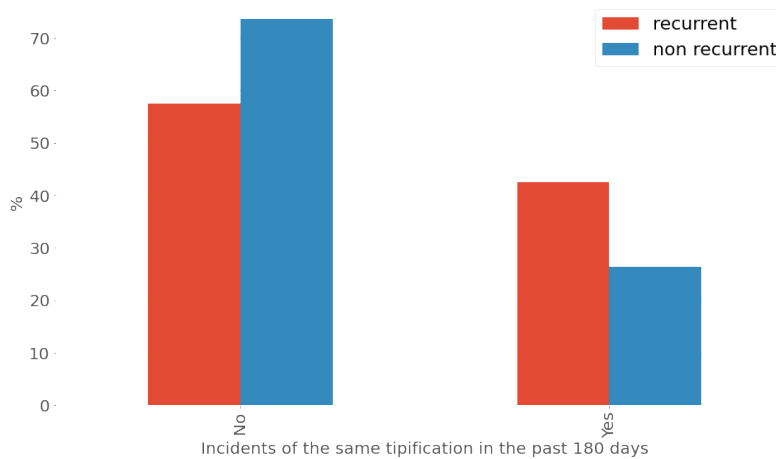
(a) Problem's typification relation with recurrence.



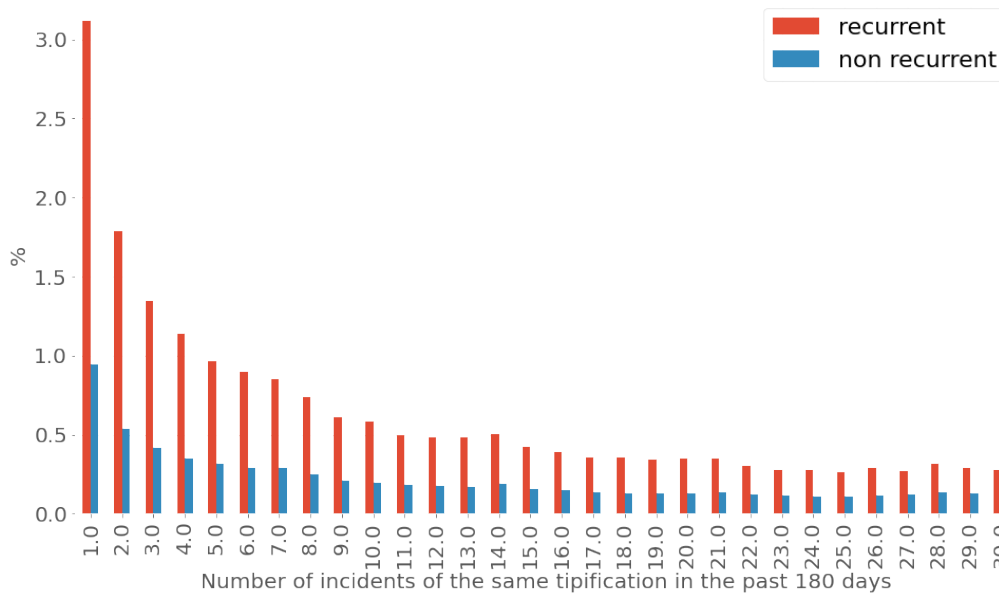
(b) Problem's category relation with recurrence.

Figure 3.6: Typifications and Categories relation with recurrence.

These problem can be reported to several teams, depending on several factors. As an example, is a customer reports a problem in a store, then the team responsible for opening a new ticket will be the "LOJAS" (stores), like seen in figure 3.8. Naturally, not all the problems are the same and usually stores are reached by customers asking for informations more than reporting for a technical problem. In the same figure, it is possible to observe that the team "TÉCNICO" (technical) is the one with the higher ratio of recurrences. This is understandable since technical problems usually need more resources to be solved and frequently need tests and iterations, leading to recurrent contacts.



(a) Occurrence of same tipification problem and target relation.



(b) Number of occurrences of same tipifications and target relation.

Figure 3.7: Same typology problem occurrences within 180 days target relation.

Time spent in a customer support call In a similar way, the time spent in a call with the customer service support may also give an indication about the probability of recurrence. Figure 3.9 shows the relation between time, in minutes, spent in a call and the probability of recurrence of a problem with the same typology. As it is possible to observe, shorter calls often result in no recurrence. This is not the expected outcome of a shorter call time, since a higher time should mean more resources into the problem's resolution. However, when taking in consideration the behavior of the company's information systems, this figure is understood. In fact, for the majority of the problems who are solvable in a single call (like customers asking for informations), customer support operators often create and end the support ticket at the same time. This causes the lower call time registries to have this noisy data into them, showing lower recurrence ratios than the rest higher call time ones.

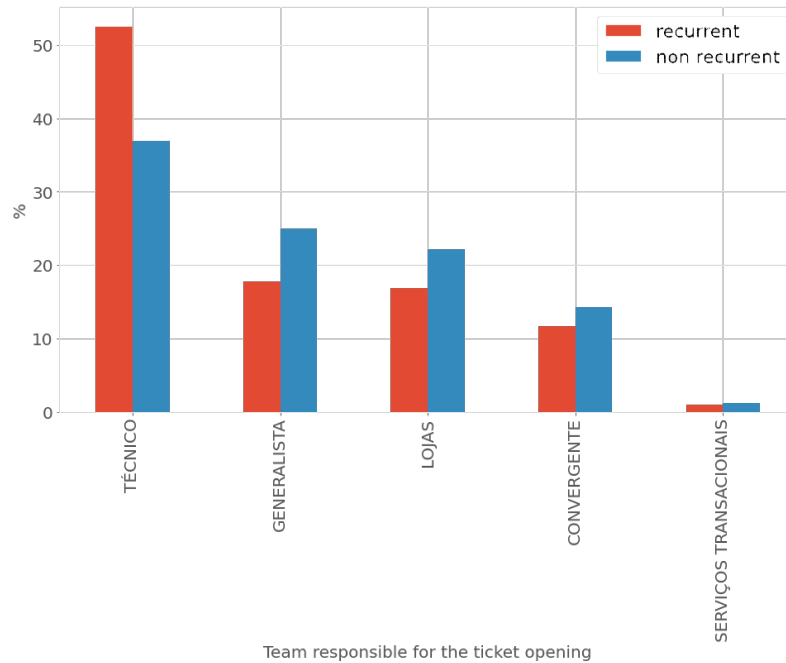


Figure 3.8: Teams responsible for ticket openings.

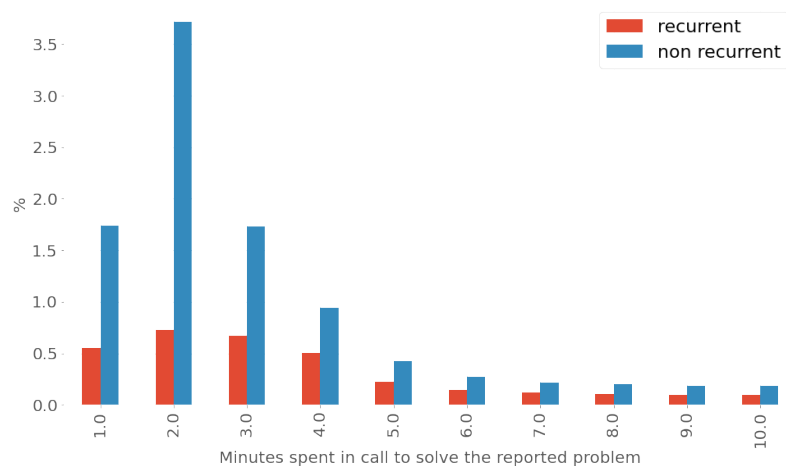


Figure 3.9: Time spent in a customer support call importance.

