Sérgio Tiago Oliveira Jorge

Roteamento Inteligente de Chamadas

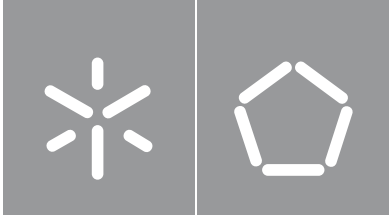
UMinho | 2020

Universidade do Minho
Escola de Engenharia

Sérgio Tiago Oliveira Jorge

Roteamento Inteligente de Chamadas

outubro de 2020



Universidade do Minho
Escola de Engenharia

Sérgio Tiago Oliveira Jorge

Roteamento Inteligente de Chamadas

Dissertação de Mestrado
Mestrado Integrado em Engenharia Informática
Engenharia Informática

Trabalho efetuado sob a orientação do
Professor Doutor Paulo Jorge Freitas de Oliveira Novais
Doutor Carlos Miguel Silva Couto Pereira

outubro de 2020

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO POR TERCEIROS

You are young and life is long and there is time to kill today.
And then one day you find ten years have got behind you.
No one told you when to run, you missed the starting gun.

So, you run and you run to catch up with the sun but it's sinking
Racing around to come up behind you again.
The sun is the same in a relative way but you're older,
Shorter of breath and one day closer to death.

Pink Floyd *in* 'Time'.

AGRADECIMENTOS
ACKNOWLEDGEMENTS

Esta dissertação é o culminar de um longo e exigente percurso de cinco anos. Para se chegar aqui, inevitavelmente, foi necessário o apoio e o incentivo incansável de diversas pessoas e entidades. Por isso, gostaria de expressar um agradecimento muito sincero a todos, mas também aos que participaram ativamente na sua realização:

Ao Doutor Carlos Pereira, um obrigado muito especial, pela disponibilidade e atenção, pelos ensinamentos e persistentes críticas e sugestões.

Ao prof. Doutor Paulo Novais pelo apoio prestado na orientação deste estudo.

À Joana, pela companhia, pelos conselhos, pela leitura crítica do trabalho e, especialmente, por tudo aquilo o que representa.

À minha família sempre presente ao longo de toda a minha vida académica, e em especial, aos meus pais, à minha irmã, pelo apoio e incentivo incessante, e aos meus avós por serem uma inspiração e um exemplo de vida. Muito obrigado!

Ao Vítor e à Diana, pela entreajuda e pelo companheirismo, que se revelou essencial para o sucesso deste projeto.

Aos meus amigos, sem exceção, pelo apoio e compreensão e pelos bons momentos compartilhados. Sem vocês, o meu mundo não teria cor. Sintam-se fortemente abraçados!

À Universidade do Minho e aos seus docentes e funcionários, por tudo o que me deram ao longo destes anos, seja a nível profissional como a nível pessoal.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Nas empresas de telecomunicações, as centrais de chamadas são os elementos que têm maior interação com os clientes, e o desempenho dos operadores é vital porque um excelente serviço satisfaz o cliente e ajuda a um melhor funcionamento. Deste modo, tenta-se utilizar dados de clientes, dados de operadores de chamadas e dados históricos de serviço de forma a melhorar o suporte. O emparelhamento de um cliente com um operador que se sinta confortável com o problema a resolver ajuda as empresas a reduzir custos, melhora o atendimento ao cliente e aumenta a produtividade dos colaboradores.

Nesta dissertação, propõe-se um modelo de previsão, baseado em *machine learning* e em otimização, que antevê o problema pelo qual o cliente está a ligar e encaminha a chamada e o cliente para o operador mais apropriado.

Após a análise de um dataset não balanceado, com aproximadamente 2.9 milhões de entradas, e recorrendo à metodologia CRISP-DM para modelação, alguns algoritmos inovadores como o LightGBM, permitiram obter um micro-F1 de 0.41 e AUC-ROC de 0.84. Deste modo, pode-se aferir alguma capacidade na previsão da razão para o cliente estar a contactar a empresa.

A alocação e otimização num cenário simulado, pós-previsão, indicou uma melhoria na ordem dos 50%, relativamente ao ganho de eficiência, eficácia e rapidez do call-center.

Os resultados mostram que a utilização de grandes quantidades de dados comerciais para a previsão pode melhorar o desempenho do suporte ao cliente. Além disso, as conclusões e as estratégias exploradas nesta tese podem levar à utilização deste sistema, e com isso ajudar a empresa a obter maiores lucros e podem ajudar futuros investigadores a tomar melhores decisões no estudo e no desenvolvimento de soluções parecidas.

Palavras-chave: Call-Center, Ciência de Dados, CRM, Inteligência Artificial, *Machine Learning*.

ABSTRACT

At telecommunications companies, call-centers have the highest interaction with customers, and the operators' performance is vital because an excellent service satisfies the customer and helps a better operation. Therefore, attempts are made to use customer data, call operator data, and historical service data to improve support. Pairing a customer with an operator who is comfortable with the problem to solve helps companies reducing costs, improves customer service, and increases employee productivity.

