NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

PEDRO MIGUEL SOARES CAPITÃO

Undergraduate in Electrical and Computer Engineering

MULTI-TASK NEURAL NETWORK MODEL FOR CUSTOMER SERVICE EXPERIENCE PREDICTION

ANALYSIS BASED IN QOS AND QOE DATA, OVER TELECOMMUNICATIONS INTERNET AND TV SERVICES

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
November, 2021

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

MULTI-TASK NEURAL NETWORK MODEL FOR CUSTOMER SERVICE EXPERIENCE PREDICTION

ANALYSIS BASED IN QOS AND QOE DATA, OVER TELECOMMUNICATIONS INTERNET AND TV SERVICES

PEDRO MIGUEL SOARES CAPITÃO

Undergraduate in Electrical and Computer Engineering

Advisers: Rodolfo Oliveira
Associate Professor with Habilitation, NOVA University of Lisbon
Susana Brandão
Lead Data Scientist, NOS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
November, 2021

Multi-task neural network model for Customer service experience prediction

To my late grandmother who left our family but left the biggest legacy anyone could ask, her person her will and her love. Forever my accomplishments are yours.

Acknowledgements

With the thesis a cycle in life finishes so that another can flourish. This thesis reflects my character and the evolution I had during the university journey. The process was never lonely, as I had immense support from my family, my friends and professors. Every one contributed with a hug, a moment of happiness, hard lessons, and challenges. To my family a special gratitude has from the day I was born gave me the conditions to get through the university but more importantly to succeed in life and be prepared to face the challenges head up with utmost confidence. To my girlfriend that gave a spark to my life, through love and challenges paved the way for confidence and dreams.

Special appreciation to Susana Brandão my adviser and colleague, who welcomed me into the company. Thank you for trusting in my capacity to delivery and giving me this opportunity that forever will shape my person. Big sentiment of gratitude to my professor and adviser Rodolfo Oliveira by always being present with desire to contribute and help.

Finally I would like to leave a note to acknowledge the life changing experiences abroad, without them the bright path ahead would not open before my eyes.

"You cannot teach a man anything; you can only help him discover it in himself." (Galileo)

Abstract

Telecommunication companies monitor the infrastructure and equipment to guarantee the quality delivery of their services to the customer. Customer experience is of utmost importance in the telecommunication area due to the fierce market competition. There are two main reasons for a client to experience bad quality of service: disruption events and service degradation. Disruptive events map to situations where the client loses complete access to the service, whereas service degradation limits the client's service and/or can origin multiple failures during the service use. In disruption situations, the client will immediately call and make a complaint. In degradation the client may not complain due to many different reasons, but mainly because the client does not use the service or does not want to go through the process of calling the operator to place a complaint.

This dissertation presents a solution that can identify customers with degradation in their services so that the company can proactively call the signaled customers and schedule an intervention to correct the problems. Currently is suspected that the intersection between the services is higher than is perceived, so it is the thesis objective to explore and verify services correlation. To achieve the goal we start by characterizing the problem, then explore the real data consumption patterns, and analyze the intersection in experience problems. To cope with TV and Internet services complains target simultaneously the thesis leverages a model with a multi-task architecture whose goal is to learn the implicit information available in the services intersection. Multi-task model solves the limitations existent in current solutions and adds the possibility to have multiple targets that share information between them. Then, the model develops the capacity to learn situations where the symptom and the resolution are in different services.

Through the multi-task model capacity to explore the implicit information contained in the services intersection the results are improved 25% against the benchmark models which do not support multi-task. Given the performance increase, the company project strategy was impacted and new uses cases considered.

Keywords: Telecommunications, Big data, QoS, QoE, Multi-task learning, Neural networks, Lift evaluation metric

Resumo

As empresas de telecomunicações monitorizam a infraestrutura e os equipamentos para conseguirem garantir qualidade na entrega dos serviços aos clientes. Num mercado feroz como o de telecomunicações, investir na experiência de utilizador é muito importante. As duas razões principais para explicar má experiencia de serviço são: eventos disruptivos e degradação no serviço. Nos eventos disruptivos o cliente perde totalmente o acesso ao serviço, enquanto que para problemas de degradação o serviço fica limitado e/ou pode falhar durante a utilização do mesmo. Em situações disruptivas, o cliente contacta immediatamente a empresa para apreentar uma queixa. Já nos casos de degradação o cliente pode não se queixar. Entre muitas razões possiveis sobressaiem as seguintes, o cliente não usa o serviço e o cliente prefere não passar pelo processo duro de contactar o operador para se queixar do mau serviço.

A dissertação implementa uma solução que identifica clientes com degradação no serviço, para que a empresa possa ter uma atitude proactiva em relação aos clientes sinalizados, e consiga marcar intervenções técnicas para resolver o problema. A tese tem como objectivo validar que a interseção entre os serviços de TV e Internet é maior do que atualmente reconhecido, e que a interseção produz informação que deve ser explorada. No primeiro exercicio caracterizamos do problema, exploramos os dados reais de padrões de consumo, e é analisada a situação de interseção entre os serviços. Desenvolvemos o modelo segundo um arquitetura multi-task que permite a partilha de informação e portanto o modelo aprende a informação implícita que existe na interseção do serviços. O modelo multi-task corrige as limitações existentes nos modelos atuais, e permite a integração de outros problemas que a empresa venha a sentir necessidade de resolver.

Os resultados obtidos quando comparados com os modelos de benchmark cuja arquitetura não suporta multi-task mostram um aumento de performance acima dos 25%. Devido aos resultados obtidos a estratégia para o projeto foi revista e novas ideias de projetos surgiram.

Palavras-chave: Telecomunicações, Big data, QoS, QoE, Multi-task, Redes neuronais, Metrica de avaliação Lift

Contents

List of Figures

List of Tables

List of Listings

Acronyms

AI Artificial Intelligence 10, 11, 12

BER Bit Error Rate 31, 32

CER Code Error Rate 25, 31, 32, 33
CM Cable Modem 6, 7
CMTS Cable Modem Termination System 6, 7
CPU Central Processing Unit 26, 27

DOCSIS Data Over Cable Service Interface Specification 72

FTTH Fiber-to-the-Home 8

HFC Hybrid Fiber Coax 6, 8, 39, 46

IoT Internet of Things 5, 10
ITU International Telecommunication Union 5, 9, 10

ML Machine Learning 10
MSE Mean Square Error 50

QoE Quality of Experience 1, 5, 6, 11, 12, 35, 40, 50
QoS Quality of Service 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 25, 27, 29, 70, 75

SNR Signal-Noise Ratio 31

Tx Transmitted 30

Up Upstream 30

1

Introduction

Telecommunications services go back a while and telecommunication companies always have implemented differentiation of service delivered to the customer. Guarantee service quality increases revenue, clients are more open to paying for services and increases the popularity of the company in the market due to having the best technology and providing a good user experience. From the legacy implementations to the IP world providing quality of service is a requirement, thus forcing technology to evolve both in hardware and software. Telco distribution networks are growing, newer mechanisms and algorithms for routing, bandwidth allocation, battery consumption efficiency, and prediction of node allocation during service among many other fields are in continuous learning. Customers also get in their houses' stronger and more sophisticated equipment to better suit their needs. All sources combined to create a massive flux of data which a telecommunication company needs to process to obtain knowledge to understand if the delivered service to the customer is up to standard. Guaranteeing all of this enables customer satisfaction and company prosperity.

Concept acquired, putting into practice is more complex, hence only architectures specially developed to cope with so much data (e.g. BigData) and algorithms capable to search for valuable insights through many features and instances (e.g. Artificial Intelligence, Machine Learning) can help solve such problems. Exploring these resources enables us to create models that can learn the intrinsic dynamics of a telco network. One model can map the problems and do a root cause analysis, focusing on the knowledge discovery about the network. Another approach can be supported by a model that takes all systems into account, in other words, learn about the telecom network infrastructure and its customers (e.g. preferences, profile, technical complaints, and others). Quality of Experience (QoE) is a concept that appears through the necessity of mapping the client's pain and is a concept that immensely benefits from the vast data available nowadays.

1.1 Motivation

Customer satisfaction is an essential for telecommunication companies. Therefore guaranteeing an optimal delivery of services, both TV and Internet, allows a healthier business prospect since happy customers continue to pay and can also be targeted for new products and upgrades on their contracts.

A major player in Portuguese telecommunications service provider (NOS) is a partner in the development of the thesis. The challenge was proposed by NOS, with the objective of taking advantage of the vast data gathered from the service infrastructure measurements and metrics. Available data contains valuable information regarding the quality of service. Specifically for Internet and TV services, it is important to understand failures in the network in order to fix them and guarantee a high quality of experience.

The current situation is that NOS has models based on gradient boosting trees algorithm for the Internet service and models for the TV service. The challenge is to simplify the system architecture. The current solution in production is composed of multiple models from both Internet and TV services. Hence, the NOS development department faces the difficult task of evolving and maintaining all models. Every time the infrastructure suffers updates or the clients' consumption pattern alters the team has to explore the new distributions for the input variables and retrain the models, test, and deploy in production which is very costly and can be improved. To address this problem we propose a solution able to merge all models into a single model capable of performing the same functions.

In the actual solution, the TV and Internet client pain score function is separate, thus indicating the services are independent. One of the first steps is analyzing the system and validating that indeed the intersection between services is higher than expected. Through the analysis, we lay the foundation for our project. Current models do not consider the other service information. The evolution proposed is to create a model capable of learning all the implicit information existent in the services intersection. On top of the inference process, it is a root-cause analysis module. The root-cause analysis module is very important for the project operation. Through it, we can explain why the model is signaling a customer with a probability to complain about the service.

Root-cause analysis is an important feature for the system, and is a field that absorbs interest and investigation. In the work presented in [5] the authors perform anomaly detection to identify failures in the system leveraging the use of principal component analysis to identify low detection latency problems, and on top perform system root-cause analysis in 4G networks to improve reliability and availability. In the same context, the authors in [2] study a methodology to locate the problem source of a sequential network in which the data is in time series format. They aim to prove a solution for the same problem, root-cause analysis, however at the core of the developed analysis is a characterization of convergence rate for log-likelihood ratios.

A key of particular inference in this thesis, is to validate that the intersection between

the services is higher than currently acknowledged, and to create a solution that can leverage the intersection information into valuable insights.

1.1.1 Objectives and Contributions

The thesis is ambitious and we want to achieve an impactful contribution in the company's operation, and simultaneously develop a novel solution in the telecommunication field that proves to work in a real project with real data.

Objectives

The main goal of this project is to extract valuable insights from NOS QoS big data repository through the design, development, and evaluation of a multi-task neural network capable of capturing both Internet and TV service dynamics.

- **Objective 1.** The first objective is to prove that the intersection between the Internet and TV services is higher than what is currently acknowledged. The development of the task serves two purposes: exploring the project environment and context, and validating that the thesis mission indeed makes sense. In environment exploration, the goal is to understand the project, the tools we are going to be working with, and understand the data available for the thesis development. Secondly, we need to analyze the data, and validate that indeed there is more intersection between the services than the one current models explore.

- **Objective 2.** To deploy an end-to-end inference solution, we need to accomplish multiple sub-tasks. The second objective is to create a framework that implements all the necessary steps. From defining the training and testing dataset to evaluating the model, we need to be sure every component is working correctly.

- **Objective 3.** Finally, we want to prove that the proposed multi-task neural network improves the project capacity to capture service intersection dynamics. The objective is to develop a successful multi-task architecture that mitigates the limitations felt in the project.

Contributions

For the company, we want to leave our mark by validating the intersection between services. To do so, we need to create a solution that takes advantage of the services intersection information and improves the project performance. Completing the objective towards the company, we also contribute with a novel concept of solution in the telecommunication community. We have the goal to bring a mostly found in computer vision method to resolve a limitation currently encountered.

- **Contribution 1.** Analyze TV and Internet services QoS variables distributions and correlation with the complaint target to prove services intersection exists and is relevant. The analysis verifies that services are not independent and there is implicit information to be learned when capturing services dynamics.
- **Contribution 2.** Develop a multi-task architecture able to train multiple targets, hence multiple services, simultaneously which can explore the implicit information existent in the service intersection. The proposed model mitigates the limitations of current solutions because it allows the simultaneous training of multiple services.

1.2 Report Structure

This thesis is divided into multiple sections, each one with the objective of describing knowledge and results gathered. The introduction chapter develops the thesis premise and environment, describing problems felt and the need to address them. We state our main tasks to accomplish the proposed solution, and what are our expected contribution to the company. In Chapter 2, we present a state-of-the-art on telecommunications systems infrastructure and neural network architectures.

The following chapters, describe the thesis's practical development and achieved results. First, in Chapter 3, it is introduced the project context inside the company and explored why the problem addressed in the thesis can improve the overall project operation. After supporting the thesis development through exploratory analysis, we create an end-to-end inference process framework capable of implementing a model. In Chapter 4, the proposed architectures are detailed, describing how we created processes to manage the big data environment. Evaluation performance is discussed and analyzed in Chapter 5, where we present the results obtained and how they support the thesis achievements.

In the concluding chapter, we summarize the entire project, analyzing accomplished developments and results in comparison with the initial expectations, and wrap up with challenges felt during the project. For future projects, we leave some open ideas of features to be added on top of the thesis that can improve results with a more complex and solid approach.

2

Related Work

2.1 QoS and QoE [12]

Since the commercialization of telecommunications services, companies have invested time, resources, and money towards the quality deliver of their services to the customer.

The technological evolution brought powerful computers and the rise of the Internet. Telecom operators have high and well-defined standards for end-to-end QoS for legacy services such as telephone and TV. Hence, they faced the challenge of replicating those same QoS guarantees. Although the same QoS goals are applied to these new services, they require new, different solutions to deliver the same guarantees to the customers.

The Internet was created based on the best-effort principle. However, engineers rapidly identified the need for service differentiation (e.g. a user will detect more often and be annoyed much easier if a YouTube video is fetching multiple times waiting for the arrival of data packets than if an email from a co-worker only arrived with five minutes delay). Not only in traditional communication systems but also in new areas such as (IoT), where the perception of QoS would be even more unpredictable.

ITU provided a definition for Quality of service, which is applicable for all the telecom world [9]: QoS is the totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service.

Furthermore, it is important to take into account several features identified in figure 2.1, which forms a global view about the challenges for QoS, those being related to technical or non-technical matters or related to customers' expectations.

More recently, around 2007, another very interesting concept emerged to the telecom world related to the quality of experience (QoE). This idea is an abstraction from QoS, and implies a much more human view on the subject because its analysis focuses on the subjectively perceived satisfaction from the end-user.

Users' expectation, need for the service, utilization of the service and their belief in the company ability for delivering the service are some of the main issues when talking about and analyzing an end-user perception of QoE and thus determining the level of happiness and loyalty to the services and the service provider. Once a customer is lost may never

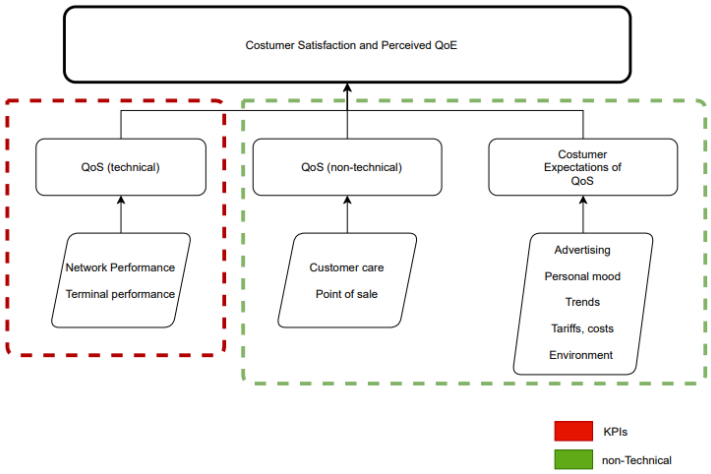


Figure 2.1: Global overview of QoS (adapted from [12]).

return. Even deeper, factors such as cultural background, user emotional state, socioeconomic profile, character, psychological profile among others influence users' perception of service's QoE.

Utilizing the words of the regulations, ITU's definition of QoE is as follows [10]: QoE is the degree of delight or annoyance of the user of an application or service.

2.1.1 Fixed Ultra-broadband

In the telecom world, it is possible to divide the communication networks in two parts: fixed and mobile. Each has its own characteristics and supports its services, nevertheless, they can also share solutions and resources. Since the practical part of this thesis will focus only on the fixed part of the network so does this chapter.

From the heritage of telephony companies, copper networks paved the way to the insurgence of DSL and Cable Access technologies, two of the three solutions in use for fixed networks. The other one, optical fiber, is the base for the transport network due to its capacity to carry bigger loads of data (e.g. higher bitrates). Fiber gets most of the research and investment and aims to replace the copper last mile. However, it is yet not possible to do so and both DSL and Cable Access are still part of the access network connecting the last mile. This happens because sometimes it is impossible or it is not practical to install fiber up to the user's end-equipment.

Table 2.1 shows examples of these technologies.

Hybrid optical-coaxial (HFC)

HFC is a popular implementation of a mixture of coaxial cable access, for the last mile, and optical fiber. HFC systems are mainly formed by the Cable Modem Termination System (CMTS) and the Cable Modem (CM). Both have the job of closing the connection to the optical fiber network. HFC implements mechanisms of QoS on the MAC and upper

Table 2.1: Fixed broadband networks and bitrates (adapted from [12]).

	Technology	Downlink bitrate	Uplink bitrate	Transmission
DSL	VDSL2	100-300 Mbit/s	100-300 Mbit/s	-
Cable	DOCSIS 3.1 full duplex	10 Gbit/s	10 Gbit/s	-
Fiber	GPON	2.5 Gbit/s	2.5 Gbit/s	Ethernet, ATM, TDM
Fiber	NG-PON	40 Gbit/s	40 Gbit/s	TDM, TWDM
Fiber	10G-PON	10 Gbit/s	10 Gbit/s	Ethernet

layers of OSI protocol in all supported services, then improves IP-based services such as VoIP (Voice over IP), VOD (Video on demand), and HDTV (Interactive High Definition TV).

The main mechanism to provide QoS starts by managing the packets classifying each one, then CMTS and CM provide QoS traffic specifications after executing operations of shaping, policing, and prioritizing traffic before delivering them to a unidirectional flow of packets known as Service flow, whose flow is split in multiple sectors for receiving each packet into their assigned group of QoS [15]. There is a differentiation between upstream service scheduling and downstream service scheduling. The first allows five different types of groups as follows:

- **Best-effort.** Standart Internet principle, which is controlled by CMTS but follows the first come first serve (FCFS) rule. CMTS queues CM on a fixed time interval for transmission within service flow, basically reserves a place for each CM to transmit
- **Non-real time polling.** CMTS queues CM on a fixed time interval for transmission within service flow, basically reserves a place for each CM to transmit
- **Real-time polling.** Good for variable bitrate traffic due to being delay and throughput tolerant, also uses a queue mechanism as the previous one but the main difference is in the time interval duration which in this case is shorter.
- **Unsolicited grant.** As the two examples before this scheduling type is reservation-based and allocates fixed-size data capacity in an approximately fixed time interval. Used in voice traffic.
- **Unsolicited grant with activity detection.** Reservation-based that implements a hybrid system between polling and unsolicited grant, this means CM, when comes active, must request polling from CMTS and then is provided with an almost fixed time interval slot where that CM can upstream its data up to a granted fixed-size packet. While CM has data to upload both data and time reservations stay on, however when CM goes inactive loses the slots, which implies the redo of this cycle every time it activates.

Downstream service flows only allow one form of scheduling which is Best-effort, with the same QoS specifications described above for upstream traffic.

Optical Fiber (FTTH)

The main protocol for fixed services is the fiber as discussed before. Normally denoted as Fiber-to-the-Home (FTTH), optical implementations could also be named as FTTP (fiber to the premises), FTTD (fiber to the desk) whereupon both integrate FTTH filter which means the customer as fiber all the way to his house divisions. FTTH matches the position of the optical network termination (ONT) which are the final links in this type of network. In solutions such as HFC, the further that fiber could be placed is 0.25 km from the user's house and is denominated FTTB (fiber to the building) or FTTC (fiber to the curb/cabinet) that share the same place as optical network units (ONU). These ONU could act as an intermediary system between optical line termination (OLT) and ONT or in this case of FTTB/C can be used as final links in the network so being a termination node. Lastly, there is FTTCab that also shares ONUs as a place to cohabit and has the specification which determines that the furthest a fiber can be found is 1.5 km from the user's equipment.

For FTTx, where x denotes all the solutions aforementioned, the access networks could be divided into four architecture categories: Point-to-point (P2P), Passive optical network (PON), Active optical network (AON), Wavelength-division multiplexing (WDM) PON. PON architectures were selected to be thoroughly studied due to being the main trend in FTTH excelling in long transmission distance, high-bandwidth, cost-efficient, and service convergence.

Figure 2.2 illustrates all of these concepts into an understandable representation of an optical fiber network with PON architecture.

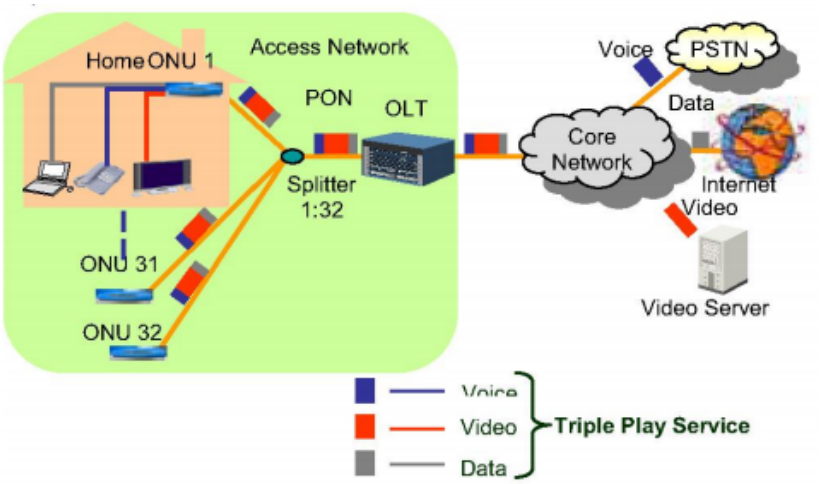


Figure 2.2: Optical fiber network components with enabling service differentiation (source [18])

A PON architecture is supported by a point-to-multi-points (P2MP) design which starts in an OLT and then is connected through a passive optical splitter to ONUs or ONTs depending if the service is FTTB/C or FTTH, respectively. For downstream transmission,

the OLT broadcasts all packets all in the end line ONUs or ONTs choose which ones to belong to themselves. For upstream each ONU or ONT has their time slot for data packet transmission since the upstream channel is shared between them all, this way the system avoids collisions [18].

GPON and XGPON are an extension of the PON architecture, and so uses only one channel (e.g. a single wavelength) which is available to a group of users (e.g. ONUs or ONTs) and through multiplexing in the time domain implements a mechanism to identify the receiver blocking all the others from the data when downstream. For upstream allocates a slot for each end-user to transmit [20]. Regarding GPON QoS the solution relies on two different algorithms, Dynamic Bandwidth Allocation (DBA) and GPON Encapsulation Mode (GEM), and both allow different levels of traffic priority.

Through the management of available bandwidth and traffic, DBA provides flexible support to multiple services guaranteeing QoS ranging from best-effort until the max (e.g. voice) hence creating a level of priority scheme ruled by the SLA (Service Level Agreement) between the service provider and service user. Bandwidth is allocated cyclically which improves the management of this resource thus allowing better utilization of the available bandwidth. A DBA function dynamically distributes different categories between all ONTs of a network and inside each ONT also its traffic flows are categorized thus improving QoS [8].

As studied in [8], for the price of 5 bytes added to the packet header, GEM allows service differentiation with stated 94% efficiency downstream and 93% efficiency upstream. GEM is responsible for GPON capacity to improve flexibility and offer more bandwidth. This is achieved through packet classification that are queued into one of the eight traffic flow physical ports thus enabling service differentiation.

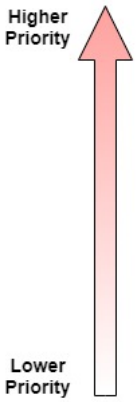
In situations where bitrate is high and the service area is densely populated with users, there is another architecture in play, NG-PON2 (NextGeneration PON). This system, as represented in table 2.1, can provide high through-put of data transfer. NG-PON uses multiple XGPON systems (normally 4), aggregating them through multiplexing in the wavelength domain which allows the architecture to have higher bitrate and unbundling that also allows each operator to work on a separate wavelength. Optical amplifiers are inserted into OLTs to increase downstream signals and preamplify the upstream connections [20].

2.1.2 QoS Classes

Through this section, it is noticeable that service differentiation is crucial for QoS schemes in telecommunications networks and more precisely in fixed ultra-broadband networks discussed in detail earlier. In table 2.2, all classes defined by ITU are shown alongside some of their specifications, real-world usage, applications, and examples.

When shaping, policing, and prioritizing traffic these are the rules that must be followed to achieve the best service possible. It is in this table that differentiated services

Table 2.2: ITU QoS Classes, types of traffic and applications (adapted from [12]).



QoS Class	Upper bound on IPTD (milliseconds)	Upper bound on IPDV (milliseconds)	Upper bound on IPLR	Upper bound on IPLR	Type of Traffic	Applications
Class 0	100	50	10^-3	10^-4	Real-Time	VoIP, Video telephony, Gamming, Voice
Class 1	400	50	10^-3	10^-4	Real-Time	VoIP, Video telephony
Class 2	100	Unspecified	10^-3	10^-4	Transaction Data	Signaling traffic
Class 3	400	Unspecified	10^-3	10^-4	Transaction Data	Interactive applications
Class 4	1000	Unspecified	10^-3	10^-4	Short Transactions	Video streaming, bulk data
Class 5	Unspecified	Unspecified	Unspecified	Unspecified	Unspecified	Best-effort Internet applications
Class 6	100	50	10^-5	10^-6	Not Yet Specified	Emerging applications
Class 7	400	50	10^-5	10^-6	Not Yet Specified	Emerging applications

start to gain shape.