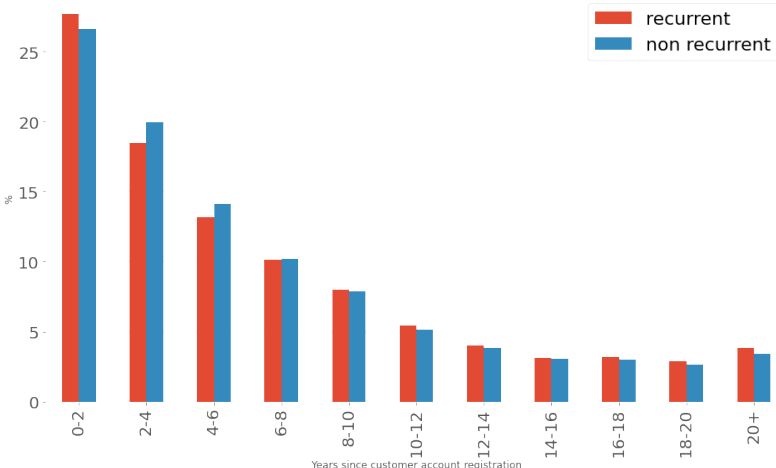
Time left to the end of the contract The time left until the end of the contract is also really important to explore, since it could represent an opportunity for customers to renew their contracts with better benefits or even changing to a new service provider. This opportunity to change could make customer more susceptible to complaints about the service. In figure 3.10a, it is possible to see the recurrence and non-recurrence percentages of customers with different months left until the end of the contract. The 0 months mark represents customers that are not under any kind of retention. The rest of the figure shows the increase in customer complains the farthest they are from their end of the contract. The most frequent categories (Fig. 3.6b) suggest that this could be due to initial contract complains (in case the service is not going accordingly with the contracted or expected), addition of extra services, equipment requests or changes or even informations about

the service. It is, however, interesting to see that a small bump occurs in the 12 month mark. This could be due to customers trying to get new benefits, changes in promotional offers (as an example, TV Boxes are often offered with a discount in a yearly base, instead of a full contract) or just because some customers have contract of only 12 months. It is important to note that the majority of the contracts are established for a 24 month time period.

It is also interesting to observe that customers within a range of 0 to 2 years of account age tend to recur more often (Fig. 3.10b). This situation can occur due to initial service setup but also because customers who frequently change between different service providers have higher level of technical needs.



(a) Time left to the end of the contract importance.

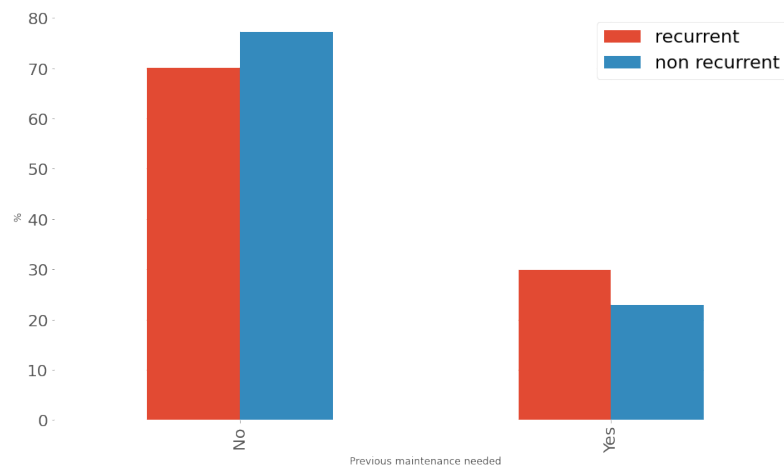


(b) Customer's account age importance.

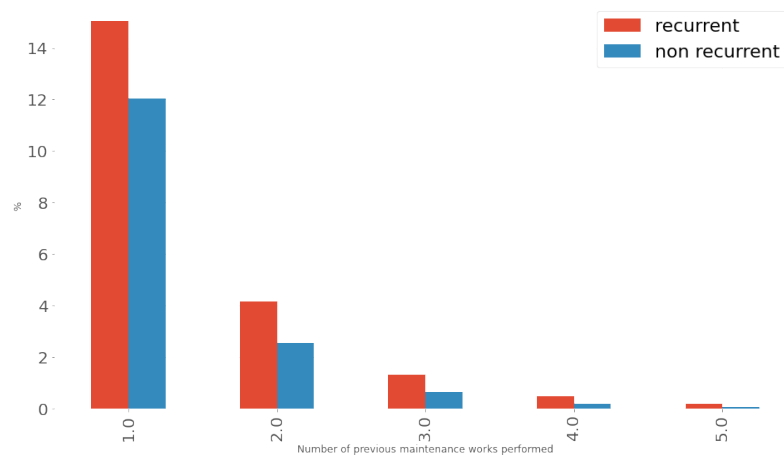
Figure 3.10: Retention and account age relation with recurrence.

Maintenance works Similarly to the time spent in a call, maintenance works show how many resources were applied to solve a certain problem. A higher number of maintenances shows either signs of a bad maintenance procedure or a location with an infrastructure problem. In figure 3.11a it is possible to see how previous maintenance works hint about the recurrence probability. When customers don't have any previous maintenances made they more often don't recur. In opposite way, customers who have already seen their service maintained tend to recur.

The amount of maintenances work performed, in figure 3.11b, show that the more maintenance works are performed, the higher the probability for recurrence. This supports the idea of customers having problems or complains that are not fixable with a maintenance but instead are due to an infrastructure problem.



(a) Previous maintenance needs.



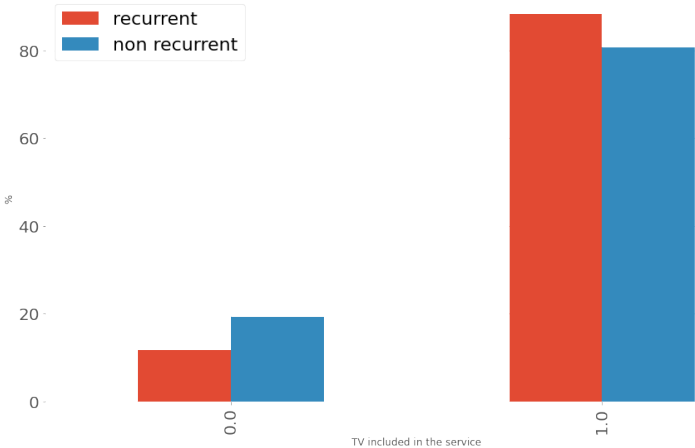
(b) Number of previous maintenances performed.

Figure 3.11: Maintenance's relation with recurrences.

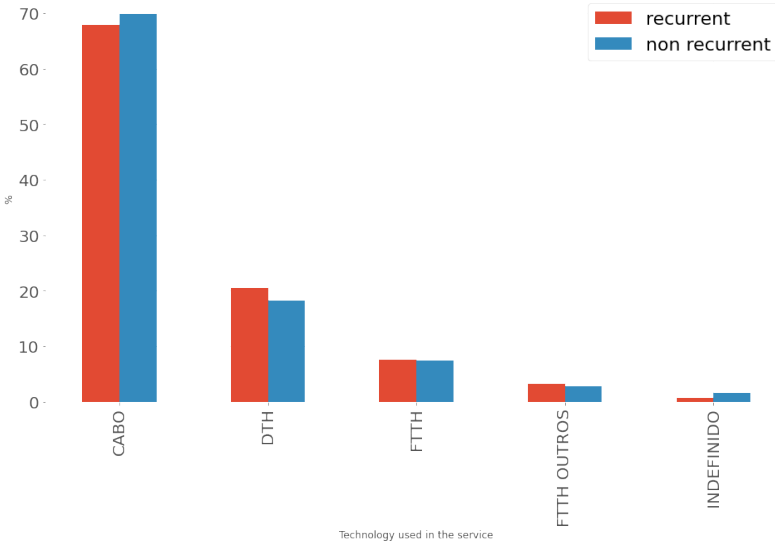
Customer's Services Contracted Users with more services contracted have, naturally, a higher probability of their service failing. Furthermore, these customer tend to be more technological and demanding, noticing any failure and reporting that same problem. In 3.12a it is possible

to see that customer that don't have TV service included tend to recur less, which may be due to having a lower probability of failure or due to them being more demanding of the service.

The technology used in the base service installation is also to be taken in consideration when trying to understand the outcome of a customer contact. By the figure 3.12b it is possible to understand that customer with DTH will more probably recur in a problem in comparison to the ones who have FTTH.



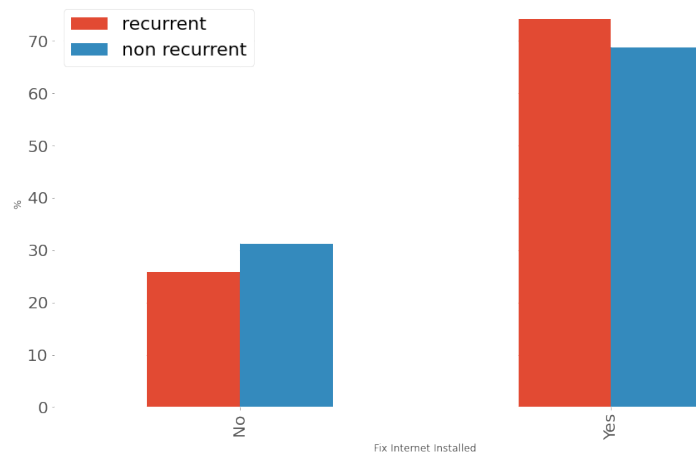
(a) Predictive model usage in business.



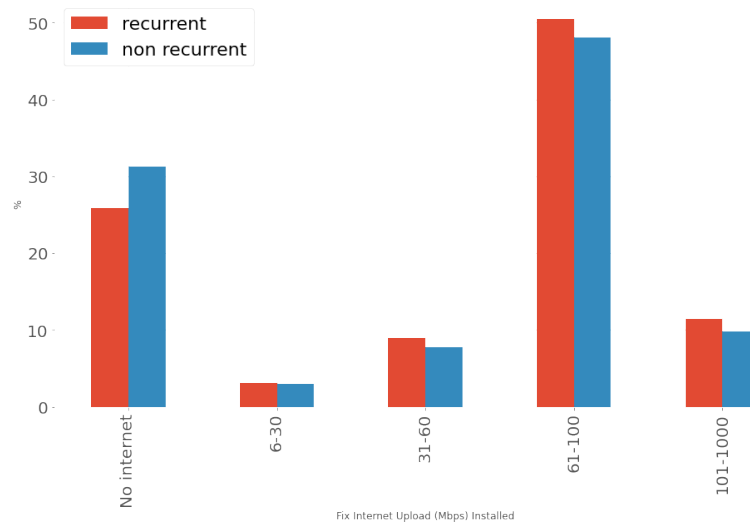
(b) Predictive model usage in business.

Figure 3.12: Contracted services and technology relation with recurrences.

The customer's expectations are also to be taken in account when trying to predict recurrences, since more demanding customers tend to recur more often. Through figure 3.13a it is possible to observe that customer who have internet service contracted usually recur more often but there are no hints about the motive this happen, like in figure 3.12a. However, after analyzing the figure 3.13b it is possible to observe a higher recurrence in customers with higher internet speeds contracted.



(a) Predictive model usage in business.



(b) Predictive model usage in business.

Figure 3.13: Contracted services and technology relation with recurrences.

3.2.3 Data Preparation

Cleaning

In order to reduce the noise fed to the models and improve the performances, the dataset needed to be cleaned. Among other problems, this cleaning tried to remove rows and columns with data having no variance, repeated, unnecessary or null. Furthermore, specific rows also needed to be eliminated due to repetition, errors or simply because they had data not related to the problem, created by the internal information systems' operation.

Remove Unwanted Observations Firstly, all the rows not related with the base (TV, Internet or Landline) or mobile (Cell phone or Broadband) services were removed. Also, all the entries which state could still be changed (due to ongoing resolutions) were removed, as they didn't allow

to correctly trace the procedures and their outcomes. Some entries also didn't have customer account information or where null, making it impossible to have important data about the customer itself and were also removed. This situation could occur when people reaching out to the service provider aren't customers yet.

Customer support operators have the possibility to insert additional information to the ticket, which leads to great inconsistency in the field that stores this data. There are also personal information fields that are all filled with the same data or repeat other columns information. Some fields with information about the internal system who created the entries were also unnecessary and were removed.

In opposition to the previous related situation, which were full columns or rows to delete, there are several rows to delete by specific conditions. Some examples are:

- Entries from different systems that the one encharged by customer complains;
- Entries that are not caused by the customer but, instead, by the customer service reaching out to the customer in a pro-active way, as this situation cannot be considered a recurrence;
- Entries created by very specific teams, that do not participate in the customer support service itself but, instead, have the permissions to work some problems related with customer support;
- Entries duplicated due to automatic system propagation procedures.

Handle Missing Data Although customer support operators have the possibility to add their own ticket information this doesn't occur frequently. In fact, there's not enough detail on the problem to complete all the automatic fields on a new customer support entry. These fields have a very high null frequency and were deleted from the dataset.

In order to clean the data is also important to not only delete rows or columns but also to complete the fields with null values. This is important because deleting any entry that has a completable null field means a lost of information about a particular individual, meaning that all the entries related to it would loss their value to the model. Some examples are:

- Customer age values, replaced by the median;
- Account age, replaced by the median;
- Service information (like TV box count, internet speed) that, sometimes, is not registered in the ticket but it available through other data sources.

Fix Structural Errors Some columns had information about previous recurrences of the same problem for each level of description available. Those, however, were filled incorrectly. After analyzing the data it was possible to see that due to system propagation a entry could be repeated for several times. This lead to an extensive work of filtering the repeated entries through several conditions.

Table 3.2: Examples of target problems count and distributions during the cleaning process.

Target Problem	Before Cleaning	After Cleaning
ADESÃO / ALTERAÇÃO SERVIÇOS - SERV. ADICION.	621395 (10,6%)	360025 (9,6%)
EQUIPAMENTOS - 2A VIA CARTAO SIM	305890 (5,2%)	135228 (3,6%)
EQUIPAMENTOS - PRESTAÇÕES	198729 (3,4%)	77094 (2,1%)
EQUIPAMENTOS - TROCA EQUIPAMENTO	150427 (2,6%)	70540 (1,9%)
Informações @@@ CARACTERÍSTICAS PRODUTOS	121885 (2,1%)	115261 (3,1%)
PEDIDOS @@@ TRANSFERÊNCIA DE MORADA	112655 (1,9%)	984 (0,0%)
FATURAÇÃO - VALORES	108315 (1,8%)	98424 (2,6%)
DESLIGAMENTO @@@ MÓVEL PRE-PAGO	97203 (1,7%)	27123 (0,7%)
PEDIDOS @@@ FORMULÁRIO MÓVEL	88767 (1,5%)	43103 (1,2%)
PEDIDOS @@@ ALTERAÇÃO TITULARIDADE	71437 (1,2%)	321 (0,0%)
Total (including all the other problems)	5884432	3746847

After doing performing this operations, the data suffered heavy changes in number of entries but also in the problem's distribution, as seen in table 3.2.

Feature Extraction

Extracting additional features increases the data quality, thereby improving the chances of a higher performance model. As many of the problems reported are due to technical difficulties or malfunctions, getting information about the installations is very important. For this reason, the following features were extracted:

- #Manutencoes - a count of the number of maintenance works performed up to the date of the ticket; This includes all the works related with fixing reported problems.
- #Instalacoes - a count of the number of different installations performed; This includes all the new installations of the different services a customer may have gone through.
- #OTAlteraServico - a count of the number of times the service has changed and a technical team had to do equipment changes; This includes all the works performed due to service changes like changing internet routers due to higher speed contracts or changing TV boxes due to higher quality sources.