2.1.3 IoT, Big Data and AI

Both of the previous concepts are ever more important with the insurgence of Internet of Things (IoT) technologies and services. In this era and for the foreseen future demand for devices connected and services running onto the Internet is exploding, consumers are buying more and more IoT equipment and services as reported by [16] whose studies show an increasing investment into the IoT solutions market generating a potential economic value of approximately 1000 billion US dollars from the year 2020 to the year 2025 (via McKinsey). According to ITU definition [11], IoT is a global infrastructure for the information society enabling advanced services by interconnection of different physical and virtual things, based on ICTs.

Additionally, users are spoiled with yearly upgrades in ultra-broadband speeds and availability on their equipment which in conjunction with cloud computing provide almost infinite possibilities for content creation. Due to this phenomenon, the volume of data available for processing, mining, and analysis is enormous creating data sets commonly denominated as Big Data, which implies great complexity when trying to organize and extract meaningful information. Having access to so much information (e.g. Big Data) requires machine power to process it, thus the need for Artificial Intelligence (AI) and Machine Learning (ML) and associated algorithms which when used open the possibility for exquisite information findings. This process is normally addressed as knowledge discovery.

QoS can be measured by key performance indicators (KPIs), the main component

of the technical factors, as by non-technical factors and the customer's expectation. Big data provides so much information that there is a need to call upon AI to analyze all of these parameters and factors aiming for discovering much-needed information. As stated before, QoE is very hard to quantify due to the singularity of each user character and overall lifestyle, but in this process, QoE leaves as a winner. Through knowledge discovery on QoS data, it is possible using AI to predict/estimate end-user perception of satisfaction with the delivered service, hence improving the knowledge about end-user experienced QoE.

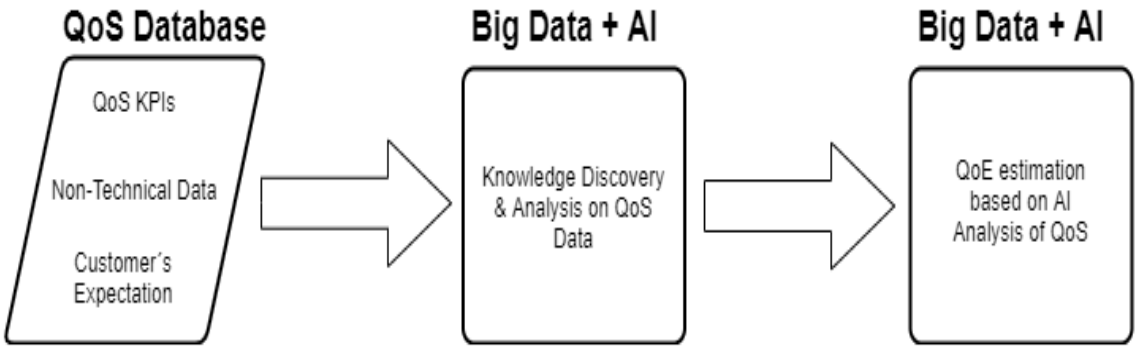


Figure 2.3: Knowledge Discovery on QoS and Estimation of QoE (adapted from [12]).

Consequently, the investment in AI methods and algorithms that aim to improve the end-user experience is a concern for telecom companies but also an area of telecommunication development that can improve not only existent algorithms but also deliver new knowledge about end-user usage patterns and satisfaction with the overall network performance of a telecom system.

2.1.4 QoS and QoE Estimation for IP-based Services

Although the thesis objectives also contemplate algorithm development to analyze and extract information from NOS telecommunication services data, the need to study and create such solutions only emerges as a vehicle to improve customers QoS and perception of QoE. The knowledge appended in the sections above is of at most importance, since they contain the information to understand the problem to be dealt with and also provide insight on essential concepts for the solution of data science projects but also have a technical view on the infrastructures and mechanisms that support the service provided to the customers. Assessment of QoS and QoE will be done at the global level through customer satisfaction regardless of the service, and at the service level where each client service will be factored into the model thus creating a dependency between the received service and the satisfaction of the client. It is naive to think that the triple-play services (Voice, Video, and Data) physical infrastructure is independent and that infrastructure is independent of all service levels aforementioned. Hence, the only way to converge all

problems felt in each independent service into one model is through the help of AI and Big Data algorithms. Such solutions enable us to go over the huge amount of data needed and create a hierarchical architecture that can connect all of the links given back useful meaning. An example of an approach involving this type of algorithmic architecture to tackle this problem is given in the article [13]. The authors objective is to comprehend human visual perception of video quality and distortions in TV video or computer video. The work explores QoS video parameters such as bitrate, resolution, codec among others and through a neural network application explore users interpretation of their service giving as an output a score of satisfaction. That information turns into useful insights that can help understand the client's perception for QoE.

2.2 Neural Networks

A neural network is built around the concept of neuron, and more precisely the agglomeration of multiple neurons. This chapter will introduce state of the art architectures models of neural networks that are thought to be very helpful with the posterior practical work. However, the concept of neuron and neural network and their conceptual features will be briefly introduced.

As illustrated in figure 2.4, the neuron supports three main ideas: Weights, Bias and Activation Function. The objective behind the neuron is that given N inputs it outputs a single value. Bias is a value to aid in the tuning of the activation function. The multiple weights are stored in a vector and are real values. They define how much the neuron has to take in consideration each input. Activation functions can be linear and nonlinear. There are multiple choices available in literature, each architecture has its preferred one. To compute the output value the neuron needs to have an activation function, which will take as argument the summation of all inputs multiplied by their weights and added to the Bias.

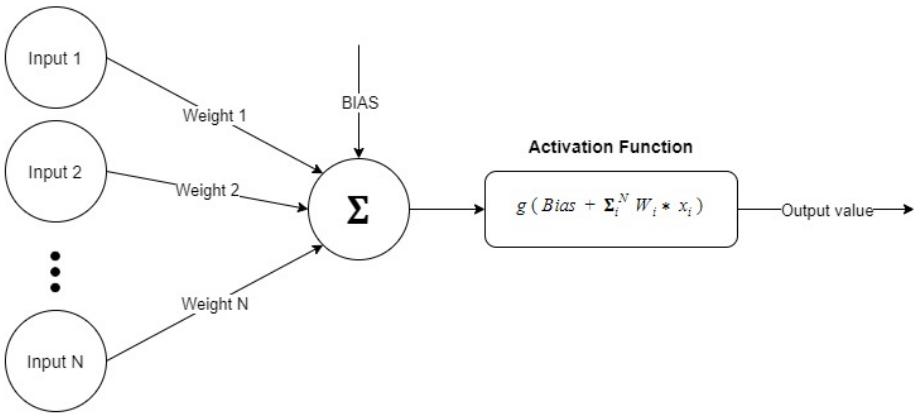


Figure 2.4: Visual concept of a single neuron.

Neural networks are a group of neurons organized into layers. Usually there is the

input layer that has as many neurons as features to be considered. This layer then feeds the hidden layer which can be constituted by one single layer or multiple layers, hence deep learning. In this part is where the training and modelling happens, meaning weights are adjusted to learn the dynamics of a problem or system. Finally, the output layer gives back the predicted answer for the input example.

From the output value given by the network it is possible to compute how wrong was the network in its predictions through a loss function that sums all the individual examples losses. The loss value is given by the difference between the real example value, thus supervised learning, and the one predicted by the network.

To increase network performance, the weights throughout the network, each connection, must be adjusted. The loss is simply a function of the network weights. A way to decrease the loss of the network is through gradient descent method which is nothing more than computing the partial derivative of a loss function with argument the weights divided by the partial derivative of those weights.

Gradient descent solution is a continuous investigation because there are still problems, such as being stuck at local minimums and expensive to impossible computation in large networks.

To absorb and process QoS data, neural networks are the most prolific option, because they allow the flexibility to merge architectures and enable the introduction of constraints in the learning phase to best suit our necessities. In the further subsections, multiple architectures are capable of addressing our needs of correlating QoS data from both services with the complain target variable. A sub-concept of machine learning which may prove very useful, multi-task learning, will be presented in more detail.

2.2.1 Convolutional Neural Networks (CNN)

Inspired by the visual senses of humans and animals, convolutional neural networks appeared in the machine learning scene to provide a reliable way of performing object recognition, text detection and recognition, scene labeling among other applications. In a sense ConvNets, how it can be also called, help with identification of actions or objects in images just as our human eye would perform them.

ConvNets architectures are constructed in a very unique way, following a sequence of steps before outputting the prediction value. Figure 2.5 represents a full convolutional neural network and its components. Every one of those is after explained in detail.

The image representation here functions as the input layer; this means each pixel of image corresponds to a neuron. Next information is based on the works of [1] and [3].

Convolutional Layer

As the name suggests, this layer is the heart of the network. Is here that through a filter/kernel it is possible to extract features from the image. Each neuron in the convolutional

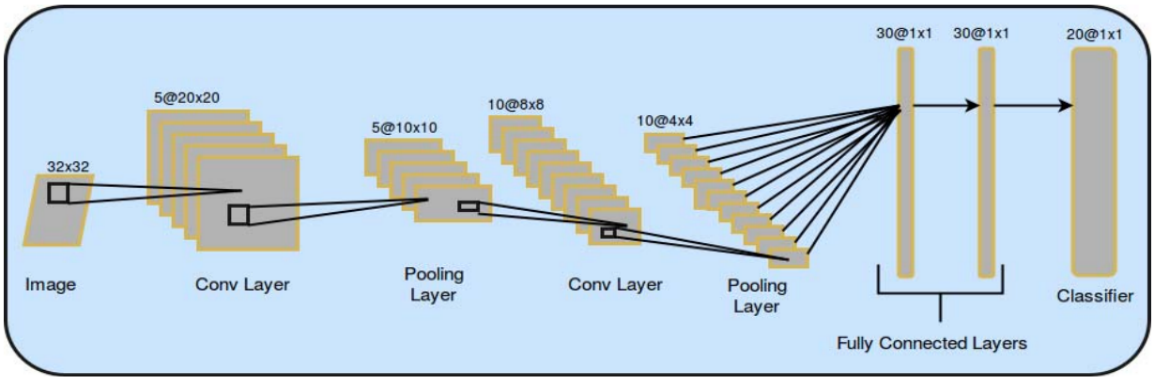


Figure 2.5: Convolutional Neural Network (ConvNet) (source [1]).

layer is connected to a hyperparameter called receptive field, which corresponds to a group of neurons in the previous layer. This receptive field has the size of the filter.

The convolutional layer has three dimensions' width, height and depth. Width and height are calculated by the composition of these factors: width and height of both input and filter, stride (e.g. distance to shift after each convolution) and zero-padding (e.g. zeros added to the borders). The depth of the layer is proportional to the number of filters used.

These filters are updatable which means that in the learning process the objective is to tune them to obtain the best possible performance. Each layer is assigned a filter and all of the neurons in the layer share that filter, method called parameter sharing.

A map from the input is multiplied dot wise with the filter resulting in a single value. This value will be the argument of the activation function, that in ConvNets normally is ReLU (Rectified Linear Unit). This function takes all negative values and assigns them zero, and all the positive values keep their original value.

Polling Layer

A polling layer usually follows a convolutional layer and its main purposes are to reduce the spatial dimension of the problem thus reducing computing complexity and also helps with overfitting phenomena. This process is accomplished by aggregating a group of neurons in the convolutional layer and applying one of the following pooling rules: max-pooling, average pooling, stochastic pooling, spatial pyramid pooling, and the list continues.

Once again the output is given by a neuron where its activation function receives as parameter the result of the pooling operation.

Fully Connected Layer

In this point is where the layers stop being three dimensional and are converted to an array through the process of flattening. Is common to add fully connected layers at the

end of a convolutional operation to introduce non-linearity. Fully connected layers means that each of the neurons present in a layer are connected to all of the neurons who provide the input. Normally in the last fully connected layer researchers use softmax in order to provide an output prediction.

The learning of Convolutional networks is done through backpropagation, a method that allows to increase the performance of gradient descent and so minimizes the value of the loss function improving the algorithm.

2.2.2 Autoencoder

Autoencoders are a type of unsupervised neural network deep learning algorithm. For an overview, autoencoders are trained to learn the most important parameters from the unlabelled input data and find low-rank representations of a dataset, meaning they find a reduced set of vectors that can capture the information given by the larger set of input vectors. With that, they train the model to output a vector as much equal to the input vector as possible.

Autoencoders architecture is based on three sections. The first one is the encoding part that takes the input data and through its multiple layers learns a compressed vector. After, there is the latent space representation that contains the compressed neuron layer that learns the compressed representation of the input data. Lastly, the decoder layer takes the output of the latent space representation layer and tries to reconstruct the information into the input data again. The learning process is through the management of the number of neurons on each layer. The number of neurons starts at maximum in the first layer of the network and is gradually reduced until reaching the latent space representation layer. From there the number grows again until the last layer, the output layer [4], [23]. Figure 2.6 illustrates a basic autoencoder architecture.

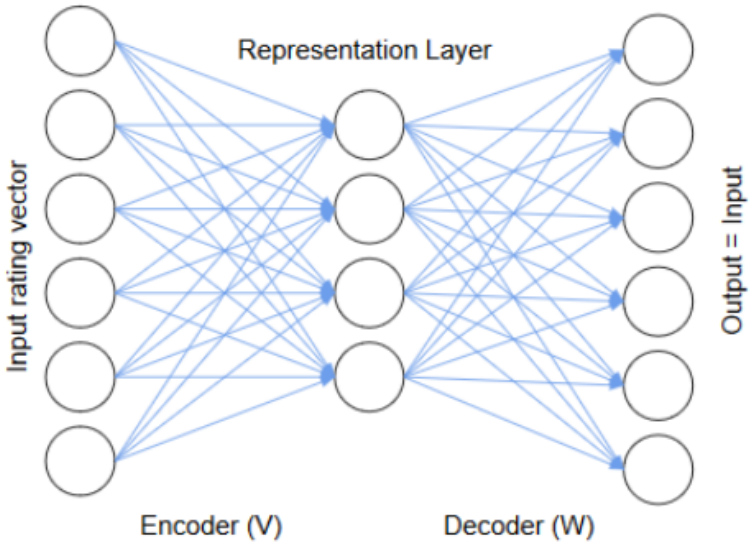


Figure 2.6: Autoencoder architecture (source [23]).

Although autoencoders are relevant in many fields such as medicine, bioinformatics, music and others they are specially relevant for this thesis be-cause of the capacity of detecting anomalies. To identify potential problems in the network signals an autoencoder could be used to catch signals that when reconstructed do not match the input, and so the software can detect a failure in the system.

2.2.3 Recurrent Neural Networks (RNN)

This type of neural network architecture is studied due to the necessity of processing sequences forcing the new states of the network to take into account the previous or even a chain of previously predicted states. Some applications of recurrent neural networks can be found in fields such as natural language processing (NLP), music, translation and speech recognition and others. In some of these areas one can immediately think that only taking into consideration the previous state may be insufficient, thus Long Short-term memory (LSTM) networks come into play.

The basic component of RNN is a cell that receives an input vector, computes an hidden state that is stored within the cell and gives an output based on the multiplication of a weight and the hidden state value of the cell. To further understand this cell system figure 2.7 displays a single recurrent cell procedure.

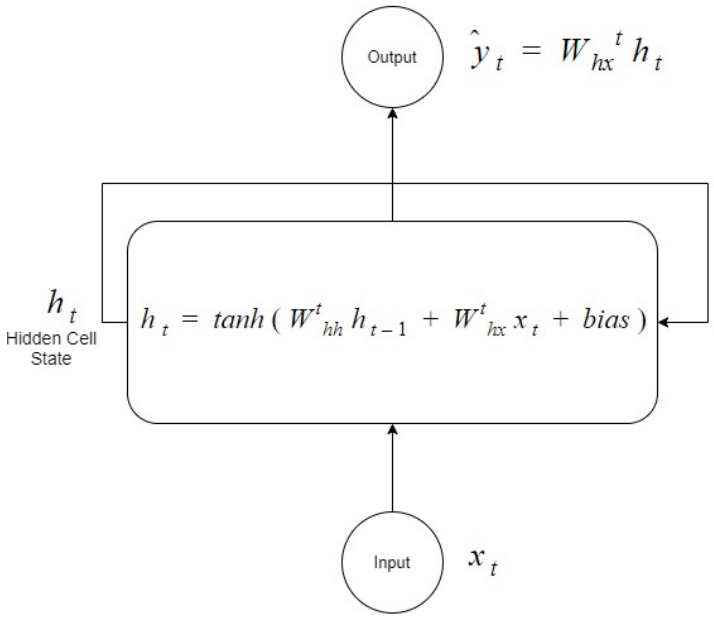


Figure 2.7: Recurrent hidden cell example (adapted from [26]).

To build a recurrent neural network many of these cells need to be connected, where the hidden state of the previous one is one of the inputs of the next cell along with normal input x_t, thus creating a network that takes the previous state in consideration.

In the same fashion to ConvNets, the learning process is done via backpropagation. In RNN case is backpropagation through time. This method goes in two ways, one

it computes the loss of each cell independently and the other way is done by analysis multiple cells, normally addressed as cell chain. Vanishing gradient is a problem that is common in RNNs, because when it is tried to acknowledge long term dependencies the gradient value goes so low that training the network becomes impossible. There are multiple ways to tackle this problem, but one is directly correlated with LSTM networks due to gated cells that have ways to fight the vanish gradient issue.

Long Term-short Memory networks (LSTM)

Long term-short memory networks are one of the solutions inside the branch of sequential neural networks. LSTMs have been gaining popularity due to its performance in many fields such as those aforementioned in the beginning of this Section [26].

As introduced a couple lines ago and as the name suggests LSTM networks are used to compute long term dependencies which is very important because enables the correlation between two inputs that can be far from each other in the temporal space. This is accomplished through gated cells, figure 2.8, and these cells also tackle the problem of the vanishing gradient that appeared when RNN networks tried to learn long term dependencies.

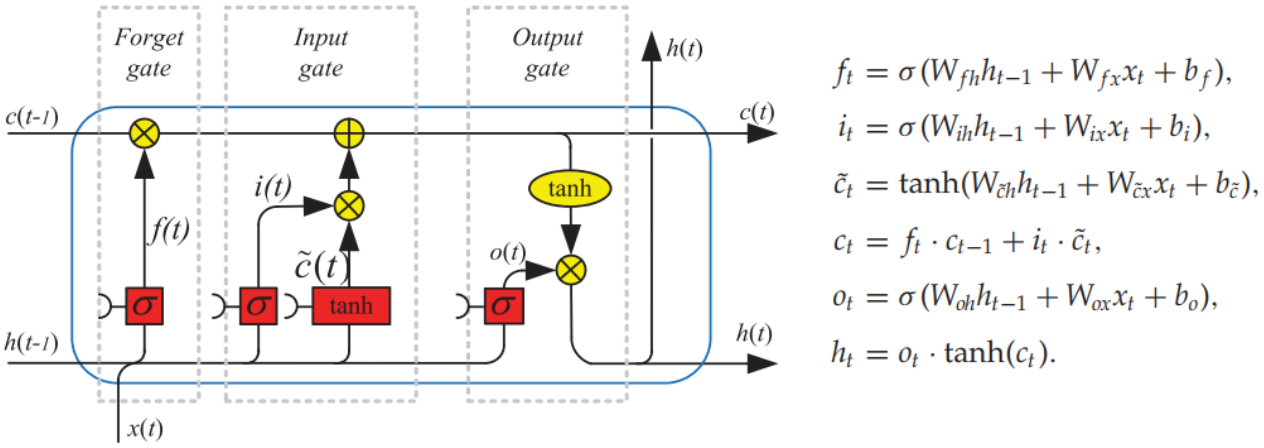


Figure 2.8: LSTM with forget gate and corresponding mathematical formulas (source [26]).

The authors in [26] describe multiple solutions for LSTM networks, however this may be the most popular and with better results. Input gate and Output gate are the standard for LSTM architectures.

Input gate computes both sigmoid and hyperbolic tangent where both receive as input arguments x_t plus $h_{(t-1)}$ and a bias value. This value will be summed to cell state c_t.

To output a value there is the output gate. There are two parts in order to obtain the output value h_t. The first is calculated through a sigmoid with input parameters x_t plus $h_{(t-1)}$ and a bias value, which after will be multiplied by the value obtained from the calculation of hyperbolic tangent with argument the value of cell state c_t.

The mechanism that allows LSTM networks to store long term dependencies is the cell state c_t and the Forget gate. Sigmoid function ranges between 0 and 1, which says when multiplying the result of f_t with the previous cell state $c_{(t-1)}$ all of the information about the previous states of the system could be forgotten in case of $f(t) = 0$, or all information can be passed to the next cell, in case of $f(t) = 1$. Any other value enables a ratio between forgetting and passing all the information.

A network can be built through the combination of multiple LSTM recurrent cells where the cell state c_t and the output value h_t are passed to the next cell.

In [26] can be found many different architectures for LSTM networks and also combinations of LSTM with other architectures.

2.2.4 Applications leveraging LSTM, CNN and Autoencoder

All of the presented neural networks architectures, Convolutional neural networks (CNN), Autoencoders, and Long term-short memory networks had a reason to be studied together. The practical dataset will have different data structures such as profiles and performance indicators but also time series which have to be taken care of in 1-Dimensional world. Conjugating this information requires an understanding of the global panorama and not a temporal slice. For that LSTM networks can manage the importance of factors through time, storing information while it is helpful and discarding information when it is not needed anymore. Another source of study in the thesis are root-cause analysis and failure detection, where the second thrives when using time series data. To extract information from time series both CNNs and autoencoders are viable options, and can be either used separate or together.

The authors of [25] propose a solution combining CNNs, autoencoders and LSTM networks to make predictions in time series data. To take care of the time series data a convolutional autoencoder is used because it offers unsupervised extraction of local features. Both encoding and decoding process involves the learning of a kernel, as it would in a normal CNN, but here adopts the methodology of an autoencoder shrinking the space state, thus compressing most valuable features. Decoding process also requires a kernel to be per-formed. When the signal is reconstructed by the convolutional autoencoder, the output is given to a LSTM network in which each cell takes a part of the signal, forwarding the acquired knowledge. As in a normal LSTM architecture. In the end, each cell gives the computed output and the network shapes the prediction result for the input time series data.

Multiple works combining these types of architectures have been proposed in the most diverse areas where time series are the support of available data. Is possible to combine the networks with each other in multiple scheme orders and proposals. An example is in the work [14] where convolutional networks are used to combine cell state, last output from previous cell and new input to feed into LSTM cells, allowing better learning of important features.

2.2.5 Multi-task

As the name suggests, the Multi-task learning concept is based on learning multiple tasks within a shared model. A task is what the model aims to learn, in our case the service complaints made by the company's customers. The model learning process is done by feeding the network with data from the various tasks, this allows for correlated features between the tasks. The idea is that more complex situations benefit from the learning of simpler ones, in other words, the model does not have to learn every task from scratch. The multi-task technique aims to improve data efficiency, reduce overfitting, and allows faster learning due to mutual benefits between tasks.

Literature in [6] separates the multi-task learning world into two families: hard parameter sharing and soft parameter sharing. Before the definition of both, the authors reinforce that this separation theme is not yet close and prefer to detail hard parameter sharing as the method to create multi-task architectures. Hard parameter sharing says that all tasks are trained together and so the model when in the learning phase has to minimize the loss functions as a group, hence all the weights are shared by all tasks. Complementary, soft parameter sharing is used for multi-task models optimization and contrary to the aforementioned concept in this method each task has its model with separate weights. It is considered multi-task because there is information sharing between the tasks so that they can learn from each other. Figure 2.9 allows a visual conceptualization of the hard and soft parameter sharing concepts.

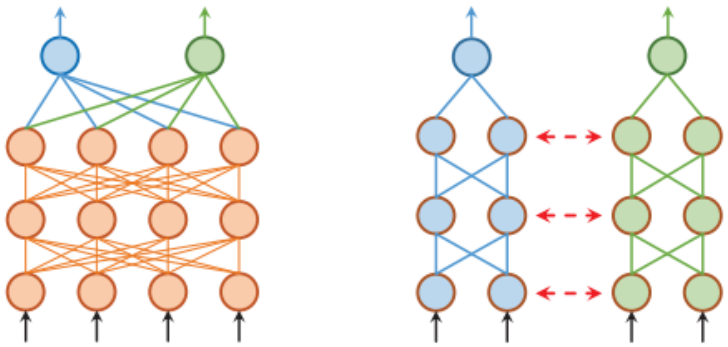


Figure 2.9: Illustration of parameter hard sharing and soft sharing architectures (source [24]).

The orange nodes represent neurons that are shared by all tasks, where blue and green ones are specific for the neural network of each one. These two models can be merged, one can use hard parameter sharing for the common architecture and a soft parameter sharing connected to the merged network output to create branches for each task individually. This may aim to optimize the prediction results of the global model.

Sources in [24] suggest an architecture concept based in sparse parameter sharing. The motivation behind this novel architecture is to give more flexibility to the network,

since tasks can choose which nodes to compute. Doing this search for the better nodes to suit each task creates its own sub-network inside the global model.

3

System Environment

The thesis is focused on a project whose context is very rich and influential inside the company operation and development for newer approaches. Understanding the needs of the customers and being able to act upon them can deliver essential differentiation in a crowded and vicious telecommunications environment. The project aims to develop tools to identify degradation in the delivery of TV and/or Internet services, and proactively approach the client to solve it. As one can imagine, success with this project can impact positively the customer's satisfaction with the service and ultimately with NOS.

The end-to-end project architecture is very complex. A single team does not have either the knowledge or the capacity to implement all pieces needed to create a solution. To implement a project of this size, the team needs to deal with multiple technical challenges and have diversified skills. Every member has been assigned his responsibility to a component of the project. Each component can be viewed as a block that will integrate the general framework. From the multiple blocks composed in the framework, the thesis's main contribution is in the inference block via a new model solution.