- #OTRestabelecimento - a count of the number of times a technical team had to perform works to re-establish service connection; This includes situations where, for example, a cable gets destroyed by an environmental situation.

There is, however, much more data that can be extracted from the data sources. As the objective of the model is to predict recurrences, it is important to get information about previous problems, the time of customer reporting a problem, the alterations in service usage and the timings in which the problem is reported. In order to better understand how the reported problem affected the user, the following features were engineered:

- delta_ultima_inc - which indicates the amount of time that has passed since a problem with the same typification has occurred. This could be months ago if the problem is not happening frequently or a week before it is recurrent already. If it never happened, the value would be virtually infinite;
- #Incidencias Tipo (X dias) - that holds a count of the number of times the reported problem has happened in the last X days (with X = 30, 60, 90, 180). This helps to understand if a problem is common and if it is becoming more recurrent or not;
- Dias existência CA (Ativação) - holding information about the number of days a customer has his account registered with the service provider. This gives an idea of the customer relationship with the company. This value could be binned into years to facilitate his understandability.
- Dt_min_Fim Permanencia_SA - which indicates the number of days until the end of the service contract. After this period a customer can leave the service without any penalization. This helps to understand how prone a customer could be to terminate the contract or asking for a renewal.
- up_down_ratio - representing the ratio of data used in upload or download;
- var_up and var_down - which relate the upload and download data used in the month before the problem was reported with the usage average of the previous months. This helps to understand if a customer usage profile has changed and, if so, it may be due to service constraints;
- var_t_airt_fixo - which is the variation of time used of the fixed phone in comparison with the previous months average;
- var_qtd_seconds_mon - that holds the variation of television watch time in comparison with the previous months;

- `t_sms` and `t_data_vol` and `t_airt` - the variation of SMS, mobile data and call time spent in comparison with the previous months of service usage;
- `delta_fecho_criacao_X` - that holds the time spent to solve a customer report, with X = minutes, hours, days. This helps to understand if a problem is taking a lot of time to solve or if it not enough time has putted on into it;

Some temporal features were extracted and a cyclical transformation was applied. This is very important because, for example, the day 30 of a month is very close to the day 2 of the next month. By representing the days of the month in a range from 1 to 31, this proximity is not represented. Applying the sine and cosine functions the days get

- `day_of_the_year` and `month_of_the_year` - allowing to get a better sense of the dates which customers use to call;
- `parte_dia` - giving information if the customers called in the morning, afternoon, night or in the dawn. This could hint about a possibly different quality in the solving of the problem;
- `week_day` and `feriado` - representing the day of the week in which a report was made and if that day was an holiday or not;
- `estacao` - indicating what the season in which the problem was reported. This could be important because meteorological factors sometimes impact the signal's quality.

Transformation

The transformation performed in this work consisted in both outlier treatment and encoding.

A vast majority of outliers were removed after the cleaning process, but some features still presented information that could result in a wrong interpretation for the models. This outliers are created by input errors or system anomalies and should be removed since some of the used models could be sensitive to them, as normalization is needed. The variables affected by the transformation were:

- Customer account age - the amount of years that the customer has been associated with the company. Since the company has 25 years of existence, any customer with an entry superior to that is an outlier. The value was replaced by the number 25, which is the maximum real value.
- Remaining contract time - the time left for a customer to end his subscription plan, which then is free to shut the service without any penalty. The maximum time of a customer retention is 2 years, so any value superior to that is also replaced by the 2 years limit.

- Customer support call time - the time a customer is engaged in a customer support call. In some cases, this calls are reported to take up to 4 hours, which are probably wrong values and were replaced by the value at 95% percentile.

The encoding was also necessary for the neural networks models, as they cannot handle multiple values in one column. The necessary strategy was, therefore, one-hot encoding.

In other models that did not require having one column for each different feature, the used method was label encoding as it is simple and fast to reproduce, without high computational or memory costs.

4. MODELING AND EVALUATION

4.1 MODELING

After the data processing phase, several experiments are needed to conclude about the best models to solve the problem. Predicting if a customer will recur in a contact with the customer support is a classification task. In order to deal with the numerical features' different scales, *MinMaxScaler* was used. Normalization was applied since it keeps the data's distribution but scales it to a range between 0 and 1. This range allowed for a direct use in deep neural networks (that need non-negative values), that being the main motivation to use it. Although standardization is useful in some situations, a lot of numerical features were not normally distributed, making it's use unreliable. Due to the data unbalancing, some sampling methods were set to be used. However, using the methods would change the meaning of each entry in relation to the previous ones from the same customer, reason why they ended up not being applied. Hold-out was the method chosen for testing. The data set was divided in a 65%-15%-20% way, with the first set being used to train the model, the second for validation purposes, and the latter for the final test. It is important to notice that no cross-validation was applied because that would not allow to keep the temporal sequence of data in the training phase.

Solution using classical machine learning models were tested, as well as deep learning alternatives. TabNet, a model developed by Google, who's claimed to outperform existing methods on tabular data was also tested. Because the problem to be addressed is an unbalanced problem with only 9.1% of cases recurring, accuracy is not the best metric to evaluate model performance. Instead, AUC-ROC and F1 scores were used to evaluate the models. Since these models are to be deployed by the company, the computing resources must also be taken into account. For this reason, the memory and the time that each model needs to train were taken into account.

4.1.1 Classical Models

Several classical models were trained in order to assess their capability of solving the problem. CatBoost, Random Forests, Logistic Regression, KNN, Naïve Bayes and SVM (with Linear kernel) were the classical estimators used, like seen in table 4.1, with different hyperparameters. Those hyperparameters were chosen to be tested due to their possibly meaningful impact on each one of the model's performance. The train was performed with 640k entries among all the estimators.

To test all the possibilities the grid search method was used. A selection of hyperparameters was made accordingly to the problem and to what should give the best results and grid search evaluates all the possible combinations. As there was a certain level of uncertainty using some hyperparameters due to the problem being very complex, grid search explores a wide variety of ranges if the user specifies them, while keeping the number of runs constant, instead of the bayesian method, which is a more novel approach.

Table 4.1: Hyperparameters tested for each estimator.

Estimator	Hyperparameters	Configurations	#Combinations
CatBoost	depth	[5, 10, 15]	18
	learning_rate	[0.01, 0.1]	
	iterations	[50, 100, 200]	
Random Forest	n_estimators	[10, 100, 500]	18
	max_features	[sqrt, log2]	
	min_samples_leaf	[5, 10, 50]	
Logistic Regression	tol	[1e-2, 1e-6]	24
	C	[1, 0.1, 0.01]	
	penalty	['l2', 'none']	
	solver	['sag', 'lbfgs']	
KNN	n_neighbors	[5,10,50]	24
	leaf_size	[5,50]	
	weights	['uniform', 'distance']	
	algorithm	['ball_tree', 'kd_tree']	
Naïve Bayes	-	-	1
SVM Linear	dual	[True, False]	10
	C	[0.0001, 0.001, 0.01, 0.1, 1]	

CatBoost

CatBoost was used because it shows very interesting results against other boosting algorithms like XGBoost in terms of performance and time. As it is a boosting algorithm it allows to weak learns to constitute a single strong one. For the data available this could be interesting since there are lots of distinct problems and perspectives to analyze.

The following parameters were tested with different values:

- depth - the depth of tree. Optimal values range from 4 to 10 but a depth of 15 was also tested because the problem could need more specific analysis;
- learning_rate - This setting is used for reducing the gradient step. It affects the overall time of training: the smaller the value, the more iterations are required for training;
- iterations - The number of iterations is related to the possible overfit or underfit of the model.

Random Forest

Random Forest is a bagging algorithm that creates trees with splitted parts of data from the dataset. This allows for the variance to be averaged. If the data contains columns that are very strong predictors those could be used independently in each one of the trees.

The following parameters were tested with different values:

- `n_estimators` - the amount of trees the estimator will use. This is important because the number of trees needed to solve a problem varies from problem to problem. More trees than the necessary would not make the performance worst but will eat more CPU time, while the lack of trees will degrade performance;
- `max_features` - this resembles the number of maximum features provided to each tree in a random forest. It is a good convention to consider the default value of this parameter, which is set to square root of the number of features present in the dataset. The ideal number of `max_features` generally tend to lie close to this value.
- `min_samples_leaf` - specifies the minimum number of samples that should be present in the leaf node after splitting a node. Allow to control the growth of the tree by setting a minimum sample criterion for terminal nodes. helping to prevent overfitting as the parameter value increases.

Logistic Regression

Logistic Regression was used because there is no clear evidence of a high correlation among the predictors, which could result in a good performance for this algorithm.

The following parameters were tested with different values:

- `tol` - tolerance for stopping criteria. The default is $1e-4$ but tests for lower and higher values were made;
- `C` - inverse of regularization strength. Like in support vector machines, smaller values specify stronger regularization;
- `penalty` - used to specify the norm used in the penalization. The default is `l2` but using no penalization could be beneficial so that was also tested;
- `solver` - algorithm to use in the optimization problem. `sag` is faster for large datasets and `lbfgs` is the default one, so both were tested.

KNN

KNN was used because similarities between entries and their problems could result in a clustering identifiable by the algorithm.

The following parameters were tested with different values:

- `n_neighbors` - represents the number of neighbors to use for neighbors queries. Using `n_neighbors=1` means each sample is using itself as reference, that's an overfitting case. More neighbors usually represent an improve in the test score. If too much neighbors are used, the performance gets worse;
- `leaf_size` - leaf size passed to `BallTree` or `KDTree`. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem so several were tested, while the default is 30.
- `weights` - weight function used in prediction. In uniform all points in each neighborhood are weighted equally. In distance, weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away;
- `algorithm` - algorithm used to compute the nearest neighbors. `ball_tree` and `kd_tree` were tested, and brute-force search excluded.

Naïve Bayes

Naïve Bayes was used because it cannot represent complex behaviour, therefore it would not overfit. It was tested more as a comparison and used as a guarantee that other models weren't overfitting. The Gaussian Naïve Bayes was used and no parameters are changeable.

SVM Linear

SVMs work very well with a huge number of features. Linear kernel was used because other kernels are more computationally heavy and take too long to fit.

The following parameters were tested with different values:

- `dual` - Select the algorithm to either solve the dual or primal optimization problem. Preferably `dual=False` when `n_samples > n_features`, so both were tested, with the default being `True`;
- `C` - Regularization parameter. The strength of the regularization is inversely proportional to `C`. The default is 1 but lower penalizations were tested.

All the tested hyperparameters were selected based on the most commonly used optimizations, which often produce the biggest impacts in the overall performance of the model. There could be other hyperparameters that could allow for a better optimization but considering them would exponentially increase the time of the experiments. With the limited amount of time, this would not be possible.

4.1.2 Deep Neural Networks

To get the best model, several experimental configurations of different fully connected MLPs were tested. When designing the models, two main architectures were selected: the first with 3 wide hidden layers and the second with 15 narrow hidden layers. According to (Karsoliya, 2012), architectures with one, two or three layers with about 2/3 of the neurons of the input size in each layer can achieve similar results compared to deeper architectures, which is why the first architecture was created. Fewer layers also contribute to a less computing-hungry model with shorter training times (Stathakis, 2009). However, deeper neural networks can sometimes find complex relationships within the data (Bianchini & Scarselli, 2014), which led to the second architecture being created for our tests.

Batch normalization layers, which are not shown in Table 4.2, were used between all the MLP layers in order to improve the speed, performance, and stability of the artificial neural network (Ioffe & Szegedy, 2015).

Table 4.2: Description of the different MLP structures used.

Structure ID	#Hidden Layers	Structure	#Connections
1	6	4673-3100-3100-3100-3100-1000	4.32E20
2	3	4673-3100-3100	4.49E10
3	15	4673-3100-2000-(2x)1000-(5x)500-(5x)100	9.05E39
4	2	3100-3100	9.61E6

The learning process is determined by the loss function, which in turn is affected by class imbalance. Various class weights have been tested to address this problem. After pre-evaluating the results obtained using equal class weights in comparison with balanced weights (King & Zeng, 2002), the latter were chosen due to the better performance obtained. For this reason, all experiments were carried out using balanced class weights. Binary cross entropy loss function was considered suitable for this problem. The ADAM optimizer was also chosen since it has been showing the best performances in similar problems (Kingma & Ba, 2014).

Since the architecture of the DNN is not the only factor that needs to be considered when creating a model, several hyper-parameter configurations were tested for each of the structures specified (Table 4.3).

The usual method for evaluating several hyperparameters in a neural network is to do some kind of grid-search. However, this method has a high computational cost and was not feasible due to time and computational limitations. Therefore, after analyzing some previous tests and leveraging the existing team knowledge, a set of hyperparameter configurations were chosen (Table 4.3).

The following parameters were tested with different values:

- **Batch size** - The batch size defines the number of samples that will be propagated through the network. At the end of the propagation, the network's weights are updated. This helps to reduce the memory needed for training the model because not every samples are passed at once, while increasing the speed of training. The risk with too low batch sizes is that they could make the optimization fluctuate too much.
- **Learning rate** - The learning rate is controls how much to change the model, accordingly to the estimated error each time the model weights are updated. A value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. The Reduce LR option cuts the learning rate by half (in this case, 50%) when there is not a big enough evolution in the performance, meaning that the model is reaching close to the best point.
- **Loss function** - The loss function is important for calculating the error of the model during the optimization process. Different loss functions are used depending on the problem, since the type of problem greatly affects what is considered to be a loss.
- **Optimizer** - The optimizer is responsible for updating the weights on the neural networks, accordingly to the loss function calculations. Different optimizers allow for a faster approximation to the optimum results.

Table 4.3: Hyperparameter configurations used for each different structure.

ID	Structure ID	Batch Size	Learning Rate (LR)	Reduce LR	Train Size
1	1	64	0.001	50%	320k
2	2	64	0.001	50%	320k
3	2	1024	0.001	50%	320k
4	2	10000	0.001	50%	320k
5	3	10000	0.001	50%	320k
6	3	64	0.001	50%	320k
7	2	64	0.0001	no	320k
8	2	64	0.01	no	320k
9	2	64	0.001	50%	640k
10	2	1024	0.1	no	320k
11	4	1024	0.001	50%	320k
12	4	64	0.001	50%	320k
13	4	64	0.001	50%	640k

4.1.3 TabNet

TabNet is a novel high-performance and interpretable canonical deep tabular data learning architecture. TabNet uses sequential attention to choose which features to reason from at each decision

step, enabling interpretability and more efficient learning as the learning capacity is used for the most salient features. It outperforms other neural network and decision tree variants on a wide range of non-performance-saturated tabular datasets and yields interpretable feature attributions plus insights into the global model behavior (Arik & Pfister, 2019).

Among several possible hyperparameters that could have been optimized, the following ones were selected as the ones offering the best possibility of increasing the prediction’s performance. Table 4.4 resumes the searched possibilities.