3.1 General Framework

Project framework blocks respect multiple processes from technology to business implementations. Only when put together can we address it as a finished product. As a data scientist trying to improve the inference block is of utmost importance to understand the framework. Our job is also to act as an interpreter between the engineers that manage and guarantee the pipeline for the data and the business colleagues that impose requirements in order to go on the field and contact the clients.

In the data science product, we are responsible for 6 main phases: ingestion, feature engineering, inference, leads, delivery, and monitoring. The data science team is responsible for the inference component and partially for the feature engineering and leads. When the inference brokes the whole project is distressed. Because of noisy data and consumption patterns, sometimes the models have to be readjusted which can stop the origination of leads. Without leads the team does not have clients to contact, thus the

goal of improving customer experience is put on hold. Due to that situation, the thesis objective is to have a unique easy to maintain, and easy to evolve the model. The current situation forces the data science team to evolve each model separately and when in failure recover multiple models instead of one.

From the thesis perspective it is important to understand the entire framework, the engineering and the business tasks because we want to act as a connector that delivers a solution for the project.

NOS is a big company, and so, it is segmented into multiple structures each one responsible for a service. NOS provides multiple services: Internet, TV, mobile, and even fixed telephone (legacy). All the company services have to monitor the infrastructure and devices to collect valuable information on the service delivery quality. The service structures are where the raw data comes from. Each structure is responsible for ensuring the performance and maintenance of its assigned service. Service structures collect and store the data for analysis and error detection purposes. The data available for the project comes from the service structures processes. Access to the data is normally restricted just to the service structure personnel. To access, one needs permission to view the content of a structure.

The first block respects the ingestion of the data originating from TV and Internet structures. The part of the team that does the ingestion job acts as a bridge between the low-level acquirement of the data, in the structures, and the inference block where we need data more organized. Data is collected in different conditions. Some are already structured because it originates from a process that needs the data organized. Other is consequential from logs in text, denominated as non-structured data. One of the main contributions to the ingestion block is the creation of automatic processes that receive all data from multiple sources and conditions and manage data into organized fields.

Data is unstable, another job of the ingestion block is to ensure data quality. Signaling, recovery and treatment of failures in the data sources are automated jobs.

A sub-step between ingestion and inference blocks is the creation of the panel. The panel is the collection of variables specified by the data scientist that need to be available. Through a process of aggregations, the panel is created by taking the data pre-processed in the ingestion block and organizing it into a structured dataset. At this point, data can be addressed as variables. The process of aggregation is responsible for two things: one joining multiple sources and the second computing variables from the source given the data scientist requirements.

Panel supports all operations, so errors or failures in the daily creation can jeopardize the success of the project.

The creation of the panel is more than just a computation of aggregations. The whole team had to synchronize efforts to produce the panel. Every member's feedback is required. The data scientist needs to know which variables better describe the dynamics of the system. The models need to consume information, therefore, the variables we feed to the model map the knowledge we want the model to learn.

Figure 3.1 represents the general framework in blocks. The process is sequential, and until this point we have addressed where the data comes from and how it reaches our hands as a data scientist to deploy models for the inference process.

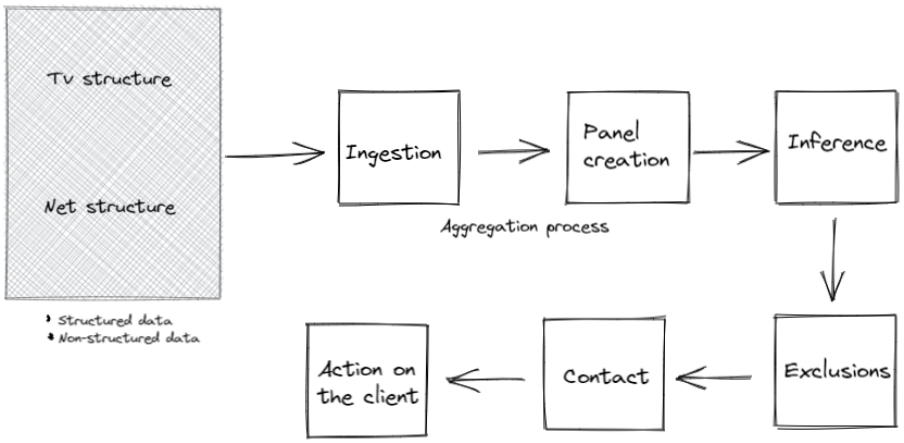


Figure 3.1: General framework schema.

The inference block will be the main focus throughout the thesis report as the proposed objective is to improve current model solution approaches. There are multiple components when referring to inference: exploring of data, creating the dataset, pre-processing, model, and lastly explaining variables that influence results (explainability).

Data exploration is present in multiple phases of development. Most important surmises the need to sustain the ground to justify the committed effort to enhance the project, which needs to be supported by data analysis. Other explorations may surface to debug problems or to help the business operations with insights. Create an ensemble the dataset filters the information we select as relevant for the model to learn. Pre-processing phase aids to clean the data, and transforming variables to be fed into the neural network.

The inference process consists in taking the training data and through a machine learning model, in this case, a neural network, capturing the system dynamics. The learned knowledge is then used to predict results in unseen data to signal to customers that are suffering from degradation in their service. More details about this subject will be explored later.

Explainability is out of scope for the thesis development. However, it is an important step, especially when producing a model to be used in a company's real operation. Actions in real-life cost money, which elevates the importance to explain to the contact team why a customer is being signaled. The evidence in the data that signals that the customer is suffering from problems in his/her service is the reason that explains why our model is better than random and outperforms human prediction.

The exclusion block can be described as a cleaning process that runs on the inference predictions. Contacting the customer is expensive, and so the goal is to be as efficient as possible in the selection. The team defined some filters to select which customers are

eligible for contact. Filters are based on business rules, and the exclusions are processed day by day. Important to notice, the base filter is the confidence the model predicts a customer is in pain. A customer whose service is having problems is considered in pain. However, there are other relevant filters such as if the problem originated in the infrastructure, or if the variables that are out of parameters do not match any use case. Even the case of a customer that has been contacted for a straight period with no response is excluding filter from the list.

The last two blocks of contact and action on the client premises can be clustered in as part of the operation task from the project. The contact teams receive a list with a group of signaled clients, and their job is to approach them proactively via a call to confirm with the customer that indeed he has been suffering from problems in their service.

Given the confirmation of problems in the service, the agent then tries to schedule an intervention via a field agent to the customer's house. Intervention closes a full cycle of the project. From gathering the data, signaling the customer and them intervening to repair the failures in the delivery, all tasks culminate in the improvement of the customer experience.

With the conclusion of a project cycle, the team is able to improve customer satisfaction in relation to the delivered service, but also we impacted positively their perception of the company.

3.2 Data: Input variables and Targets

Real data is ingested from the sources and stored structured within the company's architecture for a distributed environment. Structured data means each defined variable represents a set of information (data) that traces to a specific network measure or a computed measure achieved through the relation of two or more network measures.

From all available variables to be used to feed the neural network model, we selected the set that could capture all the systems dynamics. We mustn't look only at what happens in the present day, trying to catch degradation in the system is very helpful to signal a customer that is on the verge of calling to show his displeasure with the service. Degradation leads to the capacity of leveraging insights into the perception that each client has about his service. Variables that map service usage is also seen as very valuable as they explicitly map if the customer uses the service.

Other variables are very unique to each service, Internet or TV, and their main criteria are to advise on possible problems with a specific component of the service, being the network or the client's equipment. However, there are relations between variables that should not be ignored. Sometimes acknowledging problems in one variable could indicate that the real problem is on another variable, like a sequential effect that is felt on components that are trying to suppress other issues.

For both TV and Internet services, the type of variables follows the same logic. For each group, we choose the aggregated (computed) variables that contain information

about the measures one week and one month before the day we are analyzing. Some are computed differences between one week and one month. Others are calculated specifically for a week before or for a month before and could be counts of problematic events, or statistical measures such as mean, standard deviation, and maximum values. Leveraging weekly or monthly analysis offers the capacity to understand degradation. Comparing differences in patterns can signal a change in service delivery which indicates the customer probably felt issues and will report is unhappiness or if continuously bad over a long period could indicate the customer has given up and is ready to churn. Both use cases should be the target for customer retention and have value to the company.

Internet variables

First, it is important to state that is physically impossible to get all measures from the source for every type of equipment. Although other telco measures could add value, they are out of the study to maintain fairness across equipment. By adding variables that exist only in a couple of pieces of equipment we could skew the decision of the model. Both metrics from connections via wireless or cable are available for the exercise.

- **Signals (DOCSIS 3.0).** Signal variables are separated into two groups. From the field team that supports the service delivery operation, we receive the knowledge of the thresholds, upper and lower bound, each variable is signaled as problematic. By counting the number of times a variable is in the bad zone we can map how the network exchange of information is failing. Although such variables can help predict the target, their capacity to provide explainability is of utmost relevance. For the support team to operate over our predictions is vital to explain via concrete facts why is the customer feeling degradation/pain in their service.

  Signals measures characterize network QoS. Quality in downstream and upstream signals as well as the signal-noise ratio is read from both receiving and transmission sides of the customer and the company. For Internet service Code Error Rate (CER) is an important measurement because measures the percentage of errors happening during the transmission. For CER 100% there is no transmission.

  Another group was created for the model. The objective is through a piecewise linear regression to capture each signal variable slope and intersection dynamic. With this info, we can capture the deviation between each data value and the expected value given by the linear function.

- **Carriers.** Here is mapped the information about the channels transmitting the data. The architecture integrates 8 downstream carriers and 2 upstream. Theoretically the downstream capacity is 50Mb/s, however, it is normal to only reach 80% of the capacity. Carrier information is oriented to disruptive problems, meaning that when the client has no access to the network carrier variables will be signaling problems very expressively.

- **State variables.** Here we have three main events: reboots, resets, and not online or offline status. Resets and not online or offline status can be valuable when their frequency is much higher than what is expected. In that case, probably the equipment is causing the problem.

  Reboots may be one of the most interesting variables but also one of the more dangerous to consider. They can easily break the model due to their strong correlation with the target. The difficulty is that reboots can happen for multiple reasons, including signals transmitted by the company for the device to do a self-reboot, but also could be an event with origin in the customer. The customer can experience problems and he/she can reboot the device to try to fix it or simply the customer at some part of the day turn the power down. All these possibilities make us think about the reboot variable more as a target or a trigger than an explicative variable.

- **WiFi variables.** Throughout the thesis, the challenge has been how to map the pain of the client. Through WiFi measurements, we can detect the usage of the service. For both ethernet protocols, WiFi at 2.4GHz and WiFi at 5.0GHz we can know how many devices are connected and their status with the network. This gives us the possibility to feed the model with clear information about the Internet service usage for a given client.

  The cable modem is a computer, and so has the hardware components that a common computer would have. High usage of Central Processing Unit (CPU) and memory can indicate a hardware failure. However, having both these measures in the danger zone can tell us much more. The cable modem components can be trying to mask problems in the network. For that reason, the CPU or memory is excessive effort due to the need to fight issues that originated in the network.

  The number of hops between the available channels is another variable that provides important information. The protocol does not program the modem to search viable connections in multiple channels. When this happens it is probably due to saturation.

- **Traffic variables.** Traffic variables measure the transmission of bits in the network between client and server. Although the information has direct relation with the user perception of the service it is not trivial to map the information into knowledge. Because of the bit transmission there is also signalling traffic that is not directly mapped to the user usage.

TV variables

The selection logic for TV service is very similar to the described in the Internet service. Even, for signals variables, state variables and traffic variables the overlap is almost total. Notice that although the logic very identical for both services we know from figure 3.1

that the source where the data comes from is different. Moreover, the responsibility over the source data is in two completely distinct teams. The process done in ingestion and panel creation are the steps that look for the same information in source data and transform it into structured variables with similar meaning.

- **Signals variables (DOCSIS 2.0).** The group of variables for TV signals is exactly the same as for Internet service. The objective is repeated, having variables that can corroborate our predictions and explain to the field team why the customer is in pain.
- **State variables.** Reboots remain a case of study with highly descriptive information but dangerous to be used in detriment of hurting the model performance. Resets and online/offline status have useful information if way out of normal values.
- **Traffic variables.** Same as in Internet, traffic variables measure the bits transmitted between user and server (upstream and downstream). We add a couple of target like variables, meaning binary, that signal bad traffic events over separate windows.

Apart from the three groups of variables aforementioned, there are more four groups of variables only adopted in the TV service. The set of the four groups identifies all tools that bring the distinct TV services to the client. Each sector has its own set of variables and problems and may be independent from the rest of the service. However, it is not to disregard the possibility of problems occurring in one sector triggering bad events in a different sector.

- **Image delivery variables.** Device free memory and CPU load, similar to Internet, can indicate hardware problems with the device. However, they can be triggers for another issues as they could be in excessive effort given the necessity to fight other problems.

  The older model equipment only had 2 tuners, while the more recent version has 3. Tuner job is to deliver video and audio to the customer, and each tuner is responsible to fetch a channel. Tuners correlate with QoS delivery of the linear TV service.

  We also have flag variables whose objective is to signal if the problem is the hdmi connection or if the customer does not have permission to the channel he wants to see. It is very important to validate flag variables because they allow a quick and remote solution of the problem.

- **Frontend calls.** Frontend variables map the interaction of the customer with the apps/software available in the TV service. The number of times the device experiences IP loss is count because in cases of abnormal quantities it is considered an issue.

  We map which calls are made to the servers and aggregate via percentile, being 100 the top service call functions used by all our customers in a day. First, we map the

entire customer base patterns to create a baseline. To understand if a customer had difficulties connecting to the frontend we compare his behavior against the baseline. If a customer calls many times for one function, but that specific function is the most called by the all customer base, probably there is no problem. On the other hand, we can compare the function the customer reached more times, and if his off from the customer base pattern maybe be a signal of bad event. Notice that the duration of calls is also important for debugging of a interaction, and high durations can signal of malfunction.

- **Session variables.** Mostly overall information about the customer session. Important to retain, long session setup times mostly correlates with degradation of the service.

- **VOD and NPVR variables.** Non-linear TV services, are mostly when the client wants to see content that is saved, or already passed in linear TV and VOD (video club). Variables have information about the services and the quality of the interaction which can be used for explainability purposes or to aid in target signaling.

Targets

Target variables signal events. The thesis proposal is to identify when a customer is experiencing problems with their service, degradation, and disruption. The event that signals unhappiness from the customer towards the service is when he calls complaining, which is referred to as technical participation (Pt). Technical participation can be done for either service. When the client calls, he describes the problem and the operator opens an occurrence for the customer acknowledging technical participation for the Internet or TV service. The targets we will feed to the model are Internet service technical participation (Pt Internet) and TV service technical participation (Pt TV).

For disruptive problems, when the client loses complete access to the service is out of the thesis scope. Since we are not working in real-time, even if we could identify a disruptive problem our action would not be in time. For degradation problems, however, it is possible to act upon the clients' issues before they feel the impact of the service experience. For that reason, knowing which set of variables is in bad shape when the customer complains gives important information. When selecting variables to capture the system dynamics we focused on groups that explained variance and changes in pattern. When the customer opens technical participation he activates the target, and now we have the information that the customer complained about. Having that information allows us to map every day before the Pt, where we can search for disruptive problems, but more importantly, we can search for degradation in the service thus gathering knowledge on the customer experience until the complaint.

By learning which factors lead to a technical participation event, we can create a predictive algorithm. The proposed idea is to proactively signal a customer whose variable

values are out of bounds. We can achieve a signaling prediction by analyzing the variables historically over a month. If detected degradation in the service delivery, our confidence about the client experiencing problems increases, and thus sooner or later he will complain to the company. If we can call him before he calls us, it is known that his perception of the service and the company will increase, thus increasing income, up-selling, and other business factors.

Summarizing the data section, we gathered a set of variables to capture the system dynamics and defined which is the target we want to learn. In the end, we are left with a dataset that has information about multiple months of data and each row represents a customer in a given day. For each row, all variables are filled with the respective values and, we signal the target as 0 or 1 (binary). When the target value is 1 it indicates that customer called in that day and placed a technical participation, otherwise there is no occurrence to signal.

3.3 Project importance: Use case analysis

The thesis goal is to validate that Internet and TV services have a higher correlation than is acknowledged by the team and should not be considered independent. It is important to demonstrate that searching for a higher intersection between models makes sense. Resources are allocated for the thesis, and also, the expectation for results is very demanding as validating the intersection will open new opportunities.

Constructing a model on top of a problem that from scratch does not make sense will probably yield bad results or, if good may not suit anything. So, we validated the thesis goal as viable to be explored through two analyses: customer service usage and the intersection between services when resolving problems at customers' houses.

Internet service and TV service have some differences regarding input variables. However, as stated in Section 3.2, QoS signal and state variables are the same for both services. Both signal variables and state variables are used to explain why the customer is signaled. They contain information on why the service is experiencing degradation. For the analysis of use cases to make sense, it is important to select the group of clients as similar as possible to the group we want to address through the model. Only customers who experience problems in five days or more over a month are eligible for selection. To mark a client with problems on a given day, at least one of his metrics in signal or state variables needs to be off.

In Section 3.2, we gathered knowledge regarding the available variables to understand what is their influence on service delivery. To filter the customer base with degradation in their services, we need to analyze data distributions and set the thresholds we consider a client as having problems.

Over a period of one month, we analyzed the distribution of signal and state variables. The chosen metrics represent counts on the number of hours in a week the client had a metric above or below the defined interval for good functioning. Once again, the focus is

on degradation. Selected variables map the number of hours in a week, so if a customer had disruptive problems in one day will not count for the use case analysis because does not meet the criteria of continuous problems over time.

Although the number of filtered clients has had problems is significant, the majority of the customer base has a good service. The graphics used for analysis are histograms representing the frequency of the number of hours with problems for each variable. The information is enclosed in bins. As expected, the bin containing zero or a few hours with problems contains the majority of the cases. For that reason, we decided to omit from the graphics the first bin, to have a better visualization of the distribution.

The analysis focus on distribution insights and threshold placement. The selection of clients for the use case analysis depends on the threshold defined in each variable distribution. Graphics are representative of both services, and thresholds defined over the analysis are valid for both services as well. Graphics are analyzed individually, which makes scales being different not an issue.

- **Upstream tx hours in a week.** Histograms in figure 3.2 represent the number of hours transmitted upstream was below the acceptable minimum (left figure) and in the right the number of hours was above the acceptable maximum (right figure).

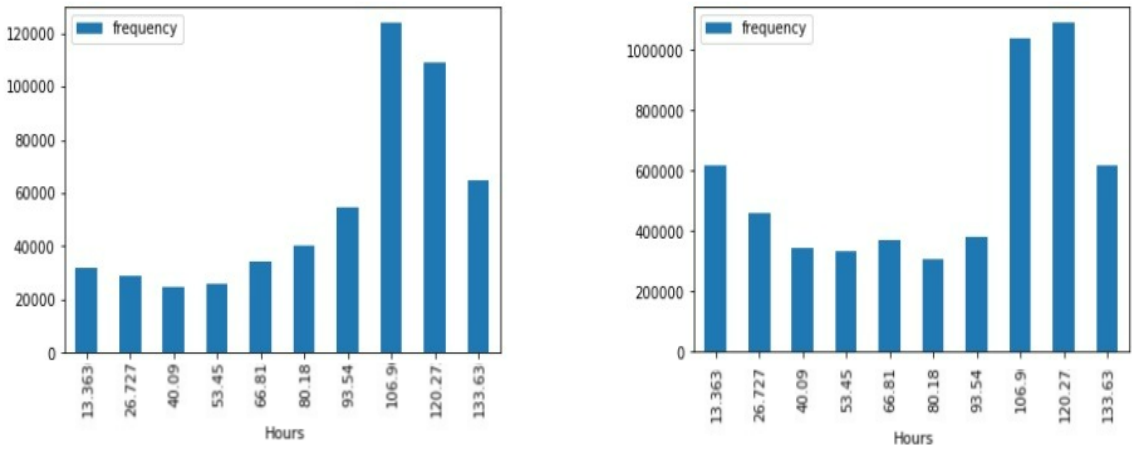


Figure 3.2: Upstream transmitted (Up Tx) metric.

Upstream (Up) Transmitted (Tx) metric measures how good is the communication between the client equipment and the company's servers. It is more natural for the upstream transmission to have problems with high values. We select the threshold in 100 hours because that mark splits the graphic into two zones: the normal zone and the exponential zone. The normal zone, below the threshold, can be explained by the low service use or because the service is starting to degrade. The exponential zone is represented by the big increase in the number of customers that have problems over 60% of the week. In the exponential zone can be the clients in pain

because they try to use the service but experience problems. For the use case analysis, we selected the group of clients in the exponential zone because it is the most probable to use the service and complain.

- **Downstream rx hours in a week.** Similar to the first variable in the exploration, figure 3.3 (left) the number of hours the received downstream was below the acceptable minimum and the number of hours was above the acceptable maximum (right). Received power maps the quality of the signal that reaches the clients' router or box. It has influence in the quality experience when using the service.

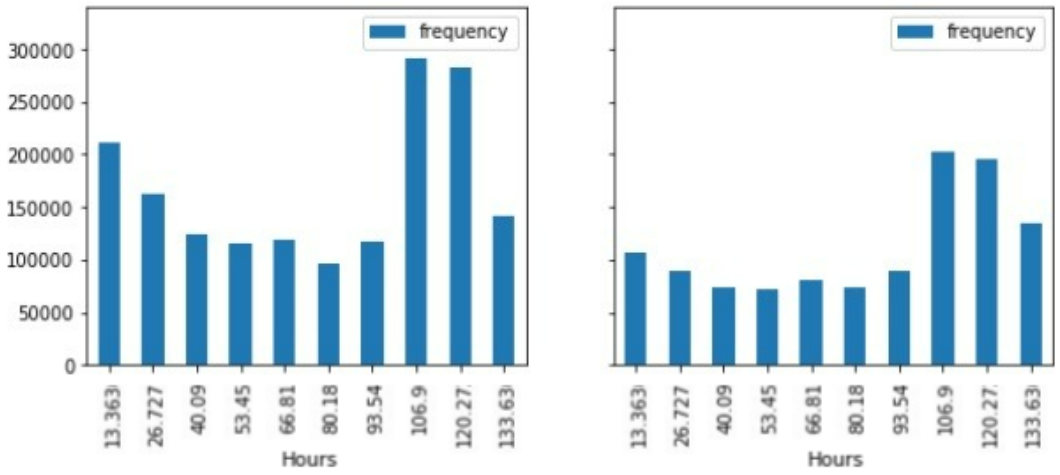


Figure 3.3: Downstream received (Dn Rx) metric .

The analysis is the same as in figure 3.2 in terms of the normal zone and the exponential zone. A customer is marked has having problems in a given day, when it is in the exponential zone. Customers with problems over 60% of the hours in one week will probably be in pain, thus are our target.

- **SNR hours in a week.** Signal-Noise Ratio (SNR) metric correlates directly with the power of signal available in the customer service. In figure 3.4 we can notice that the problems are stronger in the downstream measurement (graph on the right). If the equipment does not receive a powerful enough signal, for TV service the customer will not be able to see anything. In Internet service it will have difficulties connecting to any service or website.

  The threshold is defined again as 100 hours. There is a pattern with the signal variables that when a customer is experiencing degradation and probably uses the service, the customer will experience pain during at least the entire week. Customers selected for the use case study belong to the exponential zone.

- **CER and BER hours in a week.** Code error rate (CER) and Bit Error Rate (BER) are discriminatory variables. CER is applicable to Internet service, while BER is

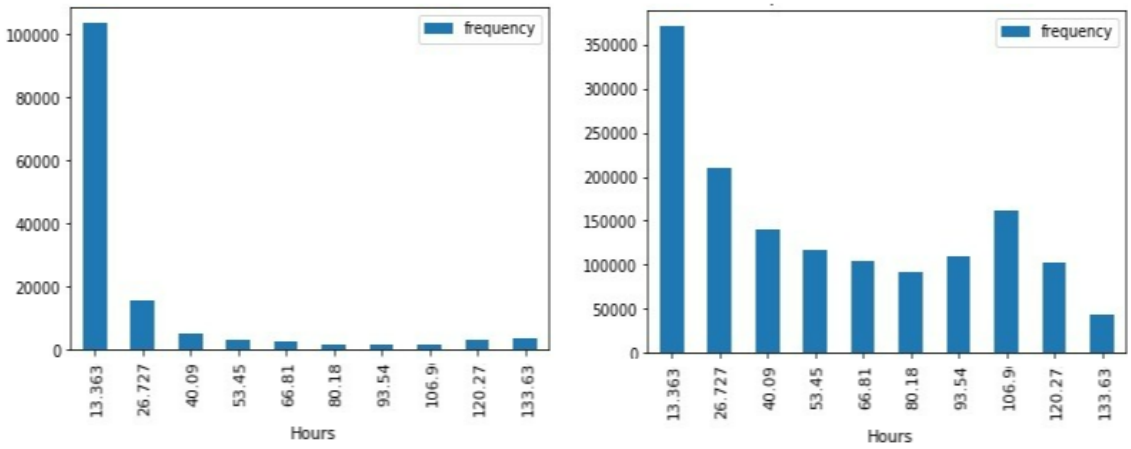


Figure 3.4: Signal Noise Ratio, upstream and downstream.

applicable to TV service. When these metrics are out of parameters it means that the servers are experiencing problems and cannot transmit information. CER and BER are only activated when the first mechanisms of error correction fail. When we see a customer with many hours with problems of CER or BER he is in pain.