The following parameters were tested with different values:

- `n_d` - Width of the decision prediction layer. Bigger values give more capacity to the model with the risk of overfitting. The default value is 8. As the model is very complex and there is low correlation with the target variables among the features, values higher than the default 8 were tested;
- `n_a` - Width of the attention embedding for each mask. According to the paper `n_d=n_a` is usually a good choice, so at each step the value was equal to the `n_d`;
- `n_steps` - Number of steps in the architecture (usually between 3 and 10). The default value is 3. As this is a very unexplored model, a value of 7 was also tested. It would allow for a possibly better fit on the data;
- `gamma` - This is the coefficient for feature reuse in the masks. A value close to 1 will make mask selection least correlated between layers. Values range from 1.0 to 2.0. The default value is 1.3, but values of 1 and 2 were also tested because it could help with the complexity of the problem.

Table 4.4: Hyperparameters tested for TabNet.

Hyperparameters	Configurations	#Combinations
<code>n_d</code>	[8, 32, 64]	18
<code>n_a</code>	equal to <code>n_d</code>	
<code>n_steps</code>	[3, 7]	
<code>gamma</code>	[1, 1.3, 2]	

4.2 EVALUATION

In this section, the performances of the models are presented, based on the validation set. This will allow to comprehend the advantages and disadvantages of each model and pick the best one from the classical models, the deep neural networks and the best configuration of TabNet.

With the selected models from each groups it will be possible to compare the different approaches and conclude about the best one. Both the classical models and the TabNet were trained with 500k entries (320k for training), with DNN being tested against a higher amount also (1M entries, 640k for training). The DNN was tested with a higher number of entries because it doesn't comes to a plateau, as classical machine learning models do, accordingly to literature.

4.2.1 Classical Models Evaluation

In table 4.5 it is possible to observe the performances of the models trained, as well as the hyperparameters that were needed to achieve those results. Memory used and time needed to train the model were also considered because computational resources can be expensive or scarce in a company.

The best model is CatBoost, which was able to achieve an AUC-ROC of 79% with the lowest memory usage and one of the shortest time to train. Random Forest was the only estimator coming close to the CatBoost's performance, with a higher time and memory needed to train.

Table 4.5: Classical models' performances comparison on the validation set.

Model	Best Hyperparameters	Memory	Time	Prec. Pos	Rec. Pos	F1 Pos	AUC ROC
CatBoost	'depth': 10, 'iterations': 200, 'learning_rate': 0.1	5 Gb	7 min	18%	75%	30%	79%
Random Forest	'max_features': 'sqrt', 'min_samples_leaf': 5, 'n_estimators': 500	8 Gb	38 min	37%	15%	21%	78%
Logistic Regression	'C': 0.1, 'penalty': 'l2', 'solver': 'sag', 'tol': 1e-06	5 Gb	3 min	15%	72%	25%	72%
KNN	'algorithm': 'ball_tree', 'leaf_size': 5, 'n_neighbors': 5, 'weights': 'distance'	6 Gb	48 min	22%	7%	10%	63%
Naïve Bayes	-	6 Gb	1 min	16%	36%	23%	67%
SVM Linear	'C': 1, 'dual': False	6 Gb	48 min	16%	71%	25%	67%

The CatBoost model achieved a F1 score for the positive class of 30% and a AUC-ROC of 79%, with a low amount of training time. The precision-recall curve (Fig. 4.1) produced hints about the possible best decision threshold to use in this problem, in case it is selected as the best overall model.

4.2.2 Deep Learning Models Evaluation

The best results achieved are highlighted in Table 4.6, with a AUC-ROC of 73%, and an F1-score of 35% with a Precision of 23% and a Recall of 72%. When comparing with the other experimental setups it is clear that the main factor is the amount of data the model was trained with. When comparing it with the same model structure and same parameters but using only 500k (ID 12), it is possible to see an increment of 5% in the AUC-ROC, as well as increase in both Precision and Recall for the positive class.

Table 4.6: Results obtained with the different configurations for the validation set.

ID	Best Epoch	Memory Used	Time to Train	Prec. Pos	Rec. Pos	F1 Pos	AUC-ROC
1	5	18 GB	16,0 H	21%	60%	32%	69%
2	7	13 GB	16,0 H	23%	47%	31%	65%
3	7	13 GB	2,0 H	19%	55%	28%	65%
4	7	15 GB	2,5 H	20%	56%	29%	66%
5	11	16 GB	1,5 H	24%	45%	31%	65%
6	6	13 GB	25,0 H	21%	64%	31%	59%
7	5	13 GB	10,0 H	20%	58%	30%	57%
8	3	13 GB	10,0 H	20%	64%	30%	69%
9	5	22 GB	32,0 H	21%	68%	32%	70%
10	none	13 GB	0,5 H	-	-	-	-
11	3	13 GB	1,0 H	19%	67%	29%	68%
12	4	13 GB	6,0 H	21%	60%	32%	68%
13	4	20 GB	12,0 H	23%	72%	35%	73%

The majority of the models were evaluated using 500k entries. From those, 320k were used to train, 80k to validate and 100k to test. In a deployment state, the models would likely be trained with the full 3.9 million entries. Increasing the training size would quickly increase the training time, as seen in the table 4.6. If the network had to be re-trained due to a sudden event, it can take weeks, which is a lot of downtime. For this reason, it can be useful to have a model with the same characteristics as the highlighted model, but with a larger batch size to allow a faster train.

Regarding batch size, it can be seen that increasing from 64 to 1024 shortens training time because the calculation used to measure loss is less frequent. However, an increase from 1024 to 10000 shows an extension of the training time. This is due to the fact that extensive calculations are carried out for a much larger number of instances that have to be taken into account.

Increasing or decreasing the learning rate from the standard 0.001 to 0.01 or 0.0001 does not have a major impact on the training time. However, if you increase this value to 0.1, it is possible that the model train is really faster, but loses all of its predictive power.

Our previous tests showed that a decrease in the learning rate is essential to prevent the model

from being set to a local minimum. This can easily happen because it is an unbalanced problem.

4.2.3 TabNet Evaluation

The TabNet model showed the best results with the parameters described in Table 4.7. The high `n_d` and `n_a` values seem to be important, due to the problem complexity. The number of steps (`n_steps = 3`) has a direct impact on the amount of time that the model needs to train. The best hyperparameters have a low number of steps, which is helpful. The best hyperparameter configuration was able to achieve an AUC-ROC of 78%.

Table 4.7: Classical models' performances comparison on validation set.

Hyperparameters	Memory	Time	Prec. Pos	Rec. Pos	F1 Pos	AUC ROC
'n_d': 64 'n_a': 64 'n_steps': 3 'gamma': 1.0	5 Gb	4,5 H	17%	78%	28%	78%

4.2.4 Discussion

The CatBoost model was able to achieve the best performance among the different trained models, when considering the AUC-ROC score, the one that shows evidence of the separability of the classes. Comparing this model to TabNet and DNNs (Table 4.8), it is also the fastest one. This is possible because CatBoost is explicitly designed to work on heterogeneous data, while building his trees in a symmetric way. Symmetric trees help to reduce overfitting because they keep the decisions simple. Using gradient boosting, the branches of the trees are still organized in a way that allows to capture complex relations between multiple features.

On the other hand, both TabNet and DNNs are based on neural networks, which usually offer better performance on homogeneous data, like the one present in sound, image or text. In the present dataset, there are lots of categorical data columns, some with hundreds or thousands or different possible values, whose data distribution is not normally distributed. This presents a real difficulty to neural networks, while it is handled by CatBoost's core.

When comparing CatBoost to DNN (Table 4.8) in terms of memory costs, it is possible to understand that this is possible because of the way CatBoost works. Being based on trees, the conditions are built layer by layer, while DNN need to work with all the features at the same time in order to produce an output. The DNN backward propagation is also inherently more complex than adjusting the CatBoost tree's branches. The memory and time usage was measured when using 640k entries to train the models, but when using 3 million entries the resource needs would increase greatly (at

least 4.5 times more, linearly). Depending on the systems where the models would be trained, this could have a big impact. If the model was set to be trained every weekend in the same computer that was used for the tests, then the training phase would take more than 48 hours, exceeding the weekend. However, if the model is set to be trained in a more capable machine, this could be a no-problem and the extra classification performance could be achieved.

Table 4.8: Best model's performances comparison on validation set.

Model	Memory	Time	Prec. Pos	Rec. Pos	F1 Pos	AUC ROC
CatBoost	5 Gb	0,1 H	18%	75%	30%	79%
DNN	20 Gb	12,0 h	23%	72%	35%	73%
TabNet	5 Gb	4,5 h	17%	78%	28%	78%

4.3 BUSINESS EVALUATION

As the CatBoost model was the one obtaining better results on the validation set and would be used in production, it is worth analyzing the precision-recall curves (Fig. 4.1) and adapt the decision threshold. The default decision threshold for the predictions is 0.5 but accordingly to the precision-recall curve a threshold of 0.75 would be optimal. This trade-off allows to identify less customers who would probably recur, but with more confidence in the prediction, making the process more efficient for the company (Table 4.9).

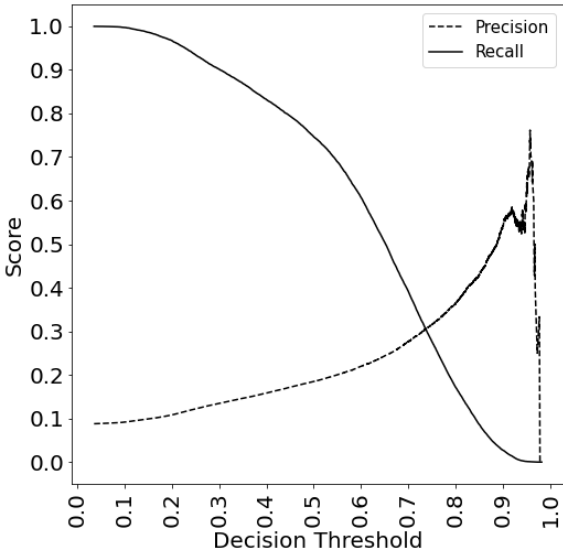


Figure 4.1: Precision and Recall Scores as a function of the decision threshold from CatBoost.

By doing so, a Recall of 61% and a Precision of 62% were achieved, with respect to the final test set. The final F1-score is 62% and the AUC-ROC is 79%. This means that from every 10

customer recurring, 6 of them would be correctly identified. Using this model in production could result in automatic measures being taken. Moreover, the service of the customers flagged could be inspected by free operators, depending on the occupation of the customer service, improving the overall quality of experience without any impact on the costs for the company.

Table 4.9: Results on the test set: default vs. custom threshold.

Decision threshold	Prec. Pos.	Rec. Pos.	F1 Pos.	AUC-ROC
0,50 (default)	18%	75%	30%	79%
0,75 (custom)	32%	28%	29%	79%

5. CONCLUSIONS

5.1 CONCLUSIONS

The telecommunications' market is highly competitive and has seen a big increase in the past years in the user base. Companies try to keep their customers satisfied so that they don't quit the service anytime later.

The evolution of machine learning and computational power, along with the data being stored with more care, gives a great opportunity to the companies to improve their business relation with the customers. Data from the customer support service, along with data from billing, service usage, and personal data, can be arranged to provide a good amount of information.

More recently, deep learning techniques have presented themselves as great substitutes of classical machine learning models, as they show great performance improvements over these classical models in several fields. The most common applications for deep learning are related with high-complexity problems, with a high correlation between the features, like sound, image, or text. However, these fields are not the only ones with high-complexity problems, therefore using deep learning in predicting customer related problems could be of great use. Recently, Google has shown great results with TabNet, a deep neural network dedicated for tabular data, the same format that's frequently used for customer related problems, showing improvements comparing to other boosting and decision trees algorithms.

In order to keep customer satisfied, a predictive model was created that is capable of identifying recurrent problems within the services contracted by the customers. This will allow for the company to act before the problems are noticed again by the customer. The available data suffered several cleanses and transformations in order to be prepared to be used in the models. Several models were tested and optimized, including classical machine learning models, deep neural networks and even novel approaches like TabNet.

The model with the best performance, CatBoost, was able to achieve a Recall of 61% and a Precision of 62% with real data. The final F1-score is 61% and the AUC-ROC is 79%, which are interesting results and show that application in the industry is possible. CatBoost is a classical machine learning approach that usually reaches a performance plateau, whereas DNNs often benefit from an increase in the data available. For that reason, however CatBoost has been selected as the best performing model, in future applications where the neural network approaches are able to perform similarly to CatBoost, it could be interesting to compare both models and maybe eliminate the plateau of classical models.

The benefits for the companies consist of a possible reduction of the churn rate, due to the customers being happier with the service, increasing the overall profits. Having the knowledge of a possible recurrence beforehand also decreases the time that customer service operators stay on the

phone, decreasing the costs for the company in the long-run. For the customer, with the application of this model in production, they will have the opportunity to experience a more fluid service and to see their problems being solved with higher efficiency.

5.2 FUTURE WORK

The work made could see some improvements on the data and on the model. More feature extraction could be done if more computing power was available. With the available resources, some of the aggregations and calculations would take too long to be calculated, decreasing the time available to the other steps of the process.

Improvements on the DNN model could also be made by using other configuration other than MLP. Including an LSTM or other structure that was able to capture patterns of problems and memorize them, could also contribute to increase the performance of the model. This would, however, increase the computational power needed, leading to investment from the company. Providing the network with a series of dedicated embedding layers could also result in a better classification and an overall improvement in performance, as they are known for reduction the feature representation memory needs (compared to One-Hot Encoding), while preserving the relations within the feature's values. This would, however, increase the time needed to study the problem feature by feature, preventing the dissertation from being completed at the necessary deadline.

On the other hand, using TabNet (based on neural networks itself) could be a good approach for higher amount of data is it becomes clear that CatBoost has reached a plateau. This would allow to eliminate the plateau through neural networks, without the need to tweak the neural network itself.

Furthermore, separating the most frequently identified problems from the others, would allow for a more specific training for each type of problem. Therefore, each one of the models would be more adapted to each one of the problematics, although that increases the overall complexity of maintaining the models in production.

Finally, in order to use this model in production, all the computational steps over the data would need to be automatic. This is a complex task, the last of CRISP-DM, but would be necessary to prove the efficacy of the solution.

REFERENCES

- Alencar, R. (2017). *Resampling strategies for imbalanced datasets*. Retrieved 2020-10-04, from <https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets>
- ANACOM. (2019). *Factos & Números - 4.º trimestre de 2018*. Retrieved 2019-09-26, from https://www.anacom.pt/streaming/Factos_Numeros4T2018.pdf?contentId=1472608&field=ATTACHED_FILE
- Arik, S. O., & Pfister, T. (2019, aug). TabNet: Attentive Interpretable Tabular Learning. *arXiv e-prints*. Retrieved from <http://arxiv.org/abs/1908.07442>
- Azlah, M. A. F., Chua, L. S., Rahmad, F. R., Abdullah, F. I., & Alwi, S. R. W. (2019, dec). *Review on techniques for plant leaf classification and recognition* (Vol. 8) (No. 4). MDPI AG. doi: 10.3390/computers8040077
- Bianchini, M., & Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks and Learning Systems*, 25(8), 1553-1565.
- Brownlee, J. (2020a). *SMOTE for Imbalanced Classification with Python*. Retrieved 2020-09-22, from <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>
- Brownlee, J. (2020b). *SMOTE for Imbalanced Classification with Python*. Retrieved 2020-10-04, from <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>
- Buchanan, B. G. (2005). A (very) brief history of artificial intelligence. *AI Magazine*, 26(4), 53–60.
- Castro, V., Pereira, C., & Alves, V. (2020, 11). Predicting recurring telecommunications customer support problems using deep learning. *21st International Conference on Intelligent Data Engineering and Automated Learning - IDEAL 2020*.
- Chepenko, D. (2019). *Introduction to gradient boosting on decision trees with Catboost*. Retrieved 2020-10-04, from <https://towardsdatascience.com/introduction-to-gradient-boosting-on-decision-trees-with-catboost-d511a9ccbd14>
- ComparaJá - NOS. (n.d.). Retrieved from <https://www.comparaJa.pt/instituicoes/nos>
- Fielding, A. H., & Fielding, A. H. (2010). *Classification accuracy*. Retrieved from <https://developers.google.com/machine-learning/crash-course/classification/accuracy> doi: 10.1017/cbo9780511607493.008
- Google Developers. (2020a). *Classification: Precision and Recall*. Retrieved from <https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall>
- Google Developers. (2020b). *Classification: ROC Curve and AUC*. Retrieved