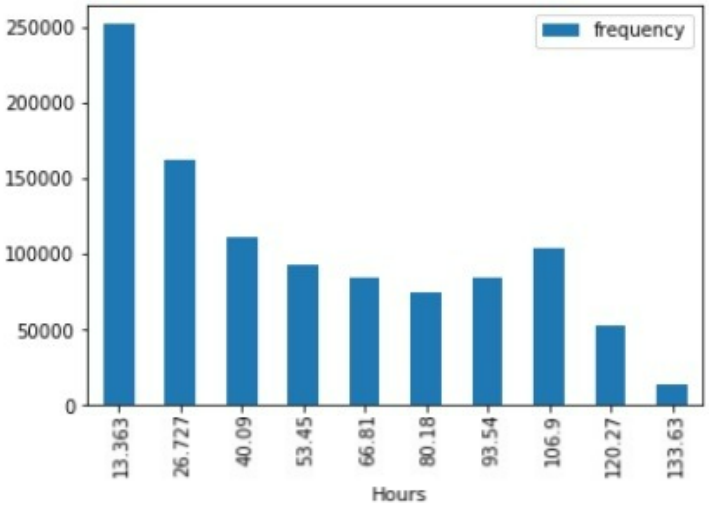


Figure 3.5: Downstream hours with problems distribution representative of CER and BER.

Fortunately, the distribution is right skewed and normally our customer does not have excessively hours of problems. However, after the 80th hour mark we can see a slight exponential zone that suggest customers in pain. We mark customers has having problems in a day if they are over the 80 hours threshold.

- **CCER distribution hours over a week.** Code Error Rate (CER) happen when CER

is triggered. The objective of CER is to aid in transmission error corrections. Normally CER starts before CER as a preventive measure, a reason which explains why both variables CER and CER have exponential zones in the same hours. Also, an explanation to why the volume of clients with CER is bigger maps exactly to CER starting preventive.

Clients are not signaled through CER, because the company only acts on CER when combined with CER.

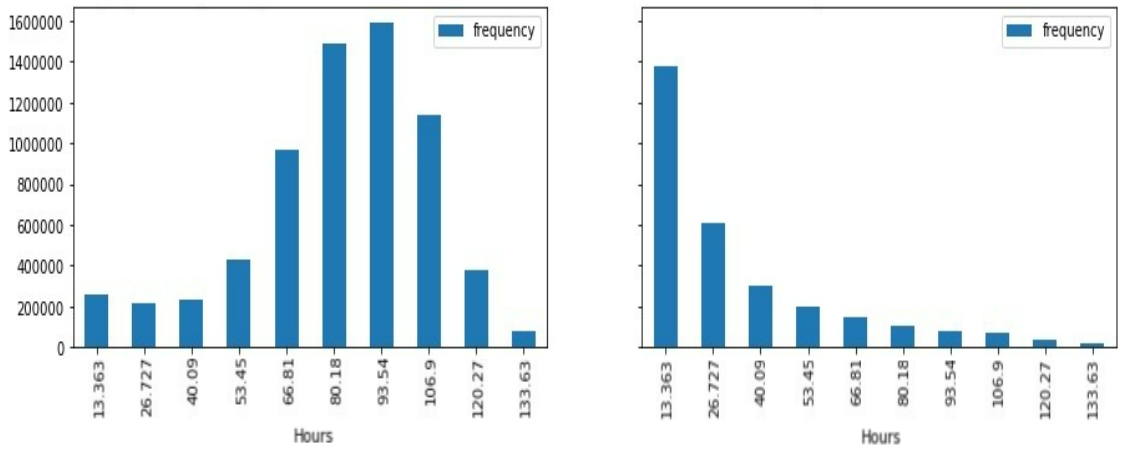


Figure 3.6: CCER hours with problems distribution distribution. Has high correlation with CER. Left graph is downstream, and right graph respects upstream.

- **State variables.** As mentioned earlier, reboot variable can be originated by multiple factors which makes it dangerous to model. Both reboots and resets map disconnections from the router or setup box to the network or with the supply power. Most of customers is represented in the zero bin which was taken off. The behaviour in the zero bin is seen as normal. Figure 3.7 shows the distribution in quantity of reboots (right graph) and resets (left graph) customers had during a week period. Every number of reboots or resets over 4 or 5, is considered potential indicative of problems. For that reason no threshold was selected as we do not have confidence in reboot and resets variables patterns. By using these filters to select customers to the use case exploration we could bias the result. The intersection and service usage analysis could then lead our work to a wrong path or simply inconclusive insights.

  Last explored variable is the device state of connection (not online not offline). As observed in figure 3.8 we again have an exponential distribution for multiple hours with problems. The threshold is set at 110 hours, to catch into the analysis the group of customers that had the entire week with not online not offline metric out of bounds.

The analysis made to the variables allows gathering knowledge on how the service behaves. The objective was to select appropriate thresholds in order to filter for clients

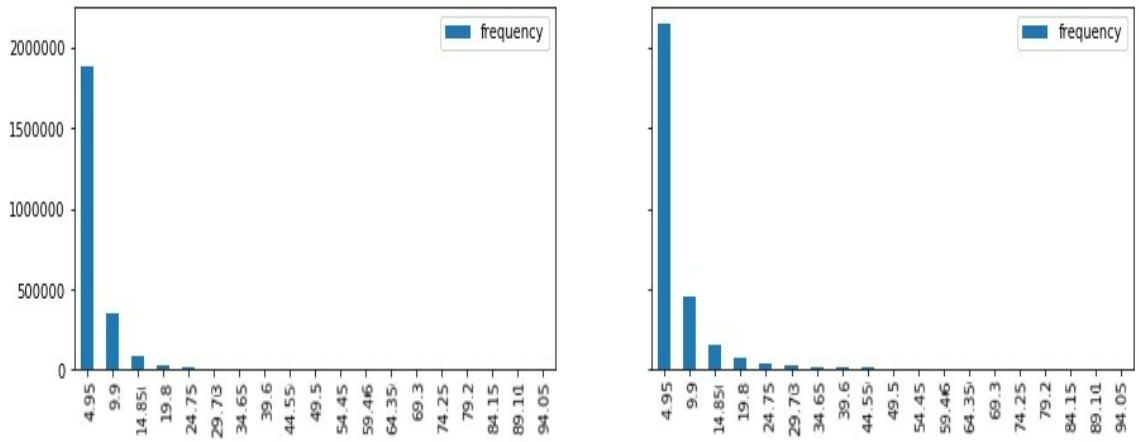


Figure 3.7: Reboots and Resets events distribution.

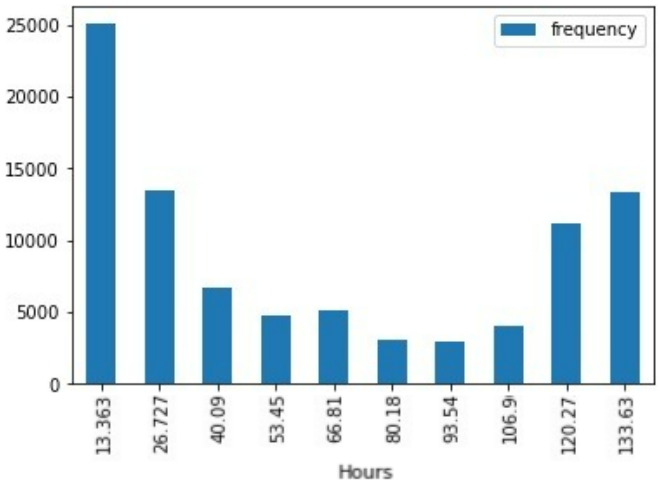


Figure 3.8: Not online not offline hours with problems distribution.

that indeed were feeling degradation in the service. Now, with the group of clients in pain, we explore the veracity of the use cases.

Intersection between Internet and TV services

The thesis project is born from the necessity of having a unique model that is easy to maintain and evolve. Current solutions are scattered, and even within each service solution, there are multiple models. In the telecommunication world, customers service use denotes very high entropy, as customers' needs and desires change very fast. When changes occur, sometimes there is the need to rethink the model to accommodate the new service use trends. Having many models to retrain can be time-consuming, so constructing a unique model solution would immensely increase the easiness and quickness we can react to the change in consumption patterns.

Another prominent factor is the intersection of the on-field operation between TV and Internet services. In the current model solutions, the customers are signaled for problems on TV service or in Internet service. Depending on which variables are triggered, one of the models can mark the customer with degradation issues. However, more times than expected, when we signal a customer has had problems in the TV or Internet metrics, the field operation receives the opposite feedback from the customer. Multiple times the customer acknowledges that he feels pain but on the other service. Meaning if we call due to bad TV metrics he will complain the problem is the Internet service and that the TV service is good, and also the other way around, bad Internet metrics but the clients' perception is that the TV service is bad. The situation relates to the perception that the user has of the service, QoE, and correlates with service use.

It is important to validate the intersection between services to understand the environment and what it is expectable to come across in the thesis research. We made an analysis of each service independently and for the intersection between them. The analysis maps the clients whose problems extended for 5 or more days in a month. For the clients with potential issues in the service, we make a discriminatory analysis on the probable symptom for the problem and the result of the on field repair.

Internet service is addressed first, with no special difference in the order. Figure 3.9 shows the distribution of clients by the number of days in a month. The clients were signaled due to issues with the Internet service. The number of clients identified as having problems over 5 days is significant, around 1/4 of the total client base. To analyze the Internet service graphic it is advisable to divide it into two zones: the first days designated as the normal zone where there is noticeable degradation, and the zone of 25 days plus where the majority of the clients are.

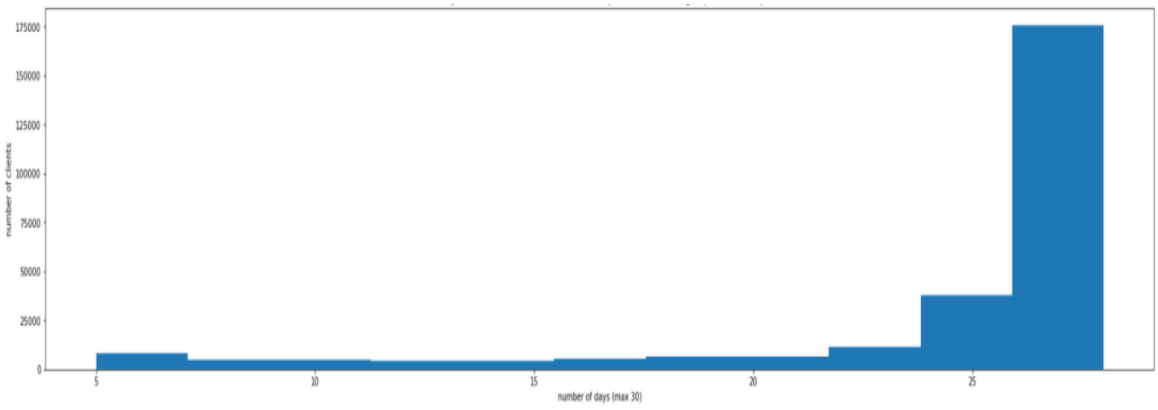


Figure 3.9: Distribution of clients by the number of days in pain over a month. Graphic respects Internet service, clients whose Internet metrics were with problems.

In the normal zone, in the case that the client experienced degradation in the service, probably called the company and the problems were corrected. In the zone where the majority of the clients are, we need to be more careful with the analysis. The mass could

represent two phenomena: the client does not use the service and hence does not have pain, or the client indeed tries to use the service but it is unable to.

From all clients filtered as experiencing bad Internet service, we zoom in on the ones that reported their unhappiness via a Pt. To construct the analysis data we grouped the Pts by probable failure symptom/cause. When the operation unit goes to the client's house, they solve the problem and report which type of solution was enforced to correct the problem. Table 3.1 contains the main symptom reasons and some solutions. Displays in a matrix like shape the information that correlates symptoms felt by the customer in the time of Pt and the respective solution discriminated by the technician.

Table 3.1: Symptoms and resolutions for Internet service Pt.

Symptom	Net cable equipment	Net cable service	TV digital service
Resolutions			
Device exchange (Net)	2357	1166	149
Device exchange (TV)	1206	89	7
Device exchange, NOS request (Net)	24	44	685
No anomaly	164	191	196
Broken splitter	28	131	99
Coaxial socket degradation	13	67	99
HDMI cable faulty	0	2	100

The total number of the clients identified in figure 3.9 that reported their discontent was 12 839, which respects to a percentage of approximately 4.8%. The remainder of clients may belong to the group that does not use the service or does not bother to resolve the problem. Another explanation, and the best situation possible, is when the problem can be fixed at distance.

Just by analysing the table 3.1, we can validate that the intersection between services is real and relevant. From the customers we identified with Internet service issues due to their Internet metrics, many have symptoms related to TV service. Also, part of the resolutions with symptoms related to Internet, was accomplished by addressing TV network or devices. Thus, is expected that by exchanging information between services dynamics we can improve the exercise capacity.

The proportional distribution of symptoms are presented in table 3.2.

Even disregarding smaller proportion symptoms, we can see that TV digital service takes almost 1/4 of the total symptoms for the analysis of the Internet problems. The number is significant and supports the premise that services have an intersection.

Analyzing exercise for TV service was constructed following the same blueprint. Again, we selected the clients whose TV metrics showed pain over 5 plus days. Followed by the detailed description of reported events by a subset of pain clients. The descriptive table 3.3 contains information correlating symptoms that triggered the Pt and the solution enforced to correct clients' issues.

Table 3.2: Symptoms proportion for Internet Pt.

Symptom	Relative proportion
Net cable equipment	34.6%
Net cable service	25.2%
TV digital service	23.2%
Net service	4.9%
Quality of Service	4.5%
Automatic recording	1.9%
TV cable digital service	1.9%
TV analogic service	1.5%
Videoclub	0.9%
Guia	0.9%

First, figure 3.10 plots the distribution of clients over the number of days in pain they were signaled in a month. Comparatively, with its counterpart of the Internet service, we can notice a normal zone in the middle that can be associated with degradation. Similarly, a big proportion of clients is identified with pain for the majority of the month, more than 25 days. The analysis is identically careful, that group of customers could be using the service but with a bad experience originating pain, or could be customers that do not use the service and their perception of the service is unnoticed.

The big difference to report is the proportion of clients with 5 to 7 days in pain in comparison with the Internet service is a big swing. Our interpretation is that the group of clients in the interval could have called via Pt and got the problem corrected, making them a group of customers that use the service thus their pain against the service went up. Or, could be a momentaneous disruption, maybe even caused by the delivery network (e.g. storms, network updates, events) that affected the service but went back to normal.

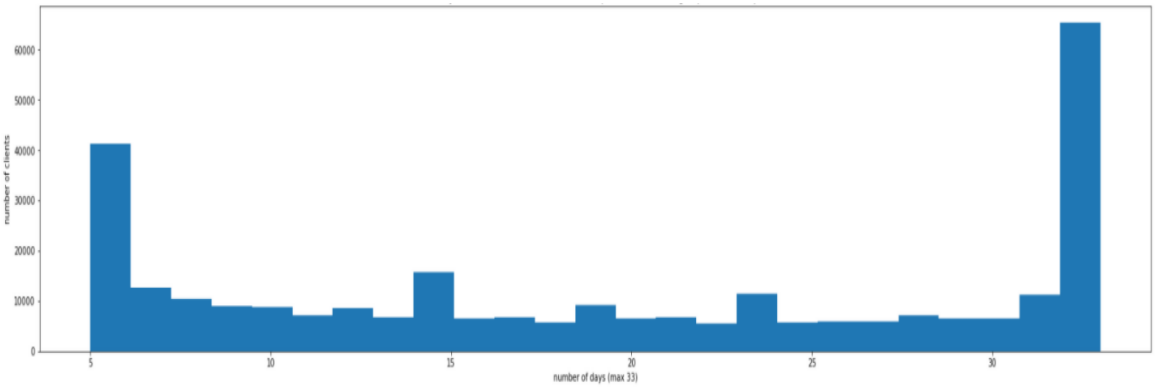


Figure 3.10: Distribution of clients by the number of days in pain over a month. Graphic respects TV service, clients whose TV metrics were with problems.

Number of customers identified with bad TV metrics more than 5 days is similar to the Internet one, around 1/4 of the entire poll of customers. From the quarter filtered clients selected for the analysis, we zoom in to evaluate how the Pts distribution were

and how was their resolution. Through table 3.3, we can analyze the correlation between symptoms and resolutions. With that information we can check the proportion of intersection between bad TV metrics and Internet services Pt.

Table 3.3: Symptoms and resolutions for TV service Pt.

Symptom	TV digital service	Net cable service	Net cable equipment
Resolutions			
Device exchange (Net)	99	854	1236
Device exchange (TV)	844	44	20
Device exchange, NOS request (Net)	6	54	535
No anomaly	207	154	91
Broken splitter	211	204	15
Coaxial socket degradation	174	74	13
Unbalance in coaxial network	113	87	20

For TV service, the number of clients that complain is approximately 4%, a total of 10 374. Same reasons why clients did not complain in Internet service, apply to TV service. Again service usage is an important factor worth exploring.

Once again, we can see that 2 of the 3 main symptoms are related to Internet service. From TV service point of view, this has theoretical support. In digital TV service when the client is experiencing problems, the service just does not work. In such situations, the TV metrics will be all over the place signaling the customer. However, the customer may not be watching TV but using the internet service. A situation when is an infrastructure problem will also be down or have a lot of issues. It is expected some Internet service Pts caught when looking for TV metrics, yet, the number may be bigger than expected.

Another note, that goes for both service analyses, is understandable for the first triage of the problem to be off the mark. It is difficult for a person through the telephone to diagnose the probable cause of failure. Only when the technician goes to the customer's house and is in the presence of the equipment and the network is capable to do a more thorough analysis.

Table 3.4 aggregates the information by symptom and provides the total contribution of each one.

The presence of Internet service Pts is more influential in the TV analysis. Here almost 50% of the Pts are Internet symptom related. Some are due to the multiple reasons explained during the discussion, and the other can relate to users being much more sensitive to failures in Internet service. If the TV lags a little bit in non-linear TV or the images from another channel preview do not appear, customers do not feel pain. However, if a middle of a multiplayer online game the service as a slight issue the company will certainly receive a Pt.

From the exploration of the TV and Internet services independently, it is visible that the intersection between the services is more than what is currently acknowledged. Based

Table 3.4: Symptoms proportion for TV Pt.

Symptom	Relative proportion
TV digital service	31.0%
Net cable service	23.9%
Net cable equipment	22.3%
Automatic recording	4.9%
Quality of Service	4.3%
Net service	3.6%
Videoclub	3.1%
Guia	2.3%
TV cable digital service	2.1%
TV analogic service	1.6%

on the obtained results, we went deeper into the research. The goal denoted trying to characterize the expectable intersection between the services.

The set of customers for the analysis is the inner intersection of the clients selected for Internet service and TV service individual analysis. Meaning, that all clients whose TV metrics and Internet metrics were out of parameters for 5 days or more belong to the study group for the intersection exercise.

The total number of customers with problems in both services is 96 770, which is around 10% of the total HFC customer base. The approximately 96 000 clients are grossly 40% plus than what is currently being addressed by current models. Before analysing the table, it is important to understand the intersection could belong to three different cases:

- **Common infrastructure.** represents the use case of the problem origin being in the delivery network, which has common elements, and thus both services experience bad quality of service.

- **TV metrics with Internet resolution.** technical resolution being on Internet service while the client was detected over bad TV metrics.

- **Internet metrics with TV resolution.** exactly the counterpart, is the technical resolution being on TV service while the client was detected over bad Internet metrics.

Table 3.5 compiles the information about main symptoms and resolutions for clients where both services experienced bad quality in more than 5 days in a month.

From the table, we can state distributions are like the ones for independent services. Pts respecting Internet service take a larger share, but that situation was already predictable as customers tend to be more sensitive to failures in Internet service. Although some cases where the symptom service is a mismatch to the resolution, the vast majority of the triage is correct, and the technician confirms through the resolution at the client's house.

Table 3.5: Symptoms and resolutions for services intersection. Analyze on Pt symptom.

Symptom	Net cable equipment	TV digital service	Net cable service
Resolutions			
Device exchange (Net)	871	62	473
Device exchange (TV)	9	355	22
Device exchange, NOS request (Net)	422	2	39
No anomaly	60	92	87
Broken splitter	10	92	115
Coaxial socket degradation	16	66	41
Unbalance in coaxial network	11	54	58

The number of customers who reported Pts within the intersection group was 5 628, which represents a proportion of approximately 5.8%. The conclusion to take here is that when addressing the intersection between the services the proportion of customers we can signal goes up by almost 2% versus TV analysis and 1% comparatively with the Internet service. For the exercise is of utmost importance the validation obtained in the exploration. It is more obvious that there is information at the intersection of the services. We aim to signal more and different profiles of clients whose problems may not be addressed because the insight on their service degradation is reflected in the other service.

Once more, table 3.6 compiles the information of Pt symptom shares across the different cases.

Table 3.6: Symptoms proportion for intersection.

Symptom	**Relative proportion**
Net cable equipment	29.1%
TV digital service	25.8%
Net cable service	25.0%
Quality of Service	4.9%
Net service	4.7%
Automatic recording	3.1%
TV cable digital service	2.1%
Videoclub	1.7%
Guia	1.6%

Customer service usage

One topic that surrounds the thesis is quality of experience. QoE depends highly on the customers' perception of the service, hence service usage is a prominent factor because a customer that does not use the product will not complain. Our efforts are channeled to identify which are the customers that use the service and are experiencing either degradation problems or disruptive malfunctions.

In service, intersection analysis was possible to validate customers do complain about the service which is not the one we signal has having issues. All the reasons why that happens or even why multiple times the customers do not even complain are also extremely important when contextualizing service usage. And also, the explanation for many of the customers' actions is correlated with their use of the service.

Before start working on a predictive solution, the decision was to validate if the customers we signal as potentially having degradation problems use the service. The logic to accomplish the analysis was to filter for the customers who had five or more days with problems in one month. With the filtered group of customers, we analyzed the number of transmitted bits between the customer and the network through the traffic variables from both Internet and TV services.

- **Internet service use.** Regarding the Internet service we measure the upstream traffic via the variable (up_traffic_1w_avg). Upload variable maps the average flux of bytes transmitted in the upstream direction over a week. Figure 3.11 shows the distribution of clients by their average service use under 10 Gb (top), and the distribution of clients with average usage under 1 Gb (bottom).

  Ahead of stating conclusions over the analysis of the upstream usage flux, is advisable to explore what happens with the downstream flux. For that we recur to the downstream variable (down_traffic_1w_avg) that maps the average bytes exchange over a week via the downstream direction. Figure 3.12 represents the distribution under 100 Gb of download content in a month, and in the bottom represents the same measure but for clients with traffic under 10 Gb.

  The obvious conclusion to infer when analyzing both graphics pairs (downstream and upstream) is that the customers signaled indeed use the Internet service. This is a very important conclusion because we validate that the pool of customers having degradation in their service is indeed in pain, thus supporting the effort made in the thesis to address the problem.

  The volume of clients consuming fewer bytes in the network is in proportion significantly less compared to the number of clients that consume more bytes. This tells us that the majority of signaled clients are over 1 Gb and 10 Gb of consumption, upstream and downstream respectively. Such analysis supports the statement that the group of customers in pain use the service.

  The high spike noticed in every graphic (left side close to zero) is normal and we do not consider it abnormal comportment. That bin encloses all clients whose service usage is from 0 (nothing) until 1% of the maximum of the graph (e.g. 1% of 1Gb; 1% of 10Gb; 1%100 Gb).

- **TV service use.** Exploratory process over clients signaled as having degradation problems for TV service is identical to the one done for Internet service. The main

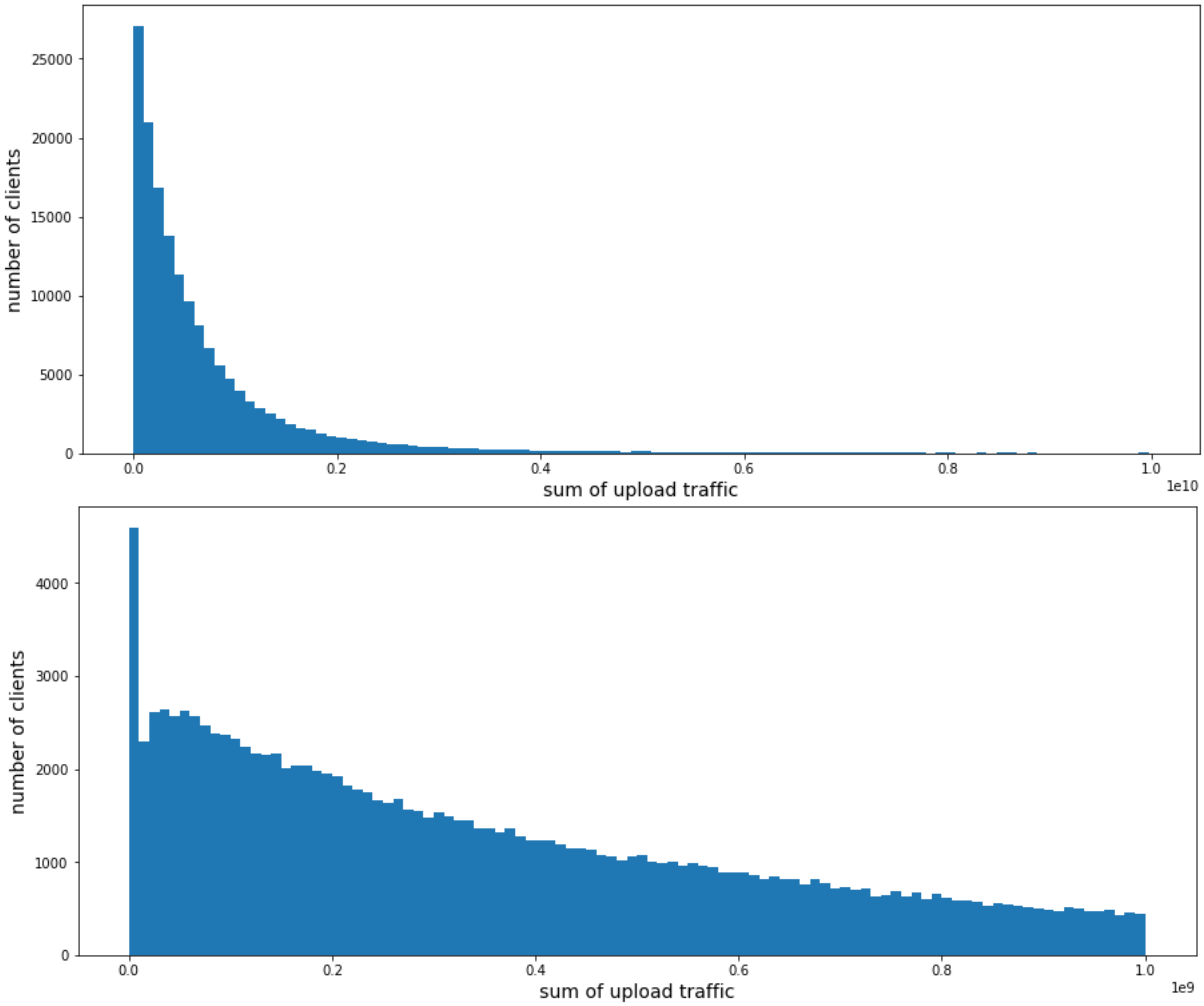


Figure 3.11: Distribution of clients in pain over 5 days given their use of Internet service (upstream). Filters for clients with monthly usage below 10Gb (top graphic). Filters for clients with monthly usage below 1Gb (bottom graphic).

difference is that the traffic variables, upstream and downstream, for TV service measure the absolute quantity of bytes transmitted between server and client device instead of mapping the average.