- from <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc>
- Grossi, E., & Buscema, M. (2007, dec). *Introduction to artificial neural networks* (Vol. 19) (No. 12). doi: 10.1097/MEG.0b013e3282f198a0
- Han, J., Kamber, M., & Pei, J. (2012). *Data Mining : Concepts and Techniques : Concepts and Techniques (3rd Edition)* (3rd ed.). Morgan Kaufmann. Retrieved from <http://linkinghub.elsevier.com/retrieve/pii/B9780123814791000010> doi: 10.1016/B978-0-12-381479-1.00001-0
- Hung, S. Y., Yen, D. C., & Wang, H. Y. (2006). Applying data mining to telecom churn management. *Expert Systems with Applications*, 31(3), 515–524. doi: 10.1016/j.eswa.2005.09.080
- Hunt, V. D., & Hunt, V. D. (1986). *Introduction to Artificial Intelligence and Expert Systems*. In *Artificial intelligence & expert systems sourcebook* (pp. 1–39). Prentice Hall of India. doi: 10.1007/978-1-4613-2261-0_1
- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, 1, 448–456.
- Karsoliya, S. (2012). Approximating Number of Hidden layer neurons in Multiple Hidden Layer BPNN Architecture. *International Journal of Engineering Trends and Technology*, 3(6), 714–717.
- King, G., & Zeng, L. (2002, 09). Logistic regression in rare events data. *Political Analysis*, 9. doi: 10.1093/oxfordjournals.pan.a004868
- Kingma, D., & Ba, J. (2014, 12). Adam: A method for stochastic optimization. *International Conference on Learning Representations*.
- Kok, J. N., Boers, E. J. W., Kusters, W. A., Putten, P. V. D., & Poel, M. (2010). Knowledge for sustainable development: an insight into the Encyclopedia of life support systems: Artificial Intelligence: Definition, Trends, Techniques and Cases. In *Artificial intelligence: Definition, trends, techniques and cases* (pp. 1096–1097).
- Larivière, B., & Van Den Poel, D. (2005). Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications*, 29(2), 472–484. doi: 10.1016/j.eswa.2005.04.043
- Lemaitre, G., Nogueira, F., Oliveira, D., & Aridas, C. (2017a). *Comparison of the different over-sampling algorithms – imbalanced-learn 0.5.0 documentation*. Retrieved 2020-09-22, from http://imbalanced-learn.org/en/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html#sphx-glr-auto-examples-over-sampling-plot-comparison-over-sampling-py
- Lemaitre, G., Nogueira, F., Oliveira, D., & Aridas, C. (2017b). *Comparison of the different over-sampling algorithms – imbalanced-learn 0.5.0 documentation*. Retrieved 2020-10-10, from https://imbalanced-learn.readthedocs.io/en/stable/auto_examples/over-sampling/plot_comparison_over_sampling.html

- Mandot, P. (2017). *What is LightGBM, How to implement it? How to fine tune the parameters?* Retrieved 2020-10-04, from <https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc>
- Mayo, M. (n.d.). *Understanding the Bias-Variance Tradeoff: An Overview*. Retrieved 2020-10-04, from <https://www.kdnuggets.com/2016/08/bias-variance-tradeoff-overview.html>
- Moursund, D. (2006). Brief introduction to educational implications of Artificial Intelligence. *Artificial Intelligence*(January 2003), 1–75. Retrieved from <http://scholarsbank.uoregon.edu/jspui/handle/1794/3114>
- Nand Kumar, C. N. (2017). *Comparative Analysis of Machine Learning Algorithms for their Effectiveness in Churn Prediction in the Telecom Industry* (Vol. 4) (No. 8). Retrieved from <https://irjet.net/archives/V4/i8/IRJET-V4I887.pdf>
- Narkhede, S. (2018). *Understanding AUC - ROC Curve*. Retrieved from <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>
- Nelson, D. (2020). *Overview of Classification Methods in Python with Scikit-Learn*. Retrieved 2020-01-26, from <https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (n.d.). *Decision Tree Regression – scikit-learn 0.23.2 documentation*. Retrieved 2020-10-04, from https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2015). *Cross-validation: evaluating estimator performance*. Retrieved from http://scikit-learn.org/stable/modules/cross_validation.html
- Pina, V., Torres, L., & Bachiller, P. (2014). Service quality in utility industries: The European telecommunications sector. *Managing Service Quality*, 24(1), 2–22. doi: 10.1108/MSQ-03-2013-0034
- Ralabs. (2019). *Logistic Regression with Python - OpenDataScience.com*. Retrieved 2020-10-04, from <https://opendatascience.com/logistic-regression-with-python/>
- Ramzai, J. (2019). *Simple guide for ensemble learning methods*. Retrieved 2020-10-04, from <https://towardsdatascience.com/simple-guide-for-ensemble-learning-methods-d87cc68705a2>
- Schaverien, A. (2019). *How Retailers Can Adapt To AI. And The Future Of Shopping*. Retrieved 2019-09-26, from <https://www.forbes.com/sites/annaschaverien/2019/03/18/ey-future-of-shopping-retail-ai-artificial-intelligence/>
- Shalev-Shwartz, S., & Ben-David, S. (2013). *Understanding machine learning: From theory to algorithms* (Vol. 9781107057135). Cambridge University Press. doi: 10.1017/

CBO9781107298019

- Srivastava, T. (2018). *K Nearest Neighbor | KNN Algorithm | KNN in Python & R*. Retrieved 2020-10-04, from <https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/>
- Stathakis, D. (2009, 04). How many hidden layers and nodes? *International Journal of Remote Sensing*, 30, 2133-2147. doi: 10.1080/01431160802549278
- Uhrig, R. E. (1995). Introduction to artificial neural networks. In *Iecon proceedings (industrial electronics conference)* (Vol. 1, pp. 33–37). IEEE. Retrieved from <http://ieeexplore.ieee.org/document/483329/> doi: 10.1109/iecon.1995.483329
- Vazquez, F. (2017). *Deep Learning Made Easy with Deep Cognition*. Retrieved 2020-10-04, from <https://www.kdnuggets.com/2017/12/deep-learning-made-easy-deep-cognition.html>
- Vazquez, F. (2018). *The differences between Artificial and Biological Neural Networks*. Retrieved 2020-10-04, from <https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7>
- Welcome to LightGBM's documentation! – *LightGBM 3.0.0.99 documentation*. (n.d.). Retrieved 2020-09-22, from <https://lightgbm.readthedocs.io/en/latest/>
- Yan, L., Biosciences, A., Wolniewicz, R. H., & Computing, F. (2004). Behavior in Telecommunications. *IEEE Intelligent Systems*, 19(2), 50–58.
- Yan, L., Miller, D. J., Mozer, M. C., & Wolniewicz, R. (2001). Improving prediction of customer behavior in nonstationary environments. *Proceedings of the International Joint Conference on Neural Networks*, 3, 2258–2263. doi: 10.1109/ijcnn.2001.938518