Figure 3.13 shows that the majority of customers signaled use the service between 1Gb and 10Gb on upstream traffic. Just through a brief inspection it is noticeable that the pool of clients use the service. Figure 3.14 plots the same reasoning but for the downstream flux.

In both distribution pairs, upstream and downstream, we can state that the clients signaled with degradation use the service and thus are in pain. For TV service as well we can validate that the customers where the thesis is focused belong to the group we proposed to act upon.

In the TV service graphics, the bin close to zero that gathers the volume of customers

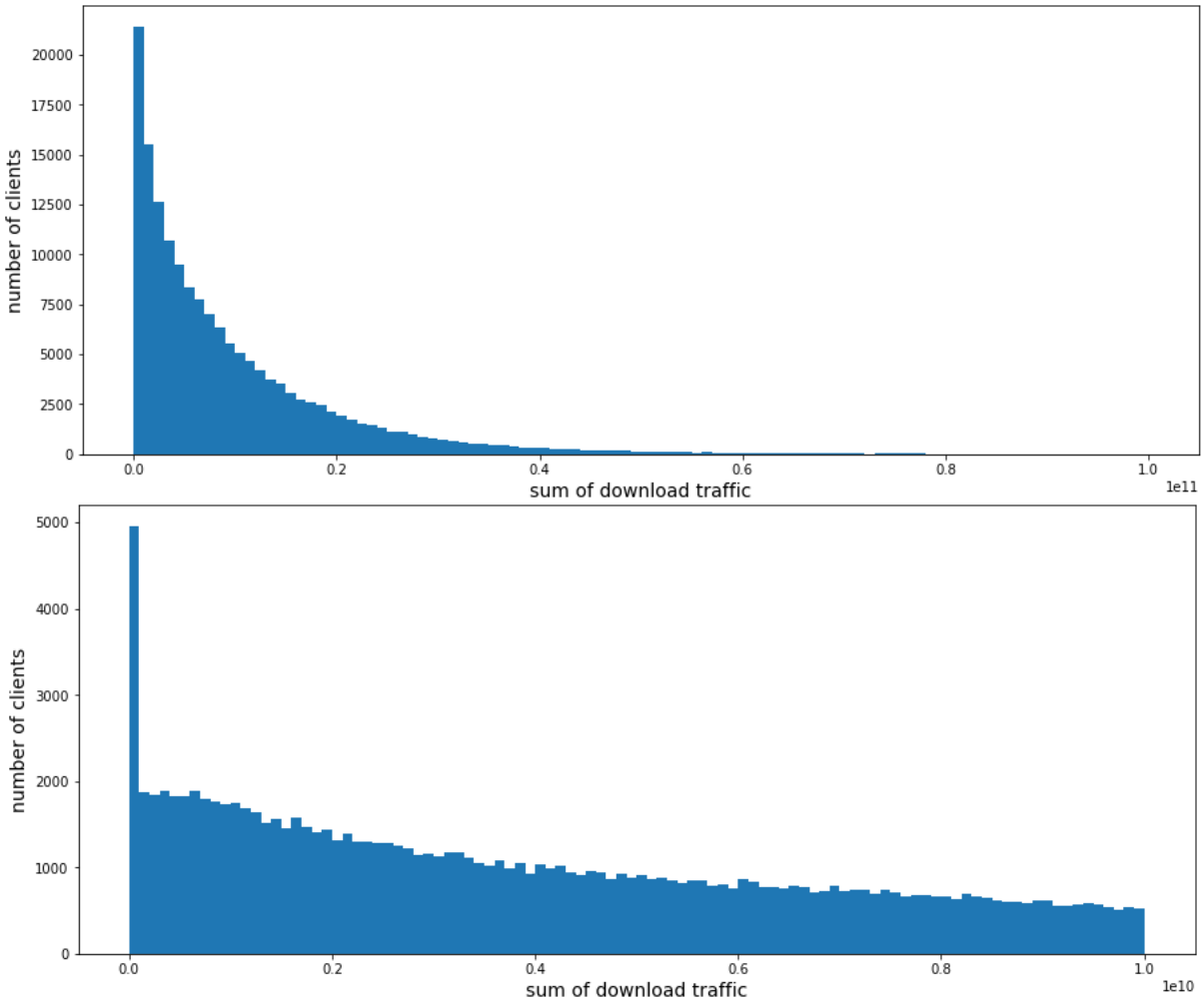


Figure 3.12: Distribution of clients in pain over 5 days given their use of Internet service (downstream). Filters for clients with monthly usage below 100Gb (top graphic). Filters for clients with monthly usage below 10Gb (bottom graphic).

that use less the service does not exist. One reason that may explain is the bytes the company exchanges with the devices to check on their condition. Nevertheless, if the customers did not use the service the peak distribution would not be around the 1Gb mark with higher tendency to increase.

Summarizing, for both services TV and Internet, regarding service use by customers signaled with degradation in their service we are confident that they use the service thus belonging to the group we want to address.

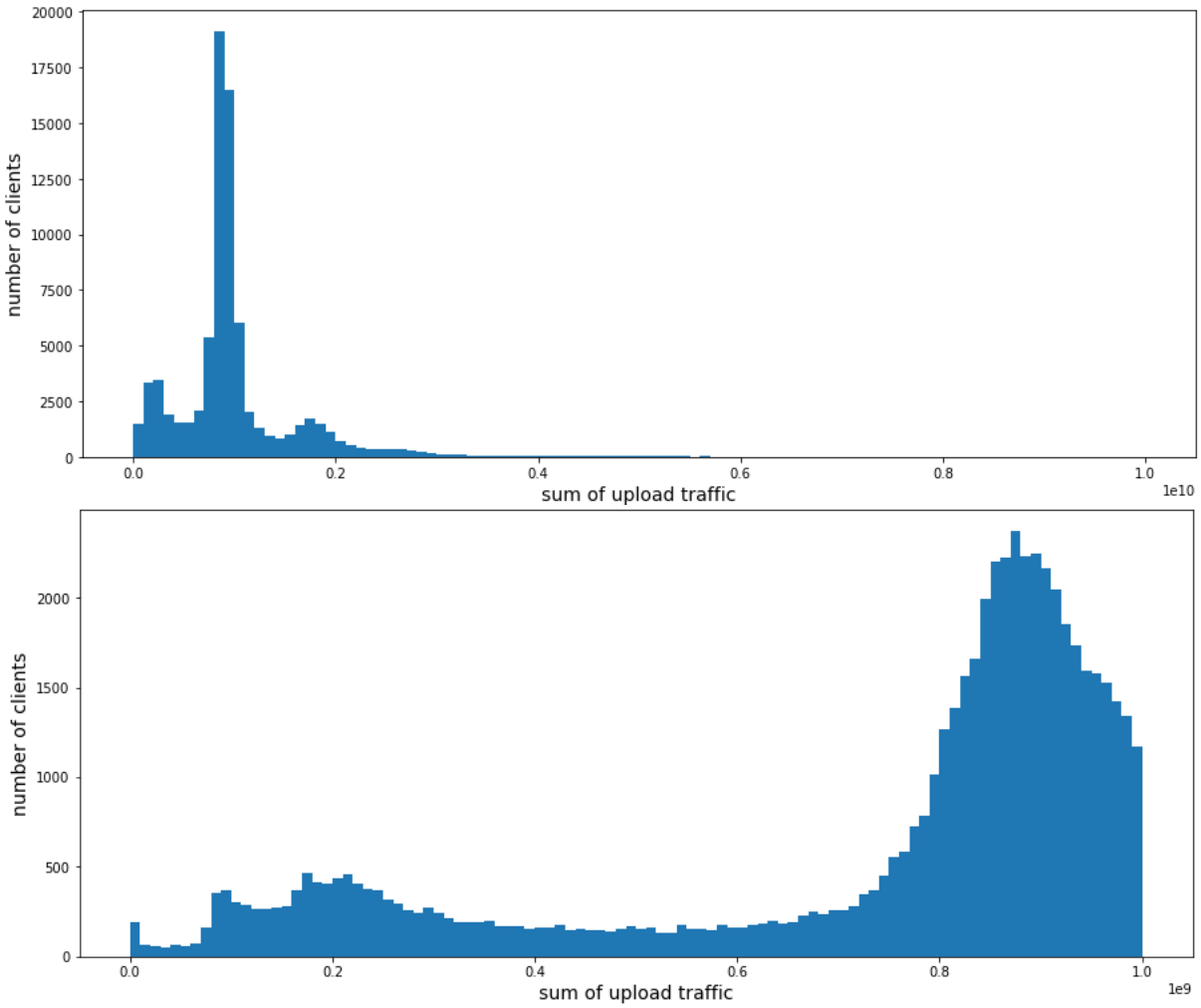


Figure 3.13: Distribution of clients in pain over 5 days given their use of TV service (upstream). Filters for clients with monthly usage below 10Gb (top graphic). Filters for clients with monthly usage below 1Gb (bottom graphic).

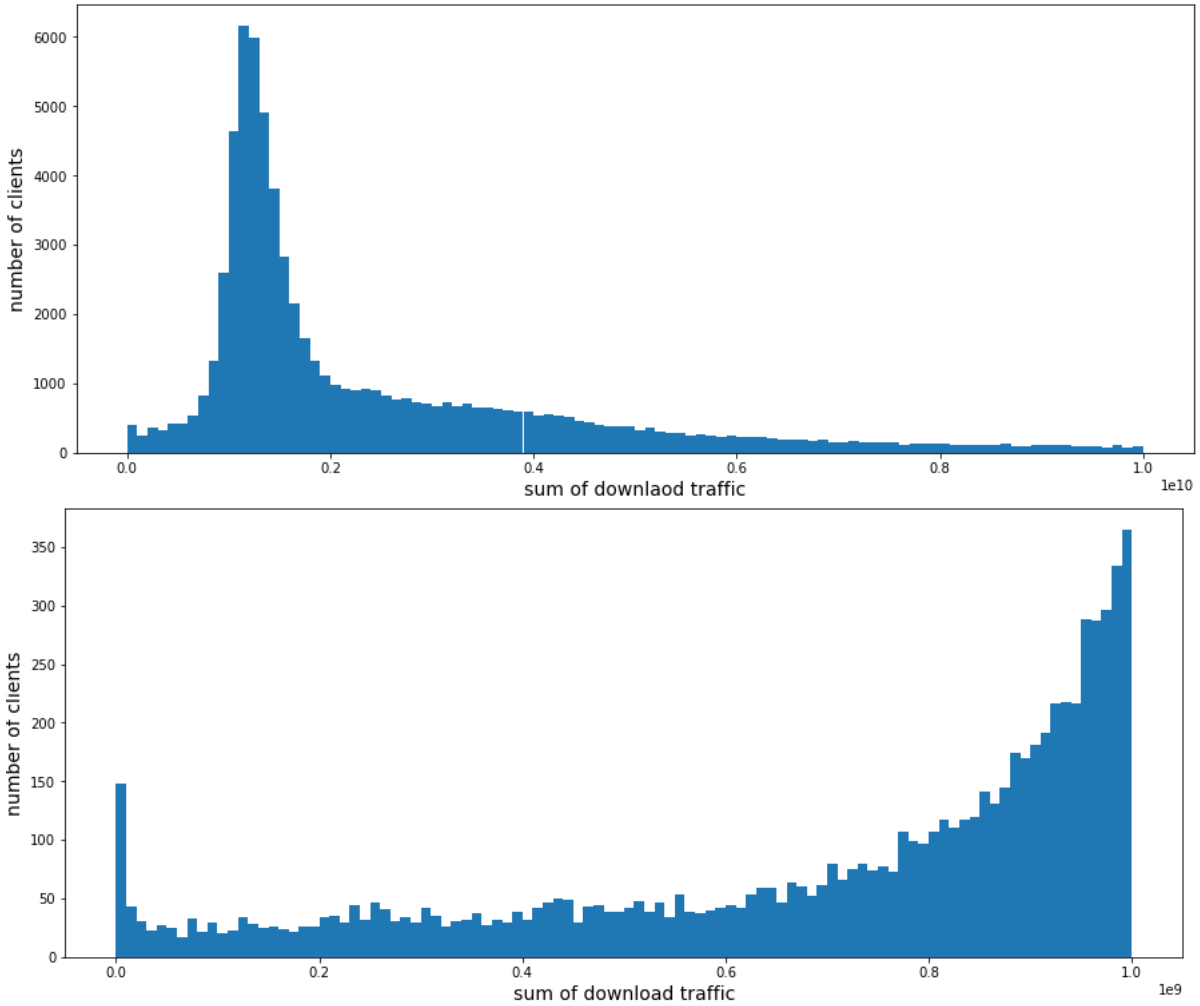


Figure 3.14: Distribution of clients in pain over 5 days given their use of TV service (downstream). Filters for clients with monthly usage below 10Gb (top graphic). Filters for clients with monthly usage below 1Gb (bottom graphic).

4

Customer QoS and QoE signaling: A Multi-task Neural Network approach

A real world data science project is a team challenge. To deploy an end-to-end solution the team needs to develop and maintain multiple blocks. Data needs to be ingested and customers issues addressed for the project to be worth. In Section 3.1 we describe in detail all blocks from the project.

The thesis goal is to provide a new model approach for the inference block, based on neural networks. Although current models use gradient boosting trees, due to special necessities in the architecture we opt by using neural networks. Further, we aim to successfully create a multi-task architecture to enable capturing the services intersection information. Multi-task architecture applied to the telecommunication customer experience is a novelty for the company and for the community. The multi-task model is considered an upgrade as it takes into consideration TV and Internet services together. Current models do not take advantage of the fact that HFC technology services share the same infrastructure.

To successfully create a solution we need to prepare the data, create architecture, train and test the model. In Chapter 4, we define each task in the end-to-end process. In the pre-processing step we clean the data to have less noisy, and normalize values to mitigate impact of variables that produce bigger values.

The thesis goal is to provide a new model approach for the inference block, based on neural networks. Although current models use gradient boosting trees, due to special necessities in the architecture we opt by using neural networks. Further, we aim to successfully create a multi-task architecture to enable capturing the services intersection information. Multi-task architecture applied to the telecommunication customer experience is a novelty for the company and for the community. The multi-task model is considered an upgrade as it takes into consideration TV and Internet services together. Current models do not take advantage of the fact that HFC technology services share the same infrastructure.

When completing all tasks, we have an end-to-end inference model. Analyzing which

model fits best the target is the next step, where the goal is to find the solution which can improve the operational performance.

4.1 Data Preprocessing

The thesis data available is very rich, as we see in Section 3.2. The environment is full of options and implicit information that we want to explore to produce a more complex and improved solution.

Having the opportunity to work with real-world KPIs gathered from the network and end-user equipment is a plus for any project. Internet and TV service structures are focused on delivering the service with quality, logging of the metrics is secondary. Due to the second priority in data gathering, punctually, there are problems with the data. Sometimes we have missing values, could happen that we have absurd values due to problems in the metrics retrieval, among others. Training our models in such a noisy environment is a challenge. Data preprocessing appears in this section as a solution to enhance data, improving models efficiency. Through preprocessing techniques problems such as imputing missing values, that reduce noise and inconsistencies in the data. Outliers in our problem context, have major importance. Although there are customers with problems, the majority of the clients have a good service. Thus, outliers help our models describe which variables thresholds indicate problems. However, one of our goals is to give the models more capacity to predict degradation problems. Outliers could difficult the characterization of degradation problems. Because they are so easy to detect the model can get stuck in learning only a tiny percentage of the cases, the percentage that maps to outliers problems.

To aid in implementation, but also to have a framework similar to the one used in the project, a configuration file was created and filled. Each set of values carry information regarding the decision for each variable. The software can read from this file and apply the different available options accordingly.

Fill missing values

Missing values is accepted inside the community has being a major problem for data science projects, as inserts unwanted uncertainty data. In the work [19], missing values defined as the absence of value for a variable, and also provides some examples of factors that could lead to this event such as the low signal-noise ratio between both communicating entities, non-responsive equipment, measurement error, or mishandling of data. It is easy to map the aforementioned situations to possible origins of missing values in this project data. Parts of the network could be saturated, end-user pieces of equipment can be having problems or the user just turned off the devices. There are multiple failure sources that can create missing data.

It is possible to find multiple approaches to handle this problem, from human imputation to usage of predictive algorithms to help choosing the best fit for each value, due to the complexity of the problem, not add more uncertainty. For each variable a constant value is assign to every missing value. Two types of constant were chosen:

- **Imputation of zero (0).** Binary targets and all KPIs variables for Internet and TV services that are not part of the imputation mean group belong to this one. Many of these variables have the power to solo impose a prediction which of course will create very bad results as the model will hold itself to only one variable. Zero acts as a neutral elemental which combats the problem. The targets are signaled by an action from the customer, if there is no positive signaling the directive is to mark it as a negative example.

- **Imputation of mean.** Variables imputed with mean mainly consist of variables that represent the maximum, mean, or standard deviation of reads made to each client. By definition, these variables are means or derivatives, and for that reason it is wise to impute the mean values, allowing to maintain consistency and avoid creating outliers or extreme values that in the normalization phase could shift wrongly the data. The process is more complex than just inserting a value into a gap. Impute function from the software library pyspark is called and for each variable, the mean is calculated for the training dataset and the result stored so that the same imputation can be applied to the evaluation dataset. At last, each missing value is filled with the mean value of that variable.

Data Normalization

Machine learning algorithms, more specific neural networks as the ones used in this project, heavily benefit from data normalization process as they improves accuracy and efficiency leading to better end results as well as lower training time [22].

Min-max normalization was the chosen technique to transform the data. This process consists of performing a linear transformation on the data, returning it compressed between a defined range (in this case between $[0, 1]$). The analysis is done on each variable singularly. However, the min value of a variable always takes the same value as the inferior interval value. Following the same logic, the max value for a variable is equal to the superior interval value. Therefore new min value and max value are set by the lower and upper bound of the designated interval. All values in between are then mapped according to a linear transformation. Equation (4.1) is used to perform min-max normalization.

$$t' = ((t - old_{(min)}/(old_{(max)} - old_{(min)}) \times (new_{(max)} - new_{(min)}) - new_{(min)} \tag{4.1}$$

The objective of normalization function is to calculate via a linear transformation a new value t' from the original value t. The original variable, as an associated minimum

value $old_{(min)}$, and an associated maximum value $old_{(max)}$. In the first part of the transformation, we are calculating the relative value of t in relation to minimum and maximum values of the variable. Then, we multiply the calculated relative value by the new interval to assign t its normalized value t'.

Important to state, our associated minimum value $old_{(min)}$ and an associated maximum value $old_{(max)}$ are not the real values in the variable. Because we aim to gain more capacity predicting degradation problems, our decision was to chop out the values in the first and last percentiles. Through this method we mitigate the influence of absurd values in the normalization process. The difference between values is smaller, thus the model does not have the stimulus to only learn the highly different examples.

4.2 Multi-task Neural Networks

From all the different groups of variables and variations possible, feeding the model with all the information is redundant. Our objective is to design a model capable of capturing the problem dynamics. In the first phase, we aim to react to each model individually. The following step is achieving a solution that takes advantage of the intersection of the service.

The project's main focus is on degradation problems. Thus, we filtered for variables that mapped variances in time. To ingest information regarding variance in behavior, we have two main groups: counts over a week or a month for bad events in metrics, and behavior variances from the day versus one week before or from one week versus one month (referential for day zero, is the day we are analyzing). For counts over a week or month, we are gathering absolute information about a variable. As in analysis done in Section 3.3, we force the neural network to decide which thresholds better define a customer with a propensity to do a Pt. In variances, the idea is to capture changes in consumption patterns. Decay in quality of variables over a month or even a week, could very well indicate the customer service is in degradation. Capturing degradation patterns helps us identify the pool of customers we are more eager to impact. Independently from the service, or if the variables are specific to one service or are shared by both, in general, the selected variables follow one of the two behavior characteristics.

Quality of communication variables and signal variables help with the explainability necessity of the project. Through these variables we can map our network, gathering information about the quality of service we are delivering, but also trigger events of disruption and degradation. State variables map the functioning of equipment inside the client's house. Although we need to be careful in their analysis because they could prove to deceive, sometimes when devices signal high use they are signaling is problems in the network. The high use of memory or CPU could map with the device's tentative to rectify issues in the network.

Of utmost importance are the group of variables that can somewhat map the clients' usage of the service leveraging the system's potential to integrate the concept of customer

satisfaction. From the QoE concepts presented in Section 2.1 we know that among other factors the perception of each client's quality of service heavily depends on usage, and so using usage variables as input is a step towards the goal of satisfaction alongside the metrics. Categorical variables have very informational traits and have high cardinally with mainly unique string values as they respect each customer and were not considered useful as input to the network.

Selected variables make an ensemble of 332 variables from both Internet and TV services but also customer information such as service usage.

Variables contain information about the system. However, the information is too encoded and is too difficult for a human to extract insights and knowledge. To extract the knowledge from the information we need to recur to machine learning processes. For the thesis, our choice was neural networks. From the many reasons that shifted the decision towards neural networks, the main was the flexibility architecture wise. The necessity to change between only one target for single target models and two targets in the intersection model looked at the decision to implement over neural networks.

For the intersection model, we implemented a novel approach in the telecommunication community. Multi-task learning allows for two targets to share the same network. By sharing the network, both targets are also sharing information. And, through the analysis in 3.3, we concluded that many of the clients signaled with problems in one service indeed complain about problems in the other.

We used the grid search method to try multiple architectures in all model developments. However, is important to state the optimization algorithm and cost function are the same throughout all experiences. The optimizer selected was the ADAM algorithm, while the cost function chosen was the Mean Square Error (MSE). The ADAM algorithm is responsible for weights optimization when the network propagates the error through the nodes to adjust. The ADAM is a stochastic optimizer that computes individual adaptive learning rates for different parameters [7]. In comparison to the stochastic gradient descent (SGD), the ADAM algorithm provides a better approach for problems with noisy data and high-dimensional spaces which relates to the thesis needs. The cost function MSE defines the variance of a model. The error value is the square of the difference between the pair, the prediction from the model and the real value of the example [21].

In practice, network architecture and training implementation is managed by the pytorch library. It was designed to help users easily create solutions based on their predefined features. Computation with pytorch allows parallelism and efficient coding, which improves the running time of the software. The MSE is computed in batches, however, the library calculates each pair (prediction, real value) individually. The final output from the function is the mean from all errors calculated.

Grid Search

Finding the best architecture fit and parameters is a challenge. Doing experiences by hand, and relating which combinations obtained the best results is inefficient and hard. The time to manually alter the architecture, and parameters and on top analyze the results is too costly. The solution we enforced is to create an automatic process that tries different combinations and stores the result so we can compare in the end. Each iteration of the grid search process produced a model with a unique combination of architecture and parameters.

Although there are some solutions for the grid search implementation, is difficult to apply them to our situation. Because we have a big component of real data with a huge volume, is important to manage the software that balances the logic between distributed systems and local computation. All the logic plus the effort to iterate over multiple instances consumes too much time. We opt by developing a simpler solution that specifically applies to our exercise.

First, we select what we want to change regarding architecture and parameters. In architecture wise, we decided to have two dynamic options: the number of hidden layers, and the size of hidden layers. The number of hidden layers respects how many layers we want in the hidden layer of the network. In a neural network architecture, we have normally one input layer and one output layer. The remainder layers belong to the hidden layer, and the parameter we are varying is exactly the quantity of layers between the input layer and the output layer. The other exchangeable parameter defines the number of neurons that each layer of the hidden layer has. Both architecture parameters influence the model dynamics. For instance, when the number of neurons raises the model gains more degrees of freedom. For the number of hidden layers, more layers mean the model can better design each target response path.

The other dynamic parameter is the learning rate. The learning rate does not map directly with the architecture, instead, its function is to control how the model converges to the local minimum. When we calculate the loss between our prediction and the real value, the model needs to readjust the weights between connections. Learning rate controls the adjustment process. Big learning rates values alter the weights quickly and aggressively. For small values, the adjustment is progressive and slow.

Production of models was incremental, implying that when a grid search was performed on one model type all the information given while analyzing the results would be processed for the next grid search performed in another model, thus increasing the probability of success in experiences. The variable parameters and the values each parameter can take are denominated as search space.

To summarize, we implemented a solution where we define the set of values for each of the dynamic parameters and test possible combinations. In the end, we are left with multiple models each one with unique characteristics and results we can analyze. Each of the models created, single target models and integrated model, have their exploration

done independently.

4.2.1 Single target models

The original premise is that current models are not capturing the intersection between Internet and TV services. From the analysis in 3.3 we stated that indeed the intersection exists, and is expected to be higher than the currently acknowledged. Exploring that information can help capture the dynamics of the system, thus impacting the model solution.

The argument in the thesis is that by using an integrated multi-task architecture model we can explore the intersection between the services and produce a better result. However, to validate that our model with multi-task is better and can explore the intersection between services, we have to have a comparison baseline. Comparing the solution versus the current models in operation would not be fair. Current models have years of maturation, with fine-tuning of variables for certain use cases, and even specific corrections to the models themselves to better adjust to the problem. Adjusting our models to the environment is a normal process in any project. With time needs appear and must be resolved, so engineers continuously make improvements to the solutions.

With current models out of scope, we need to develop replicas of the current models, to be the baseline for comparison. For the thesis, and as an engineering project, building a solution from the simplest to the more complex is perfectly normal and acceptable. By creating models for both single targets we gain more knowledge and insights about the problem. Also, we made sure building a solution via a neural network is valid, and that we could obtain results in line with the ones in operation. We believe the incremental process of constructing models for each service independently and after for the integrated solution propelled the work for better results.

Model solutions for Pt Internet target and Pt TV target are identical. The set of variables is the same because we wanted to give the models the capacity to embed information from the other service. In single target models, we want also to test how a model trained to respond to a specific service can take advantage of the intersection of the service. The neural network developed is a fully connected network with an input layer of 332 neurons(one per variable) and an output layer with a single neuron. We can approximate the output neuron to a computational unit that separates the space in two via a linear function. In figure 4.1, we demonstrate a hypothetical scenario of the frontier and both positive and negative sides. Our target is classifying each client with the confidence of making a Pt. Because we are using a relu function, as further as the client is from the frontier on the positive side more confidence we have that he will make a complaint. Closer to the frontier and negative examples, represent clients with good service experience.

The main difference in the single target exercise is the target. For Internet service we use has positive examples the pool of clients that complained about Internet symptoms and for TV the opposite, clients that complain about TV symptoms. Yet, from tables

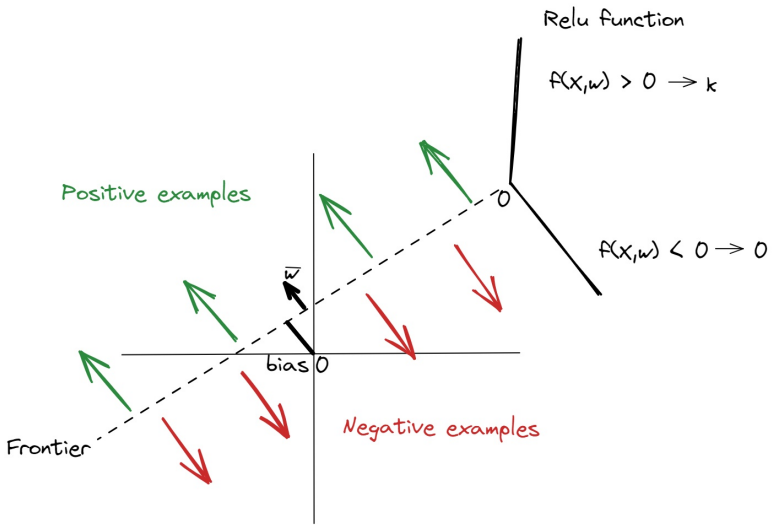


Figure 4.1: Hypothetical neuron decision space. Neuron activation function is ReLU.

containing symptoms and resolutions, by feeding the whole information we should get better results.

The architecture implemented for single target models is a fully connected neural network. Supports an entry neuron for each variable, from TV and Internet. In figure 4.2, we can visualize the three main components of the neural network architecture: input layer, hidden layer, and output layer.

To generate the best model fit possible we use the grid search method. For both single targets we defined search space. The input layer and output layer are constant, yet, the hidden layer suffers alterations through the combination given by the number of hidden layers and the size of hidden layers. The hidden layers structure is dynamic and allows different approaches to the problem. Varying the learning rate and the hidden layer conditions can be described as trying multiple learning techniques.

The implemented network does not have any breakthroughs or complex science associated. Is a simple neural network, that tries to learn each target individually. In the architecture illustrated in figure 4.2, we can vary the parameters from grid search, but we cannot force the model to learn both targets simultaneously. The major limitation of single model architecture is that despite feeding variables from both services, we can only test one target at each time. For example, if we are training the model to capture Pt Internet target dynamics we can feed the TV service variables, but not the clients that complained about TV service. Thus, we can share some TV service information, however, the Internet target model does not have the information on which set of variables identifies a complaint about TV service. What the explanation means is that there is information that is impossible to learn due to the architecture. Single target architecture has design limitations regarding sharing of information between two or more targets. For that reason, we developed a multi-task neural network.

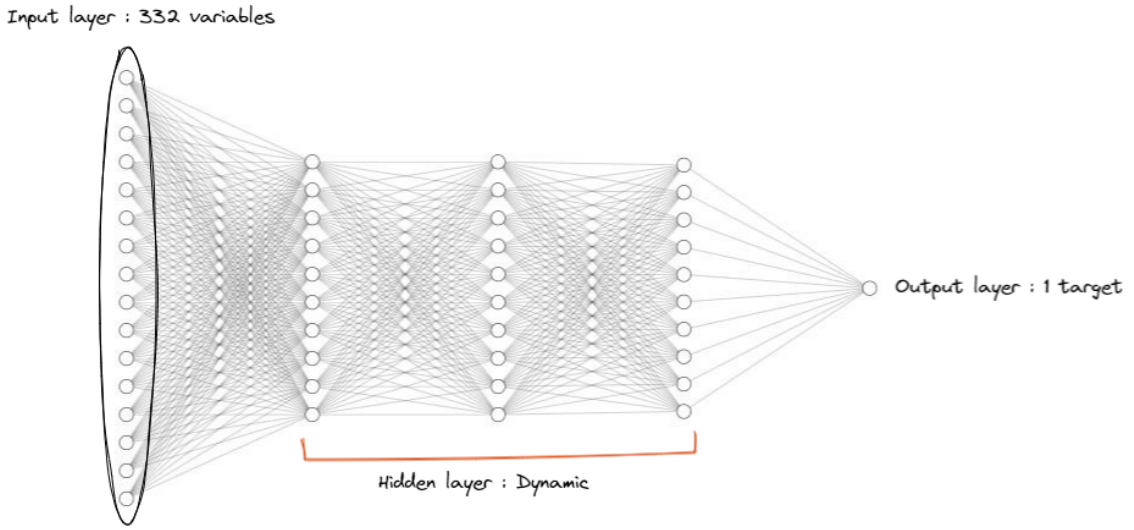


Figure 4.2: Single target fully connected linear neural network architecture.

4.2.2 Multi-task model: Pt TV and Pt Internet

The integrated model is the solution proposed in the thesis to address the problem of exploring the intersection between the services. Through the integrated model, we aim to prove the intersection between services is more than currently felt, and that through a multi-task neural network we can extract that information as knowledge.

Prediction wise the objective stays the same. We want to train the model to learn the probability of a client making a Pt. In theory, the targets are the same, Pt Internet target and Pt TV target. However, for the integrated model we learn both targets together instead of independently. In practice, the objective is to force a unique architecture to learn simultaneously what makes a customer complain about Internet and TV.

Implementing successfully a multi-task architecture in the thesis context unlocks multiple discussions and knowledge. From the company point of view, we validate that services have hidden information in their intersection which forces teams to rethink their approaches. Also, we introduce to the company a solution that works on real data and increases operation results. For the telecommunication community, we add a novel solution that improves the model's capacity to learn difficult targets. In telecommunication environments the data is noisy, and sometimes getting a clean dataset to train the model is a nearly impossible task. With our work, we aim to prove that we can use correlated targets to improve our knowledge, thus our capacity to predict an event.

The multi-task learning concept comes from the computer vision field. The idea is that if two targets are correlated they share features. If they share features, implicitly when we learn one target we are also learning the other, and vice-versa. That concept of implicit learning is exactly what we aspire to with the thesis. We want to create a model that can learn implicit information that exists in the intersection between the Pt Internet and Pt TV targets.

As we proved through the analysis in tables 3.2 and 3.4, the proportion of clients that have problems in one service and have symptoms in the other is not negligible. There is a lot of implicit information in the data that an human cannot see or explore. Through the multi-task model we expect to signal more customers, more use cases and improve the efficiency in the process.

Implementing multi-task architecture is very similar to single target models. We need to understand that the major upside in multi-task is not in the complexity of the network. The secret is in the theoretical knowledge the multi-task approach provides. In figure 4.3 is presented the architecture for the integrated model. We maintain the input layer constant with 332 neurons, one for each variable, and have a dynamic hidden layer to test multiple models with the grid search method. The difference is in the output layer, where we evolve from one target to the possibility of having multiple targets. In the thesis case two targets, Pt Internet and Pt TV.

Adding another neuron in the output layer seems like an irrelevant alteration, however it changes all network dynamics and complexity. To be clear our exercise is not multi-class where we have multiple choices and we predict the most probable. In a multi-class exercise, the output space is disjoint where the model assigns to each different available answer a probability. The sum of the probabilities must be one, which denies the hypothesis of having both TV and Internet targets positive which is false in our use case. The thesis exercise is multi-task, which means we have two binary targets being learned simultaneously. With two targets the network has to adapt the weights in the connections to capture simultaneously the dynamics of two events.

Each output neuron is responsible for predicting only one target, meaning one neuron adjusts its weights to learn how to signal a client with the propensity to complain given an Internet symptom and the other the same but for TV. The output neurons' comportment is the same as the described in figure 4.1. Each neuron's job is to decide how confident they are the client is going to make a complaint about the service. Although their functioning is the same, the information they receive is very different. If our model works properly, the information that reaches the output layer is much better because we enrich it with implicit knowledge. For the integrated model the output neurons not only have available the information regarding their target but also have the details from the other target. For cases where the symptoms do not match the metrics in failure and we could not signal the client, our solution can address it. Because we intersect the information of bad values in the variables with the complaint in the target. To exemplify, we propagate all variables values from Internet and TV services. If a client has bad TV metrics but complains about the internet, with this architecture we can learn that valuable information. In the integrated model, we cross knowledge from the services.

For the model to learn how to characterize a target, as we know, the computed error is backpropagated through the network which forces the neurons to adjust. The process that allows the model to learn both targets simultaneously is the backpropagation of the error. After the calculus of each target error independently, we sum both and create the

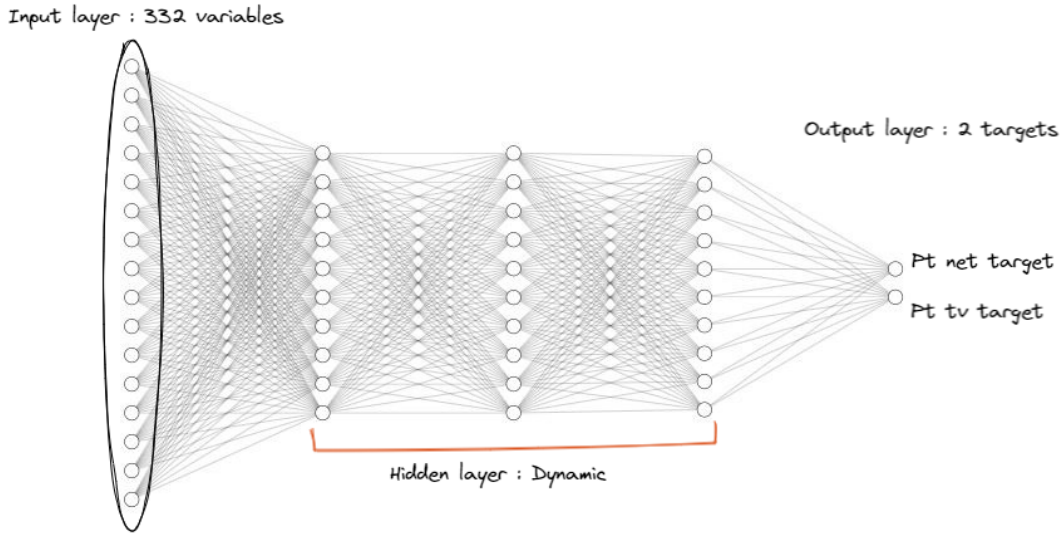


Figure 4.3: Multi-task fully connected linear neural network architecture. Hard-sharing architecture with two targets.

integrated error. The integrated error contains information about each target and also the implicit relations. When the neurons in the network adjust, they are learning over global information that has both Pt TV and Internet targets and also de implicit insights.

Hard sharing approach results in targets sharing their entire architecture. In figure 4.3 is possible to observe that every neuron contributes to the learning of targets. No neuron or no layer is specifically assigned to learn the characteristics of only one target. With a hard-sharing network, we do implement an exclusive block of neurons specialized in learning the information for a specific target. Other techniques can be found in Section 2.2.

Comparatively, with single target neural networks, a multi-task solution resolves the limitation of learning only a target by turn. The problem we could not resolve of crossing metrics data and symptoms felt by the customer is resolved in the integrated model.

The multi-task solution explores targets information intersection through architecture, more precisely weight sharing. Encoding the problem and capturing the system dynamics is the task of the hidden layers. Neurons receive feedback via the cost function and adjust their weights to better predict the target, in the multi-task case for both targets. In figure 4.4 is a visual interpretation of the neuron's functioning in the hidden layer. Because they have to encode information for two targets, by separating positive examples from negative examples regarding one target implicitly the neuron is also learning how to distinguish the other target information. This only happens because there is an intersection between the targets. In a hard-sharing network approach, every neuron needs to create a frontier that can separate positive examples from negative examples simultaneously for both targets. By doing so immediately information needs to be shared as some spaces will have the intersection between targets examples.

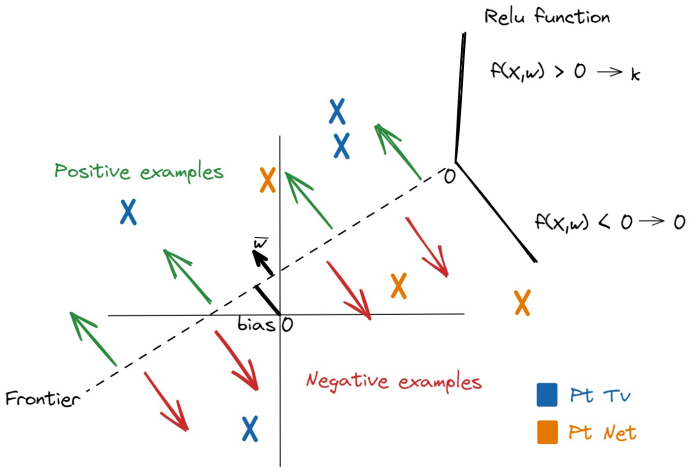


Figure 4.4: Interpretation of a hidden layer neuron working.

4.3 Training with Big Data

Available data in the problem context can be categorized as Big Data as it respects three of the more important V's: Velocity, Variety, and Volume. To work with big data we need distributed systems capable of taking advantage of parallel processing. Framework Hadoop where tools like Spark and Hive exist do so.

Current models in the project leverage solutions based on pyspark. Because data is stored in cluster and retrieved using distributed systems mechanisms, it's advisable to leverage implementations that natively interact with the system that retrieves data. It is not only efficient development wise but also allows to improve spatial and time resources. Such tools also can take advantage of distribute systems and spread the job between multiple workers increasing the amount of data processed at each time.

In the real world, there are limitations, and the available libraries in the cluster environment for pyspark mllib do not support neural networks. No classification algorithm in the library is sufficiently exchangeable to support the multi-task exercise. Given the need to use neural networks to implement the multi-task architecture, the decision was to develop the project using a trendy neural network library pytorch. To use pytorch, pyspark objects where data was initially stored do not work. The pytorch library cannot handle distributed objects. To feed the data to the models, we needed to collect the data from the cluster and store it locally in the docker to enable batch processing. Without dividing the data into batches, the docker would not have sufficient memory or computing power to support the exercise. Training big data in a local environment surfaced multiple challenges regarding running time and spatial resources. Hence, some software solutions were designed to tackle those challenges.

Batching

Data is initially retrieved from the cluster and stored in a pyspark dataframe object. In the cluster, data is divided between logic partitions. Logic partitions prevent data to be stored altogether. Through the partition key data is scattered to the multiple nodes., thus avoiding bottlenecks when accessing the servers to do read operations. Project data is partitioned by day as it provides good cardinality avoiding hot and cold partitions.

Each day has around one million entries, a quantity that is impossible to manage in the local environment due to insufficient storage. To address the issue each day was divided into smaller chunks of data denominated as batches. The goal is to spread randomly the data amongst the different batches, with each batch having an identical number of examples. By randomly selecting which entry goes where we could keep the original data distributions. Documented in annex I is the software developed to take each day stored in a pyspark dataframe object and add a column for batch indexing. Each example from the dataframe was assigned to a batch.

Batching has a direct influence on the management of spatial and time resources. As the developer, the decision of how many examples go to each batch is ours. In the first training approach selecting the batch size highly influenced the training speed and model performance. Large batch sizes, like dividing the day into two parts put tremendous effort into local docker storage. Data stored in spark objects is compressed, so when decompressed to the local environment takes up much more storage resources.

On the other hand, having the data broken in too many chunks deteriorates training running time. Each time a batch is collected from cluster to local environment and processed adds 100 seconds. For 10 batches to training, time adds approximately 25 minutes, versus 4 batches that are retrieved in approximately 5 minutes. We opt by using 4 batches as when training for multiple days the overhead time was significantly higher. By filtering the best batch size, we achieved better performances regarding running time. Also is compatible with docker storage capacities, as we could add more days to the training and thus feed more examples to the model, which improves learning success probabilities.

Dataset Cohorting/Balancing

The constructed dataset has a problem regarding target fairness, derived from the disparity between quantity of positive examples, about 0.1% of the total, and quantity of negative examples that are the remaining 99.9%. This situation described represents an unbalanced dataset. Unbalanced datasets prove very difficult to train due to the need for huge volumes of data in order for the network to train. We have access to sufficient volumes of data, however, the problem in this case is the impossible time that such exercise would take to train.

Balance the dataset is done by gathering only positive examples from the target or targets (multi-task) and storing them separately in a dataframe. The developed software

allows the user to specify the number of negative examples he wants to extract. Randomly extracting a sample of negative examples is then stored in a dataframe also. Now with target(s) positive and sampled negative examples, a simple append of one dataframe to the other is sufficient to build the training dataset for that batch. Code and functions developed to achieve the balancing is present in annex II.

Reducing the data feed to the network prevented noisy data from having too much influence. Also improved training running speed which allowed to feed the model more days and therefore more positive examples.

After each day is considered a dataframe. We added a column to each day's dataframe to index each row to a unique batch. When is the catching process we just need to filter by the batch we want and obtain a subset of the dataframe. After, each dataframe is stored in a dictionary so it can easily be accessed. In the early development phases, every iteration in the training process had too much overhead time which hurt the model performance. To complete training of one day we needed to loop through all batches, collect them from the cluster, transform a pyspark dataframe object to a pandas dataframe object, capable of operating with pytorch, and balance the dataset. To optimize the process, balancing software was developed to work in the local environment. At least the overhead of the process of balancing the dataset was 100 seconds.

All the functions and developments done left us with a small/medium dataset. Now as the information is manageable in the local environment we could store it in an excel file. By doing so the computation of taking a batch in the cluster, converting it to a pandas dataframe, and storing it in excel could be done offline. Now in the training process, only an operation of retrieving a file from local file storage is needed improving immensely running time and thus more positive examples could be trained.

5

Performance Evaluation

To develop an end to end solution for the data science project many blocks need to be implemented and then chained together. In Section 3.1 it is introduced the project framework and each block responsibility towards project operation. Main contribution of the thesis is in the inference block. To develop the training part of an inference solution we need to choose which variables contain more useful information to characterize the targets, select an architecture and build a training mechanism. Aforementioned steps towards the inference development are documented in Chapter 4.

However, training a model is only a part of the inference process, to validate the model's ability to predict the target an evaluation has to take place. It is through evaluation that we can compare which architectures best fit the learning of the target. During grid search process multiple combinations of architectures and hyperparameters are tested. The only way to differentiate the best models from the rest is through analysis of the evaluation metric. By evaluating performance we are testing the model capacity to respond to unseen examples, thus validating that the model can be deployed in unlabeled data and perform.

In the thesis context data is collected each day, we perform the training using days from one month, while the days selected for the evaluation period are from a couple of months after. This way we are evaluating the model in different data conditions from training.

It is very important to evaluate the model performance in order to be confident with the product we deploy in operation. When in operation, there are associated costs on contacting and resolving customers' problems. Having a solution capable of bringing back return is of utmost importance. Through a stochastic dispersion metric, Lift, we are able to evaluate the model's ability to efficiently predict correctly positive examples. No good comes from a model that predicts everything positive or correctly guesses almost every positive example, but at the same time, predicts incorrectly many negative examples as positive. Another important performance indicator is how good the model is signaling customers which problems can be resolved. There are defined use cases and we mostly want to operate over clients that belong to one of the use cases. It is out of the scope for

the thesis, nonetheless, an open door stays open for further research and development regarding the characterization of clients signaled by the model.

All created models, that is both single target models and integrated model require an analysis. The performance results of each architecture deployed in grid search computation must be compared. To learn new insights, one should do an all against all comparison between each target architecture. Furthermore, a comparison of architectures performance from integrated targets model versus the performance achieved for single models must be analyzed. Successful comparison between integrated model and single models proves the intersection between models. Also, the characterization sustains a novel technique approach for telecommunication problems.

When analyzing lift curves points such as overfit and capacity to predict more efficiently in higher percentiles will be valued. Architectures whose results are random, thus having presenting random performances will be discarded from the analysis.

5.1 Lift

Evaluating models through lift measure business-wise brings a couple of advantages. Current models use lift as their performance measure, so implementing the same reasoning in the thesis enhances compatibility within the project. In the company, lift measure is popular and coworkers are used to it. Analysis of lift curves becomes much more intuitive and easier to explain as the project owners already understand how to interpret it. When showing the results, the objective is to prove a certain architecture and learning rate trained with a set of input variables are the best option to characterize the target. Thus, finding which model corresponds to the best explored fit.

Lift metric supports the decision of selecting one model and not the other. Project operation aims to contact as many customers with problems as possible. However, the resources are limited, so it is more cost efficient if every customer we signal as having problems indeed has problems. To prove one solution brings more return on investment than the other, the evaluation made has to indicate that the probability to signal the right customer is higher. As aforementioned, other performance factors have to be considered. Still, a model with higher lift performance should be a front runner as the best available solution.

In all scenarios, performance was evaluated through the same evaluation metric, the lift. It is a way to compare how good a solution is versus other types of solution and be confident that the analysis is valid. For comparison of different scenarios experiences, lift measure will be the main support information.

For the project reality, the number of potential customers is hard to calculate, we can assert is unknown. Therefore express true positive rate or ROC curve is nearly unfeasible. The idea is to vary the threshold of a probabilistic classifier to obtain a different set of points and compare it to the classifier [17]. From the process, we can measure how better

is the classifier against the probabilistic classifier, in more objective terms how good is our model versus a random prediction.

Equation (5.1) presents the lift performance indicator. The denominator represents the quantity of instances from the evaluation set. The numerator is split between the probability of correctly identifying a positive target and the times we predict the instance as positive and the reality was that the instance was negative. The result, $Y_{rate}(t)$, measures how good is the model predicting positive examples given the entire set. Even if there are multiple positive examples predicted as negatives that is not as bad as predicting negative examples as positives, ratio given by $FP(t)$. Because contacting the client is costly our goal is to be as assertive as possible signaling which clients do have problems.

$$x = Y_{rate}(t) = (TP(t) + FP(t))/(P + N) \quad , with \quad y = TP(t) \tag{5.1}$$

Lift evaluation metric can be evaluated from both results of a binary classifier, however it can only act upon one at the time.

5.2 Characterization of the performance achieved for the considered scenarios

Analyzing the result of an evaluation is not as simple as looking at the lift curve and stating the obvious, which model has the higher lift. To measure the quality of lift performance that each model yields we need to read the implicit information represented in lift curve. From figure 4.1 and the relu function used in the output nodes, each example is predicted with a value representing the level of confidence. Higher value represents higher confidence that the example is positive. For instance, if example A gets a prediction of a value 0.90 and example B with 0.10, means the model is 9 times more confident that example A is indeed a positive example. The distance to the frontier marks the confidence percentile to which each example is assigned.

All examples, positive and negative, contained in a day are feed to the model. Each one is predicted with a value of confidence. The space has 100 buckets, each one representing a percentile from 1 to 100. To each bucket is assigned the same number of examples. The methodology implemented to spread the examples throughout the buckets is by the level of confidence. If the capacity of each bucket is n, bucket represented by percentile 100 receives the top n examples where the model is more confident in its prediction. The remaining buckets are filled following the same reasoning.

Aggregate examples by confidence in prediction allow analyzing how good the model is for each percentile. Due to resource allocation, in operation it is impossible to contact all customers. So the clear-cut decision is to prioritize customers signaled with higher confidence, hence higher percentiles. Given the operational side of the project, it is critical that the model is efficient in selecting customers with high confidence of having problems.

One aspect we search for in lift curves is how does the model behaves in top percentiles. High scores in top percentiles and also stability in the curve are appreciated.

Evaluated scenarios are the performance of both single models, analyzed individually, and also the integrated model. Computation time and storage capacity will not be contemplated for the performance analysis. Since the architectures explored have identical computation time, in the same order of magnitude, does not provide influencing information for conclusions.

5.2.1 Performance analysis scenarios

The mission for single targets was to model each target, Pt TV and Pt Internet, individually and obtain valid results to use as benchmark for comparison with the integrated model. Results from current models cannot be provided in the thesis due to privacy rules violations. Yet, single target modeling was only considered successful when results were in line with the ones in current models. Evaluation of lift performance was the main deciding criteria on choosing which models from grid search exploration best capture target dynamics of each target.

Pt Internet target

First target subjected to exploration was Pt Internet. There is no real reason for the order, but as data in Pt TV target was having some problems starting with the model for Pt Internet target seem like a straight-forward decision. Exploratory process was accomplished via grid search methodology. Figure 5.1 defines the search space for Pt Internet target. The search space defines all the architecture parameters and learning rate options available. All models created for Pt Internet target are born from a combination of every variables parameter in the search space.

```
Net Target Grid Search Space:
{

    Train Days: [20210126, 20210119, 20201211, 20201216]

    Evaluation Days: [20210318]

    Number of hidden layers: [3, 4, 7, 10]

    Size of each hidden layer: [150, 100, 50]

    Learning Rate: [1e^-1, 1e^-3, 1e^-4, 1e^-6]

}
```

Figure 5.1: Internet target grid search space.

After grid search completion, for each model created we save the associated MSE training error and associated MSE evaluation error. With both errors, we can inspect

for patterns in results. Thus understanding which changes in parameters best fit the target. Figure 5.2 shows the distribution of models by the associated error for train and evaluation. Legend of figure assigns a color to each possible parameter option. Through colors, we can differentiate which model has what parameter. Then together with lift curve, we can establish insights.

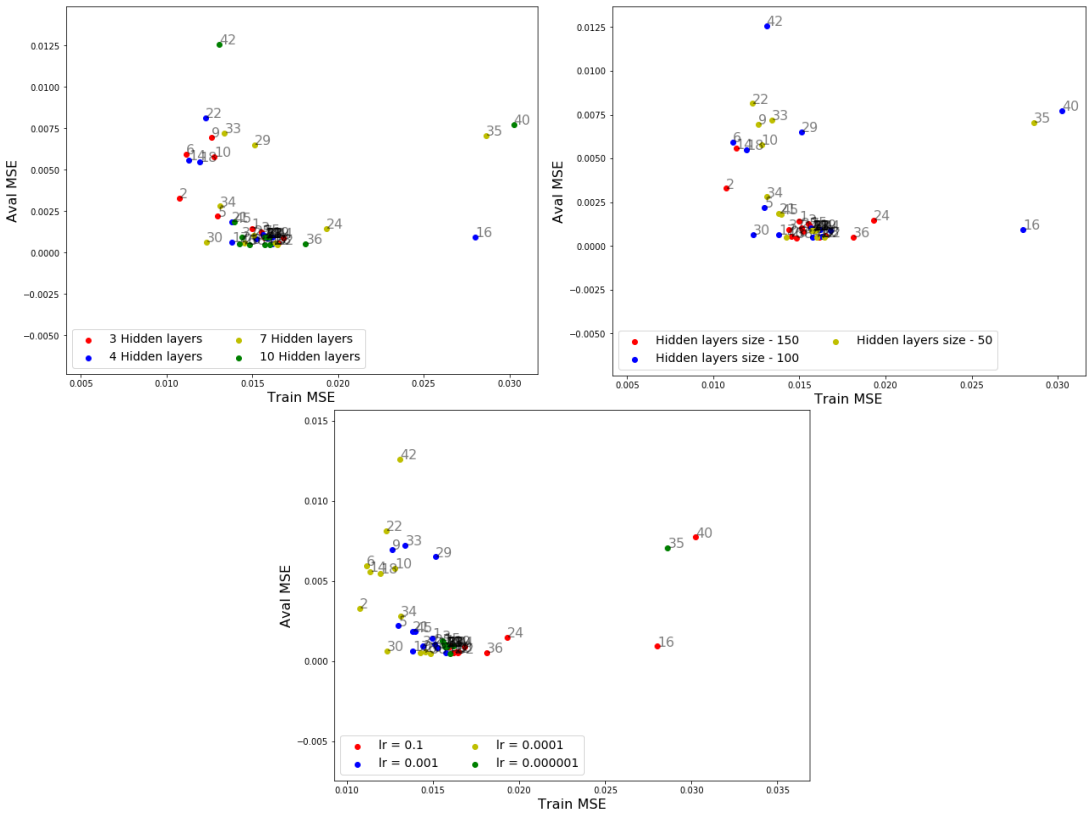


Figure 5.2: Internet grid search analysis.

Before analyzing the lift results from the evaluation of the model, not many insights can be extracted from scatter plots. Yet, one detail should stand out when looking for learning rate variable in the scatter plot. We can see models with the same learning rate seemingly aggregate in the same area. This means learning rate is a very influential hyperparameter that can change drastically the obtained results. From this point tuning of learning rate is of higher priority.

Through combination of all exchangeable parameters given in figure 5.1, for Pt Internet grid search the output was 48 models. Lift curves were inspected manually to map which models could better capture target characteristics, which were not as good because of overfitting, and the ones outputting random predictions. Connecting information from models evaluation through lift curve and scatter plots defined 3 clusters. Each cluster represents a level of performance.

Cluster 1 is composed by the group of models whose associated MSE training error is in the interval of [0.010;0.015] and the associated MSE evaluation error is between

[0.0050;0.0100]. Models in cluster 1 present the higher lifts and lowest overfiting. Those characteristics makes them the best candidates to fit the target. Some insights to retrieve is that the learning rate which provides better results is in the middle. Too small and too large values tend to decrease model capacity to learn the target. Although we can spot a tendency for better results with smaller layer sizes is not enough to take powerful conclusions.

Now, cluster 2 is in the border between what we can call as good and bad. Spatially delimited by associated MSE training error in [0.010 ; 0.015] and associated MSE evaluation error in [0.0000 ; 0.0050] (never reaching zero), models in cluster 2 are competitive in relation to cluster 1. However, in top percentiles are characterized by a tendency to overfit. We can define this group as the limit of good results, with decay in performance for the models with higher associated MSE training error. Those almost mix with cluster 3. Once more we see learning rate with value in (lr = 0.001 and lr = 0.0001) correlate with models with higher performance. All models not contemplated in clusters display random predictions, so they are not relevant to the study.

Last, cluster 3, which is defined by a big agglomerate of models contains the worst results. Towards higher associated MSE training error predictions are random. Near cluster 2 we spot acceptable lift curves but with high overfit, which makes the models not viable. Also of notice, learning rates in cluster 3 mainly consist on values in (lr = 0.1 and lr = 0.000001), thus confirming learning rates either too high or too low present the worst performance results.

One case to notice is that smaller associated MSE evaluation error does not provide better results. For Pt Internet and Pt TV target scenarios, it is noticeable that some models have poorer performance even with lesser evaluation errors. An explanation for the unexpected event is the imbalance of the evaluation dataset. The model learns to predict values closer to the mean, which is closer to zero. By doing so, produces way more accurate results as the majority of examples are zero, representing non Pt clients. Such a situation causes more overfit in the top percentiles.

Keeping in mind the objective is to predict as best as possible positive examples without confusing negative examples with positive ones, it is very important that the model can predict assertively. Especially in top percentiles, it is critical that the model predicts as positive only the examples that are in reality positive. Achieving the level of confidence needed, we are able to operationalize the results obtained from the model.

Pt TV target

Towards finding a model good enough to be used as benchmark for the thesis, we must try different architectures. From problem knowledge, Pt TV target is more complicated to characterize. Results via lift curve can seem lesser when compared to Pt Internet target. However, the goal is for the obtained results to be in line with the ones achieved by current models. Pt TV target also is tested via grid search method, and the following

dictionary structure representation in figure 5.3 shows the search space defined for the experience.

```
Tv Target Grid Search Space:
{

    Train Days: [20210126, 20210119, 20201211, 20201216]

    Evaluation Days: [20210318]

    Number of hidden layers: [1, 3, 5, 8]

    Size of each hidden layer: [150, 100, 80, 50]

    Learning Rate: [1e^-1, 1e^-3, 1e^-6]

}
```

Figure 5.3: TV target grid search space.

It is important to highlight the decision of maintaining the learning rate for both big values(lr = 0.1) and small values (lr = 0.000001), with only one medium value in (lr = 0.001). The idea is to test if the learning rate parameter has the same effect in Pt TV target as it had for the Pt Internet. For the number of hidden layers, the max value was lowered because results obtained with 10 hidden layers were not good. Values for the number of hidden layers were lowered down to accommodate insights gathered. Interpretation of the dictionary structure is the same as in Pt Internet target.

Similar to the experience documented above for Pt Internet target, using the grid search technique we constructed 48 models from the multiple combinations available. Figure 5.4, represents models distribution given their performance in train and in evaluation. Axis x, horizontal, describes each model behavior in training through associated MSE training error. On the other axis, y in the vertical points are associated to the MSE evaluation error achieved by each model. Colors are described in the legend and help in the evaluation process.

Once again, we can cluster models by learning rate. Learning rate confirms to be a deciding factor in model performance.

We defined only two clusters this time. Cluster 1 represents medium learning rate, (lr = 0.001), and cluster 2 represents the big agglomerate of models between the interval [0.050 ; 0.075] in the vertical axis, correspondent to the associated MSE evaluation error. Model 10 does not belong to any of the clusters. The characteristics of model 10 do not suit the representation of either cluster. All other models produced random results, which makes them unfit to proceed.

The learning rate lr = 0.001, provides the models with the best results. With high lifts and a low tendency for overfit, checks all the characteristics we aim for achieving a good model. It is in cluster 1 that we can find models in line with current models. Cluster 2 is

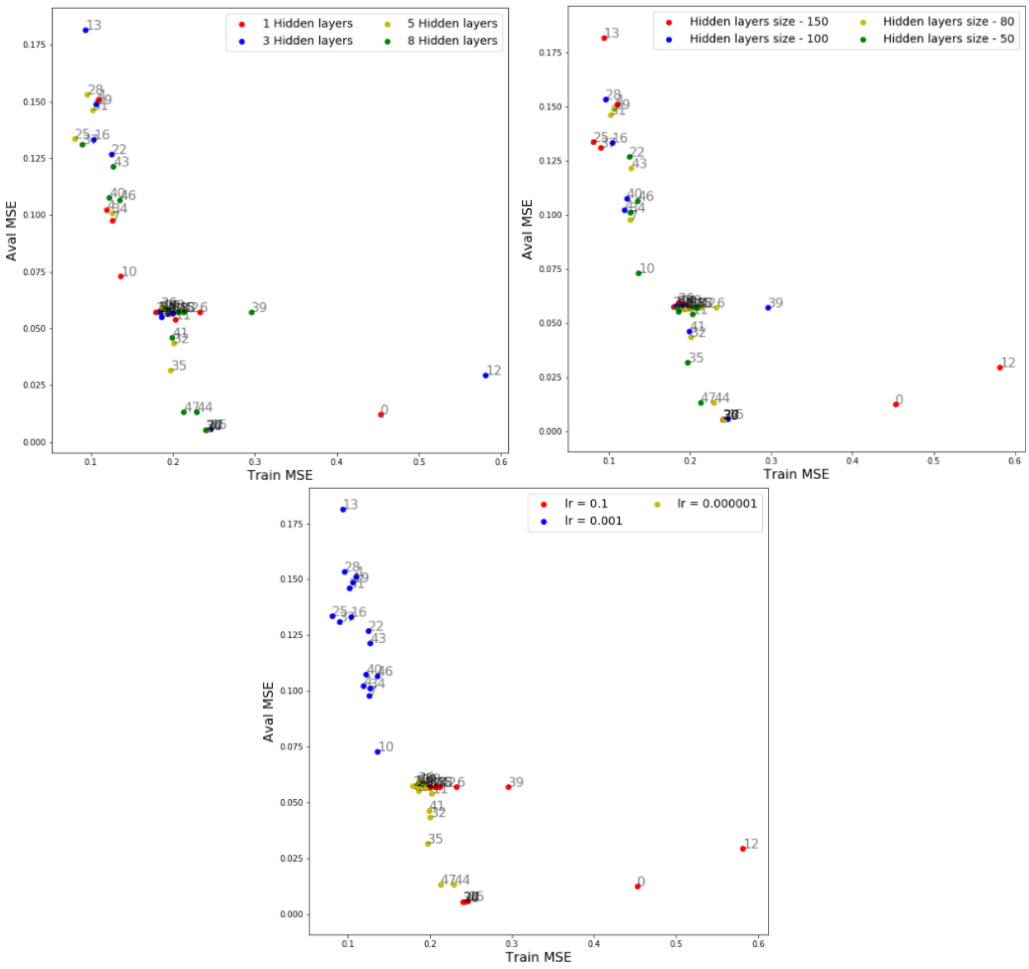


Figure 5.4: TV grid search analysis.

what can be referred to as inconclusive. It is a mixture of every kind of model, random predictions, and overfit but also some good models in line with cluster 1 (although in smaller quantity).

Model 10 is alone because yields a combination that surpasses in the order of 50% the lift performance in comparison to the other models. Model 10 is the critical benchmark against the integrated model for Pt TV target.

Every experience completed, the insights and information retrieved fuel the gain of knowledge regarding the problem and the best paths to accomplish a viable solution. The learning rate proved to be the most influencing factor. The best fit value for learning rate is clearly between medium values and small values. For the integrated model, we defined a more narrow search space for learning rate, focused on exploring options of values around middle and small intervals.

Integrated model - Pt TV and Pt Internet target

With the grid search concluded for both single targets, Pt Internet and Pt TV, we have successfully modeled single targets and found multiple models for each one. Resulting from the grid search, we obtained models that are a better fit than others. Models considered the better fit, are recognized as a valid solution. Others are rejected due to random predictions, overfit, or simply because they were not good enough. Through evaluation of the lift curve of each model, we selected a set of models from each single target to be used as benchmark. Results from the benchmark models will be compared against the integrated model.

The objective of the thesis is to find an integrated model whose results surpass the ones obtained with the benchmark, in this case, single target models. If the objective is accomplished we have proven Internet and TV services share information. Also, proves that through a multi-task neural network we were able to extract information from the intersection, thus improving prediction capacity.

All the work put in the thesis culminates in the effort of achieving an integrated multi-task model. Thus, the knowledge acquired throughout single target modeling is of utmost help for deciding which search space should be defined to create the integrated models. Since the targets are the same, the best chance to have a good result is to use the values that better fit in a single model exercise. Learning rate is the most important parameter as we have been stating, so search space options were narrowed between mid range values and smaller values. The size of hidden layers was treated as secondary, as we felt the correlation between success and the parameter was small. For the number of hidden layers, we decided that the model should need more degrees of freedom since now one architecture would have to learn much more information, the option was to have a bigger/deeper hidden layer group.

```
Net and Tv Targets Grid Search Space:
{

    Train Days: [20210126, 20210119, 20201211, 20201216]

    Evaluation Days: [20210318]

    Number of hidden layers: [3, 5, 8]

    Size of each hidden layer: [150, 100, 80, 50]

    Learning Rate: [1e-3, 1e-4, 1e-5, 1e-6]

}
```

Figure 5.5: Internet and TV target grid search space.

Dictionary representation of search space used in grid search is given in figure 5.5.
Once again, we computed a total of 48 models, each one now deploys two lift curves(one

for each target). Models' spatial representation is identical to the one used in Pt Internet and Pt TV target. X axis holds the associated MSE training error and y axis has the associated MSE evaluation error. However, there is a difference. In the integrated model, we compute two associated MSE training error, one for each target (Pt Internet and Pt TV), and two associated MSE also one for each target. This means the values represented in figure 5.6 are given by the sum of the individual errors of each target.

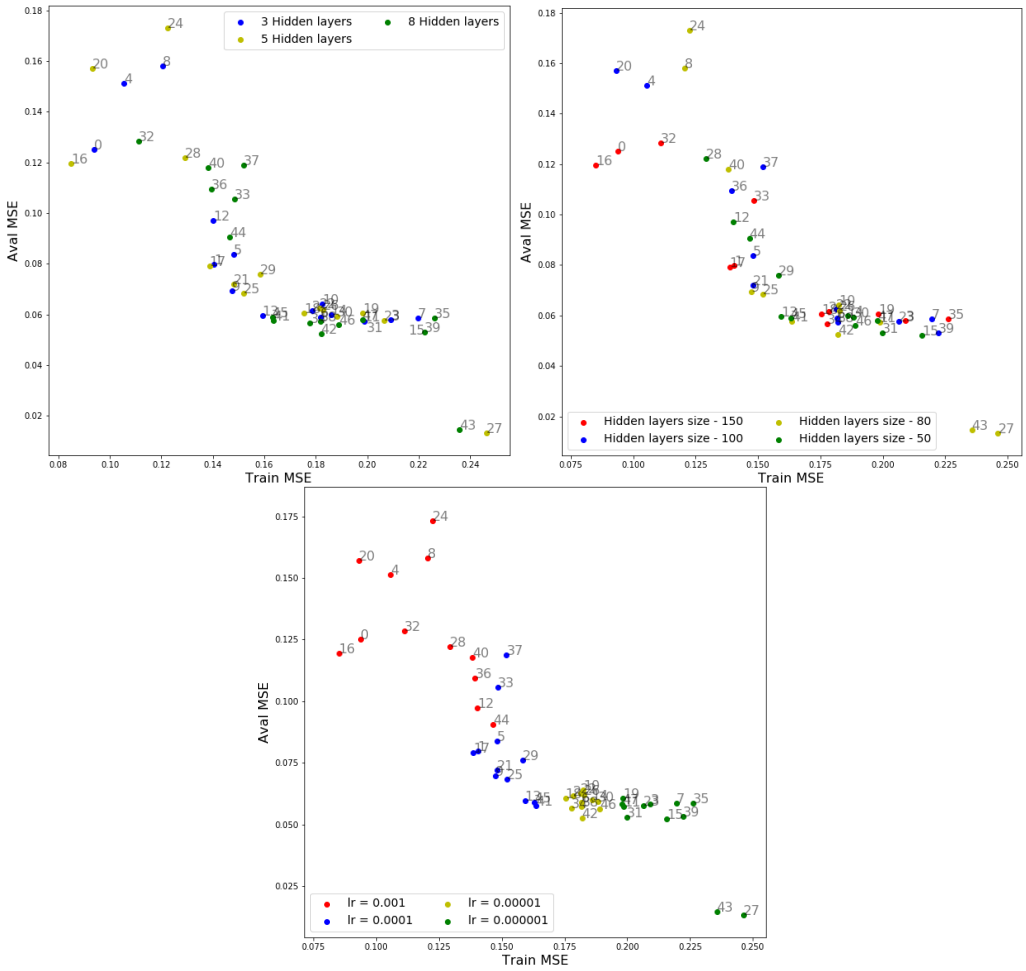


Figure 5.6: Internet and TV grid search analysis.

Contrarily to single target models, results in integrated model increased performance in lift curve evaluation when the associated MSE evaluation error decreased. As expected, the smaller evaluation error increases the model capacity to characterize the target, in this case, both targets. In figure 5.6 we can notice the smaller the associated MSE evaluation error of a model is, the greater associated MSE training error creating an inverse relationship. Actually, there is a logic behind the result, because in the training process the model did not overfit the data, it can better adapt when challenged with different data characteristics.

Because we evaluated the model with data from a different month, the patterns of consumption changed in comparison with patterns in training data. If the model had

learned so well merely the information of the training month, which relates to a specific consumption pattern, it would not be able to adapt to the evaluation data patterns. When model training starts becoming broader and less focused on learning just the training data, we start to notice more capacity to characterize both targets.

For the integrated model, learning rate hyperparameter took care of model segregation. Each learning rate value has its characteristics. The highest learning rate, defined by the red color, has evaluation error too high, which tells us those models had no capacity to adapt to the future. In green, concerning the smaller learning rate value is spatially represented in the saturation zone. The results consist of random predictions. The learning rates in (lr = 0.0001 and lr = 0.00001) that provided the best model results. Obtained lift curves were identical for both values. So, the discussion of results will be given in the description of the results section where the project culminates. As plotted in figure 5.6, the other parameters are secondary to model performance. The correlation between good results and those parameters is close to none.

The measurement of how good the model is depends on its capacity to predict both targets simultaneously. Both lift curves shall be better compared to the benchmark ones, delivered by single target model experience.

5.3 Description of the results and brief critical analysis of the main insights

In Chapter 5, we have discussed how the results should be interpreted given the company context. It is important to read lift curve results with business sense. Understand information that can be transferable to real operation actions. Theoretical explanation of lift measure and how it should be interpreted is given in Section 5.1.

The objective for the Section 5.2 was to detail each experience and the proposed goal for each one. Choosing search space is fundamental to the success of the models. Only through the tuning of the interval of values each parameter should vary that we achieve good model results. The grid search process and the analysis done over the resulting models gave precious feedback for the incremental performance in the thesis.

For the final section, the goal is to validate the thesis premise is to and state our final comparisons between the single target models and the evolution to the integrated model. Was the work done able to successfully achieve models capable of characterizing the targets? If yes, could we achieve a model that captures the intersection between TV and Internet services? Ingests information to gain more capacity to predict problems in each customer QoS? The integrated model surpassed the performance of the individual models? All results obtained to validate the proposed premises to be answered, alongside explanations on extracted insights, are discussed throughout the present section.

The main goal with the single target model exercise was to validate that through neural networks it is possible to model the targets. Also, gain knowledge about what

values are the best fit. Through the achievement of good results in line with the ones in current models, lift curves obtained assist as benchmark for comparison between single target models and the integrated model.

Figure 5.7 represents the top 3 models for single Internet target.

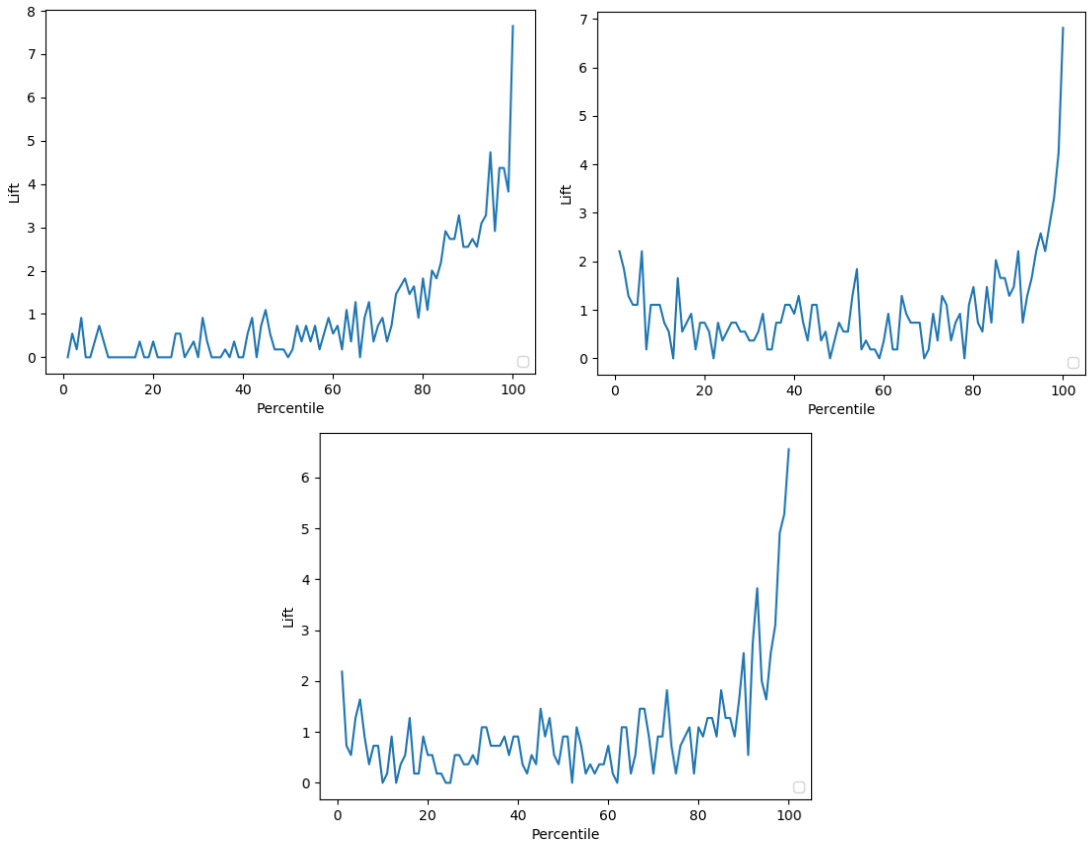


Figure 5.7: Top 3 Internet target single models. In Top-left is Model 5. In Top-right is Model 33. In Bottom is Model 18.

The reason behind choosing model 5 (in the top-left corner), model 33 (in the top-right corner) and model 18 (in the bottom) as the top 3 models to capture Internet target are: how good they perform in the last percentiles and stability in the lift curve.

Model 5 belongs to cluster 2, and so as traits we can see the highest lift for the highest percentiles, however it is a little noisy in the top percentiles. Nonetheless, because the increase in performance starts to be noticed around the 80 percentile, we can see that model 5 with more effort towards maturity could lead to the best result, both in top percentile performance and stability.

The other two models belong to cluster 1, where we stated were the best results. One common trait is the smoothness of the upswing in top percentiles. From the graphic, we can visualize that it is getting better as more confident the model is in its predictions. This effect shows stability and is more in line with the results from current models running in the project. Yet, model 18 is just a good third place. Due to the shaky randomness in

90 to 95 percentiles and lower predictive power in last percentiles, model 18 will not be used as benchmark.

Assigning between model 5 and model 33 which one should be used for benchmark resulted in a split decision. On one hand, model 5 shows higher performance in top percentiles, which must be valued. Even if we only can tackle two use cases with model 5, because we can identify more precisely which customers have problems the usage of money is much more efficient. On the other hand, model 33 is more in line with current models. It presents a smoother lift curve for the last percentiles, which can indicate more stability in results. Stability is interpreted as on a given day the capacity to obtain solid results is higher. With higher stability comes a lesser propensity to oscillate in performance.

The decision taken is to compare the results from integrated model with both model 5 and model 33 results from the single model Internet target.

Quality of data is an influencing factor when working with data gathered from the real world, that portrays the real consumption patterns of the customers. To add up, failures can happen because the systems DOCSIS and data warehouses can have problems collecting the data, or even due to the ingestion where systems can fail. There are targets more robust than others. Due to factors as these, the results obtained for TV target are not as good as the ones for Internet target. A drop in TV results performance was already expected because in current models the same effect is felt. The capacity to predict in top percentiles is not as high as for Internet, and stability wise also loses against Internet. For stability, as TV service has more dispersed patterns of consumption. The TV service has more diversified options. Thus the failure can happen in more points of the service making it more difficult to map. Stability performance is affected by the sparsity of events that can influence a customer being with problems.

Figure 5.8 presents the top 3 models results. The top-left results were achieved from model 10, the ones in top-right were obtained with model 23, and the ones in the bottom were obtained with model 1.

The first graph in the top-left corner respects to model 10, which was not aggregated to any cluster. Model 10 was selected as the one for benchmark due to its capacity to predict confidently customers in top percentiles. In the company context if we could efficiently take care of the two or three percentile would be a win. Stability may not be as smooth as in Internet target, however all models for TV target lack performance in stability measurement.

The other two models lack the capacity to predict TV target. Even tho model 1 (in the bottom) does not output random predictions and fell with more maturity could land some stability, its predictive power is too low and can not be considered. For TV target single model the challenge was significantly harder and the results are proof of that. In the end, we believe model 10 is in line with current model results and is eligible to be used as benchmark.

Both grid search experiences for single target models, Internet target and TV target,

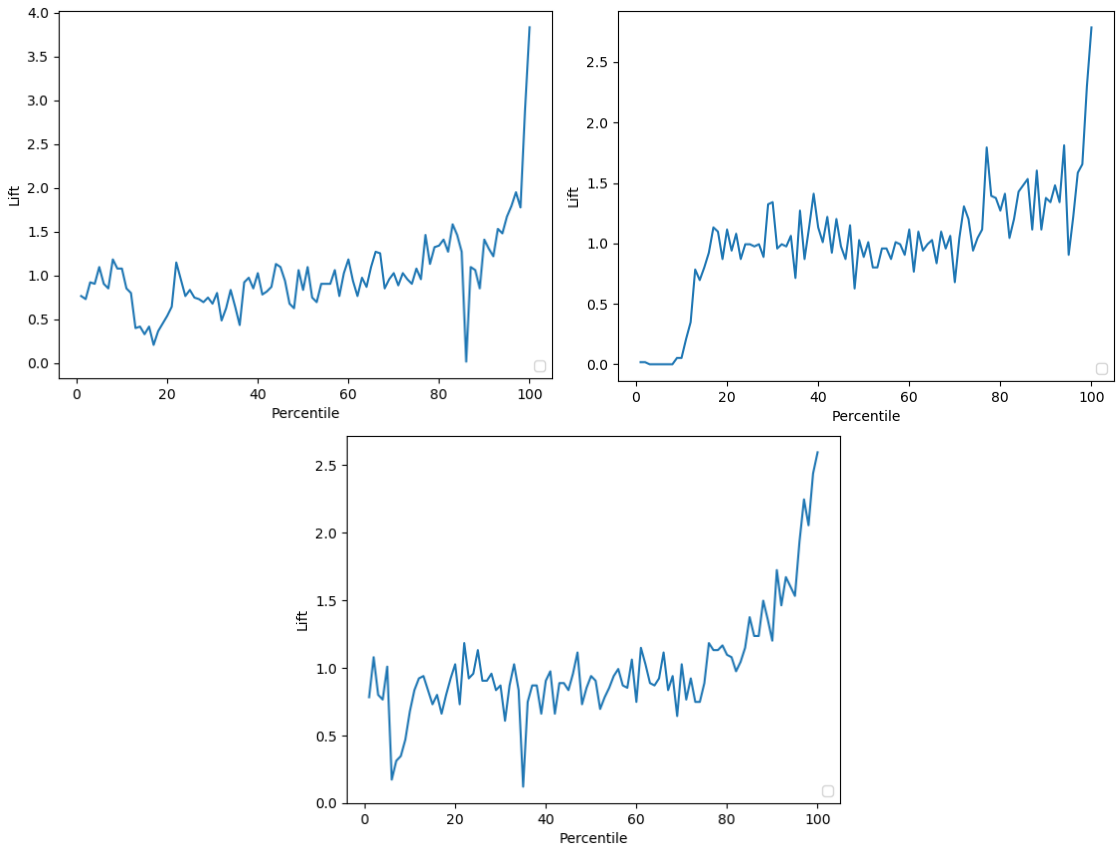


Figure 5.8: Top 3 TV target single models.Top-left Model 10. Top-right Model 23. Bottom Model 1.

were able to provide models that could capture the respective target dynamics. Selected solutions to be used as benchmark are in line with the ones in current operation for the project, and also are considered good enough to support the business specifications to enable operation near the customer. We feel confident that the comparison between single target models and integrated model is supported by a good benchmark.

Ambition for the thesis was very high as we are confident in the premise and in the solution. For that reason, all the work was implemented over real data gathered from customers' services consumption. The integrated model aimed to prove to the company, that both services have a higher correlation than the one detected. And, by exploring correlation the results can be improved. But the ambition is not limited, as we set the goal of introducing to the telecommunication community a new approach via a transfer learning algorithm, multi-task learning, that indeed was able to explore the services connection. Figure 5.9 plots the top 3 models results for multi-task exercise, which proves our approach to the problem increases performance in both services. Each row represents the result of a model. In the left are represented lift curves of Internet target and in the right lift curves for TV target.

The best model achieved for the problem is the integrated model 2 (in the top). Lift

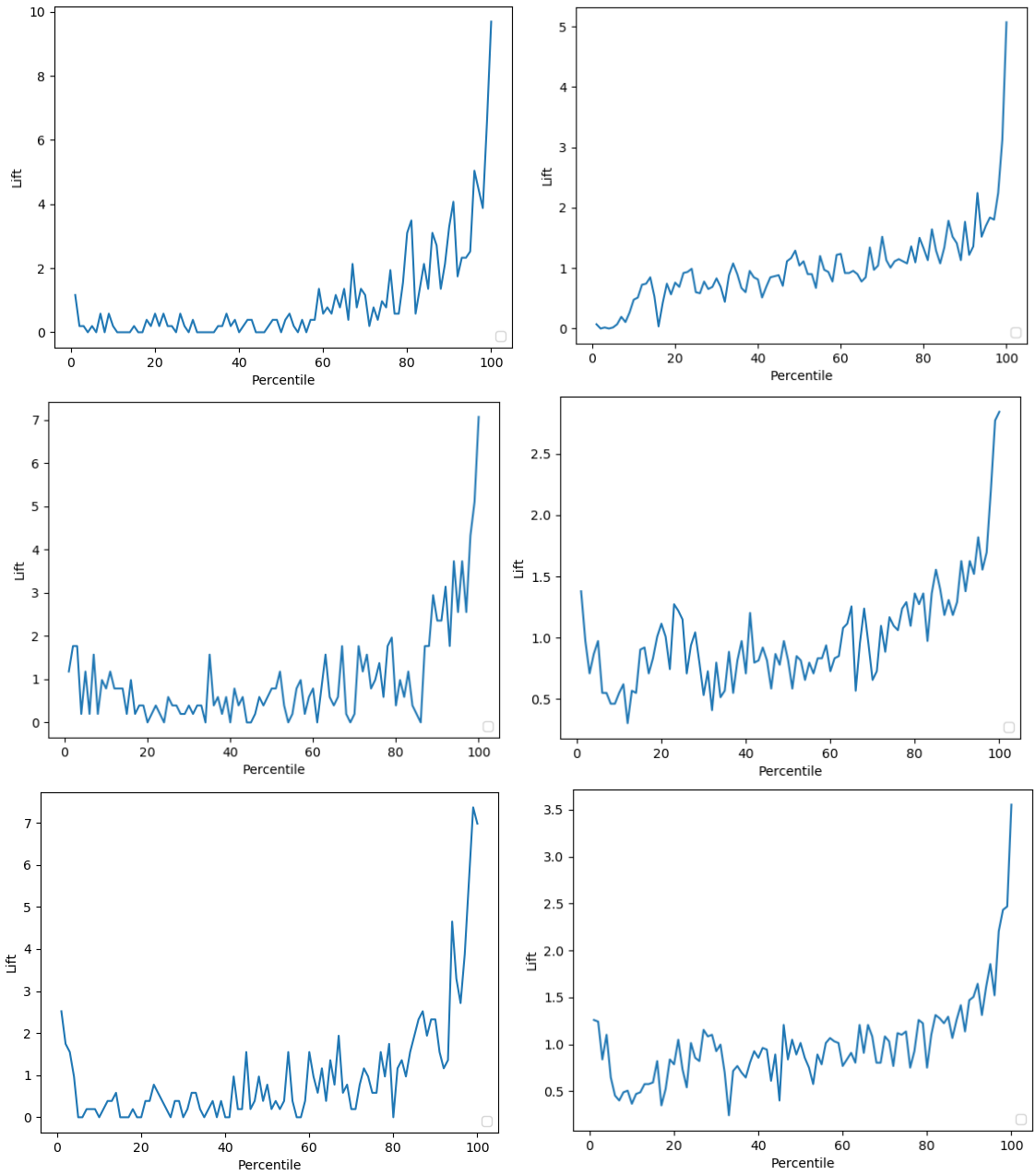


Figure 5.9: Top 3 integrated models. On the left are Internet target lift curves. On the right are TV target lift curves. Top model is 2. Middle model is 5. Bottom model is 9.

metric performs very well for top percentiles in both targets. Model 2 for TV target shows high stability and the lift measure in top percentiles surpasses every other seen in single target or integrated target. For Internet target the cost efficiency for the model is expected to improve as the lift measure grows above 25% in comparison with the benchmark results, both single and integrated.

The learning rate adopted in model 2 was lr = 0.00001. The knowledge acquired through the project, and more precisely single models grid search, converged for trying a new value for learning rate, which come to deploy the best model fit for the problem.

From model 2 results in both TV target and Internet target we can validate the thesis premise and proposed solution. Results displayed in the top row of figure 5.9, demonstrate that the services indeed intersect and there is precious information hidden in that connection, which proves our premise. On top of the premise validation, we constructed a model that works with real data and can capture the information shared between targets, transforming it into knowledge. Proactive approach towards the customer increases as new cases can be targeted. Based on model 2 conception and on the obtained results we can affirm that multi-task learning technique is able to learn from the implicit information shared by the two targets.

From a new approach, we could extract information into insights that ultimately will be transformed into knowledge. Results achieved will bring more possibilities to the team to improve project efficiency. Model 2 proves our effort and tenacity to be worth and in the end, we could create new knowledge for the community.

5.4 Integrated TV and Internet target model: Performance outburst

In Section 5.3 the objective was to analyze each target experience results. Some pros and cons were mentioned, however, we did not enlighten why the proposed model as the solution is an improvement over the single target solution. There are a couple of insights that should be more detailed and compared between solutions to evidence the differences.

The following arguments focus on integrated model comparison against single target models, and how the proposed solution validates the premise and reaches the objective of discovering new applicable knowledge.

Throughout the report, one main concern regarding thesis development relates to the fact that proactively reaching the customer has a cost. So the better is the model identifying customers with QoS pain more efficient the operation becomes. Correctly identifying all customers is not yet a possibility as the real data is too noisy. Having a higher lift in the top percentiles correlates with the confidence the model has in a prediction with the actual assertiveness when contacting the customer.

Integrated model surpassed the performance of both services single target models

regarding efficiency in top percentiles. For Internet target lift curve in the integrated model, architecture 2 in figure 5.9, we have a maximum lift of 10 against 7 or 8 from single models (5 and 33) in figure 5.7. The gap between the lift performances is about 25% which is an improvement. For the business and the operation, such an increment in performance would be enough to start piloting use cases with the integrated model. Still, in lift performance topic, the other integrated models also have higher or at least comparable lift performances when compared with single models. This information validates the premise that Internet and TV services intersect and supports the development of the integrated model because it can indeed explore the connections and absorb the knowledge.

Considering future perspectives, from the analysis of the results, we affirm that integrated model is more capable to evolve and upgrade than single target models.

If for Internet target the improvement was great, for TV target we believe integrated model made the difference and improved the quality of the prediction system. First, versus the benchmark TV model, the increment in performance is more than one lift unit, from 4 to more than 5. This difference maps to 25% increase in efficiency for top percentiles. Once again business and operation team would have no doubt in testing the integrated model for real world results.

However, the improvements are not over. One of the biggest questions for TV target single models surmises the stability problem. We could not see an organic increase in performance for the top percentiles in the results of TV target single model. That changes with the integrated model. Due to information exchange between targets, the model capacity to characterize TV target and mitigate the inherent noisy consumption of the service improved. From the lift curves in figure 5.8, where the perspectives of having a model capable to be matured into a solution were low, we achieve through multi-task learning a solution that displays optimistic perspectives.

Integrated model through multi-task learning prove to provide substantially better performance in all performance evaluation metrics, representing the main evidence of the effectiveness of the methodology proposed in the thesis.

6

Conclusion

The goals proposed for the thesis were ambitious because the objective was to have an impactful thesis that could improve the project in NOS and provide newer approaches for the telecommunication community. There were two main problems to be addressed: validating the intersection between tv and net services is indeed higher than currently acknowledged, and developing a solution that could take advantage of the tv and net services intersection.

The project where the thesis is integrated was already running for a couple of years so there was a high amount of knowledge and insights we needed to pick up. Impacting the customer is hard and we have to develop solutions that provide the best chances of success. Exploring the company system and customer base allowed us to understand our mission and the reason why succeeding in the thesis solution was very important. We took advantage of exploring the data to work on an analysis to conquer the first main objective which was validating the intersection between services. Through the analysis made we found that in reality the number of occurrences when a client had bad metrics in one service but his/her complaint was towards the other service was indeed bigger than currently acknowledged.

With the services intersection premise validated, we start working on a solution that could leverage that information into results we could use to improve the project operation. Multi-task learning provided the tool to answer current model limitations. It is through multi-task architecture that sharing information from both services becomes a reality. Accomplishing the end-to-end inference module was an iterative process where we developed multiple tasks and smaller functions in order to create a model capable of predicting confidence for each customer to complain.

The results obtained in the thesis show that we could address successfully both problems we purposed. By achieving a performance 25% better with a model that leverages the implicit information existent in the services intersection, we validated that there is more information in the intersection and we created a model capable of leveraging the intersection information.

6.1 Challenges and Final remarks

The process to achieve an end result satisfactory and capable of correcting the purposed problems was filled with challenges. To accomplish the thesis we had to improve our skills and overcome barriers. Specially in the beginning, the overflow of information to retain was very high. Because we are developing a product for a project, there is the need to understand the project context, objectives and operational ways. Blending in such structured environment is a challenge and takes time. The thesis uses real world data that is noisy, very susceptible to failures and maps the customers real use of the service. It took some time until we get the full grasp in the context and the data available to work. To understand the data we have to explore it, the issue is that due to the volume we have to use big data tools that work over a distributed system framework. The combination of properly handling distributed system framework and exploring the data was a great task in the project planning. Lastly, construct a multi-task model was an iterative process with multiple bugs that took some developing time away.

Challenges are part of any project. Difficulties map the development of solutions, and we can state in the end we were able to solve both problems with great success. Every skill and tool acquired during the process make any further developments easier and quicker to achieve. The project lays a base for a multi-task solution, and its the structure is flexible to accommodate newer features and targets.

The compromise took was to evolve the current solution, giving to the company new insights about tv service and net service intersection and a model capable of improving the project performance. Both objectives are accomplished with great results.

6.2 Future Work

Regarding future work, the objective is to describe features that can be added to the thesis project and improve the developed solution. We achieved our main goals for the thesis, however as a group of engineers we always see further developments that can take the project to another level. The two suggested features aim to improve the solution in relation to the business/operation side and increase the model range of action. We believe both are achievable and can make an impact on the project. Further research on the topics could lead to innovative ideas and solutions, which can present novel knowledge to the company and community.

The complete framework of the project is not enough to signal the customers with a bad service experience. To act over the signaled customers, the operation team needs to be confident that the resources allocated indeed are going to make an impact. It is important to have a root-cause analysis module on top of the inference predictions. In cooperation with explicative variables, root-cause analysis can map what point of the network (e.g. infrastructure, equipment, others) is at the origin of the failure or degradation problem.

An explainability component is essential to the operation, gives confidence to the business and the operation that the impacted clients return the investment resources.

Our multi-task solution is the base construction to every new target we want to add. Via improvements in the architecture, and research it is possible to add more targets to the problem. By increasing the number of targets we improve our model capacity to learn, as we are feeding it more implicit information. As long as there is some intersection between the targets, multi-task architecture can be used to further widen the project capacity to respond to more problems.

The two main targets we have in mind are customer satisfaction feedback and a global score function. Due to limited resources, it is important to target the customers we can make a higher impact. Through customer satisfaction feedback we can personalize our actions towards the customer. The global score function aims to have a unique score per client. The model can signal customers by their level of dissatisfaction with the service. Our action will be focused in improving the customer experience.

Bibliography

[1] N. Aloysius and G. M. *A Review on Deep Convolutional Neural Net-works, International Conference on Communication and Signal Processing*. 2017 (cit. on pp. 13, 14).

[2] H. V. P. Anirudh Sridhar. *Sequential Estimation of Network Cascades*. 2020 (cit. on p. 2).

[3] A. S. Arohan Ajit Koustav Acharya. *A Review of Convolutional Neural Networks*. 2020 (cit. on p. 13).

[4] S. F. Ashima Chawla Paul Jacob. *Interpretable Unsupervised Anomaly Detection for RAN Cell Trace*. 2020 (cit. on p. 15).

[5] M. T. Cheolmin Kim Veena B. Mendiratta. *Unsupervised Anomaly Detection and Root Cause Analysis in Mobile Networks*. 2020 (cit. on p. 2).

[6] M. Crawshaw. *MULTI-TASK LEARNING WITH DEEP NEURAL NET-WORKS: A SURVEY*. 2020 (cit. on p. 19).

[7] J. L. B. Diederik P. Kingma. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*. 2015 (cit. on p. 50).

[8] E. S. Faruk Selmanovic. *GPON in Telecommunication Network*. 2010 (cit. on p. 9).

[9] ITU. *Definitions of Terms Related to Quality of Service*. 2008 (cit. on p. 5).

[10] ITU. *New definitions for in-clusion in Rec. ITU-T P10/G100*. 2016 (cit. on p. 6).

[11] ITU. *Terms and Definitions for the Internet of Things*. 2012 (cit. on p. 10).

[12] T. Janevski. *QoS for Fixed and Mobile Ultra-Broadband*. IEEE Press, 2019 (cit. on pp. 5–7, 9–11).

[13] J. V. Jaroslav Frnda Jan Nedoma and R. Martinek. *A Hybrid QoS-QoE Estimation System for IPTV Service*. 2019 (cit. on p. 12).

[14] L. Y. Jianbo Yu Xing Liu. *Convolutional Long Short-Term Memory Autoencoder-Based Feature Learning for Fault Detection in Industrial Processes*. 2021 (cit. on p. 18).

[15] Y. Z. Lihua Song Peiya Li. *A QoS Mechanism for CM in HFC Network*. 2009 (cit. on p. 7).

[16] K. L. Lueth. *IoT market analysis: Sizing the opportunity* (cit. on p. 10).

[17] T. C. Miha Vuk. *ROC Curve, Lift Chart and Calibration Plot*. 2006 (cit. on p. 61).

[18] R. L. Na Zhang Mingsen. *A service-classified and QoS-gurantanteed triple play mode in FTTH Network*. 2006 (cit. on pp. 8, 9).

[19] N. A. E. Nadzurah Zainal Abidin Amelia Ritahani Ismail. *Performance Analysis of Machine Learning Algorithms for Missing Value Imputation*. 2018 (cit. on p. 47).

[20] T. Rokkas. *Technoeconomic analysis of PON architectures for FTTH deployments*. 2015 (cit. on p. 9).

[21] U. Sara and M. Uddin. *Image Quality Assessment through FSIM, SSIM, MSE and PSNR—A Comparative Study*. 2019 (cit. on p. 50).

[22] L. A. Shalabi and Z. Shaaban. *Normalization as a Preprocessing Engine for Data Mining and the Approach of Preference Matrix*. 2006 (cit. on p. 48).

[23] A. M. Shubham Maheshwari. *Hierarchical Autoencoder for Collaborative Filtering*. 2018 (cit. on p. 15).

[24] X. L. Tianxiang Sun Yunfan Shao. *Learning Sparse Sharing Architectures for Multiple tasks*. 2020 (cit. on p. 19).

[25] W. S. Xia Zhao Xiao Han. *Time series prediction method based on Convolutional Autoencoder and LSTM*. 2019 (cit. on p. 18).

[26] C. H. Yong Yu Xiaosheng Si. *A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures*. 2020 (cit. on pp. 16–18).

I

Annex for Training with Big Data software: Batching

Listing I.1: Batching for train and evaluation dataset

```
#create dictionaries to hash spark dataframes with batch column
#key is the day
train_dict = {}
aval_dict = {}

#get distinct days(keys)
unique_day_t = train_df_.groupBy("cal_day_id").count().toPandas()
unique_day_a = df_aval.groupBy("cal_day_id").count().toPandas()

#select df per day and store into dictionary (train)
batch_size = 250000
for day in unique_day_t['cal_day_id'].values[::]:
    #get day df
    day_df = train_df_.where(sqlF.col("cal_day_id") == int(day))
    #count number of entries in each day
    rows_Entries = day_df.count()
    #assign an ID to each row
    day_df = day_df.withColumn("Batch_ID", (sqlF.rand()*(rows_Entries//batch_size)).cast("int")
    #store the indexed day in dictionary
    train_dict[day] = day_df

#select df per day and store into dictionary (aval)
batch_size = 250000
for day in unique_day_a['cal_day_id'].values[::]:
    #print(day)
    day_df = df_aval.where(sqlF.col("cal_day_id") == int(day))
    #count number of entries in each day
    rows_Entries = day_df.count()
    #assign an ID to each row
    day_df = day_df.withColumn("Batch_ID", (sqlF.rand()*(rows_Entries//batch_size)).cast("int")
    #store the indexed day in dictionary
    aval_dict[day] = day_df
```

II

Annex for Training with Big Data software: Dataset balancing

Listing II.1: Distributed to local process. Dataset balancing.

```
root = 'batches'

days_train = [rnd for rnd in train_dict.keys()]
days_aval = [rnd for rnd in aval_dict.keys()]

targets_b = ['pt_target_exit_tipification_trust_0d_5d_flg_tv', 'has_pt', 'event_done',
'augmented_churn', 'Net_Score_tv_ix', 'Net_Score_net_ix','Q1_J', 'Q2_J', 'Q3_J', 'Q4_J']

for days in days_aval:

    #get day batch
    df_day = get_df_day(train_dict, days)
    #get unique batches ID
    #unique_train_days = [i.Batch_ID for i in train_dict[days].select('Batch_ID')
    .distinct().collect()]
    unique_train_days = [i.Batch_ID for i in aval_dict[days].select('Batch_ID')
    .distinct().collect()]


    for batch in unique_train_days:
        ind_batch = get_batch_for_training(df_day, batch, 'null')

        #Convert spark batch to pandas
        pd_batch = spark_to_pandas(ind_batch)

        #balance dataset
        pd_batch = balance_batch(pd_batch, targets_b, 0)

        #Save batch to csv
        path = os.path.join(root, f'batch_{batch}_{days}.csv')

        pd_batch.to_csv(path, index=False)
```

Listing II.2: Extract day in spark dataframe object

```
def get_df_day(train_dict, day):
    """
    Input: train_dict: dictionary with spark dfs for training
           day: (int) value of the day to retrieve -- this value is a key for the dict
    Output: return spark df of the day (cal_day_id)
    """
    df_day = train_dict.get(day)
    return df_day
```

Listing II.3: Extract instances of batch, given batch in function parameter

```
def get_batch_for_training(df, batch_index, target):
    """
    We are not able to train all data points in one time. Also Multi-task training method
    is to sequentially train different tasks in small batches.
    Random retriving of data points
    Input: df - training dataset
            target - which target to retrieve
            batch_index - what is the desirable index to retrieve
    Output: batch to train

    Spec: df col name where indexes are stored has to be: Batch_ID
    """
    #first iteration will not have target specification
    #get data with batch index: BATCH_index
    batch_train = df.where(sqlF.col('Batch_ID') == batch_index)

    #second iteration may leverage target selection
    #batch_train = batch_train.select(*([sqlF.col(col) for col in batch_train
    if(col not in targets)]) + target))

    return batch_train
```

Listing II.4: Balance dataset.

```
def balance_batch(pd_batch, targets, negative_rows):#, n_bin, n_zero, n_neg):
    """
    Function balances a pandas df to obttain a train data with all positive examples
    and complementary negative examples
    Input:  pd_batch: pandas df
            targets (list): which targets to search for. targets must come in order so
            that we process binary first - different than zero second -
            and different than -9999 third
            negative_rows: number of negative examples to select
            -- not for now n_bin, n_zero, n_neg: these variables contain the number
            of targets that are of each condition
    Output: pandas df balanced (smaller)
    """
    df_pos = pd_batch.loc[(pd_batch[targets[0]] == 1) | (pd_batch[targets[1]] == 1) |
```

```
    (pd_batch[targets[2]] == 1) | (pd_batch[targets[3]] == 1) |
    (pd_batch[targets[4]] != 0) | (pd_batch[targets[5]] != 0) |
    (pd_batch[targets[6]] != -9999) | (pd_batch[targets[7]] != -9999) |
    (pd_batch[targets[8]] != -9999) | (pd_batch[targets[9]] != -9999)]

    df_neg = pd_batch.loc[(pd_batch[targets[0]] != 1) | (pd_batch[targets[1]] != 1) |
    (pd_batch[targets[2]] != 1) | (pd_batch[targets[3]] != 1) |
    (pd_batch[targets[4]] == 0) | (pd_batch[targets[5]] == 0) |
    (pd_batch[targets[6]] == -9999) | (pd_batch[targets[7]] == -9999) |
    (pd_batch[targets[8]] == -9999) | (pd_batch[targets[9]] == -9999)]
    #randomly select n rows from neagtive examples
    #df_neg = df_neg.loc[random.sample(df_neg.index, negative_rows)]
    df_neg = df_neg.sample(n = negative_rows)

    #join both dataframes
    df = df_pos.append(df_neg, ignore_index = True)

    return df
```