In this thesis, we propose an approach based on machine learning and optimization, which predicts the problem for which the customer is calling and routes the call and the customer to the most appropriate call operator.

After the analysis of an unbalanced dataset, with approximately 2.9 million entries, and using the CRISP-DM methodology for modeling, some innovative algorithms such as LightGBM, allowed obtaining a micro-F1 of 0.41 and AUC-ROC of 0.84. These results allow us to gauge some capacity in predicting why the client is contacting the company.

The optimization in a simulated scenario, post-prediction, indicated an improvement in the order of 50%, concerning the efficiency gain, effectiveness, and speed of the call-center.

The results show that using large amounts of business data for the prediction can improve customer support performance. Also, the findings and strategies explored in this thesis can lead to the use of this system, thus helping the company obtain higher profits and help future researchers make better decisions in the study and development of similar solutions.

Keywords: Artificial Intelligence, Call-Center, CRM, Data Science, Machine Learning.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÓNIMOS

A

AdaBoost Adaptative Boost

AUC Area Under Curve

C

CRISP-DM Cross-industry Standard Process for Data Mining

CRM Customer Relationship Management

F

FCFS First Come, First Served

FIFO First In, First Out

I

IA Inteligência Artificial

ID Identificador

IDE Integrated Development Environment

L

LightGBM Light Gradient Boosting Machine

M

ML Machine Learning

R

RNA Rede Neuronal Artificial

ROC Receiver Operating Characteristic

ROS Random Oversampling

RUS Random Undersampling

S

SAC Serviço de Apoio ao Cliente

SBR Skill Based Routing

SC Stacking Classifier

SMOTE Synthetic Minority Over-sampling Technique

T

TI Tecnologias da Informação

V

VC Voting Classifier

X

XGBoost EXtreme Gradient Boosting

GLOSSÁRIO

Bias Viés/Tendência.

Big Data Quantidades extremamente grandes de informação.

Churn Desistência dos serviços da empresa, por parte do cliente.

Dataset Uma coleção de exemplos. Cada exemplo contém uma ou mais características/ *features* e uma etiqueta (se utilizar aprendizagem supervisionada).

Feature É a especificação de um atributo e seu valor. Ela expressa um pedaço de informação mensurável sobre algo.

Inteligência Artificial (IA) Programas de computador concebidos para resolver problemas difíceis que os seres humanos (e os animais) resolvem rotineiramente. Permite às máquinas tomar decisões e realizar tarefas que simulam a inteligência e o comportamento humano.

Machine Learning (ML) O uso de algoritmos orientados por dados que funcionam de melhor forma, pois têm mais dados para trabalhar. O objetivo fundamental do ML é generalizar para além dos exemplos fornecidos no conjunto de treino.

Outliers É uma observação que apresenta um grande afastamento das demais da série.

Overfitting Quando um modelo criado corresponde aos dados de treino de forma tão próxima que não faz previsões corretas sobre novos dados, ou seja, dados que nunca viu.

Target Variável que se pretende prever.

Threshold Limiar ou fronteira que sendo excedida, despoleta algo.

Underfitting Quando um modelo não se adapta bem sequer aos dados com os quais foi treinado.

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO E MOTIVAÇÃO

Num mundo cada vez mais conectado, destacam-se as empresas que conseguem captar dados e informações dos seus clientes com a maior agilidade e precisão. A era digital e a área de *big data*, relativa à enorme quantidade de dados, têm de certa forma, auxiliado as empresas a tomarem melhores decisões nos seus negócios. Isto acontece, principalmente, porque em alguns métodos relacionados com a inteligência artificial e, em particular, *machine learning*, verificam-se melhores resultados e melhores modelos, quanto maiores forem os conjuntos de dados fornecidos às máquinas.

O mercado de telecomunicações tem, também, visto um crescimento contínuo desde o início do século XXI. A empresa que tornou esta tese possível constitui um grupo de comunicações e entretenimento português, que oferece soluções fixas e móveis de última geração como televisão, internet, voz e dados para todos os segmentos de mercado. Assim sendo, a quantidade de clientes (e dados) que possui é enorme, levando a potenciais grandes lucros e também a desafios no que toca a manter os seus consumidores satisfeitos. Por esse motivo, a resolução de problemas de clientes (dúvidas, questões técnicas, não técnicas) de uma forma rápida e eficaz é algo fundamental e muito procurado, não só nesta empresa, mas em qualquer organização nesta área. A empresa, como uma das maiores do sector no país procura, por isso, melhorar a qualidade e eficiência do seu atendimento ao cliente. Deste modo, métodos avançados de previsão e otimização, com base nos dados, são necessários para abordar esta problemática, devido à sua complexidade.

Até aqui e de um ponto de vista puramente económico, as empresas aprenderam que é menos dispendioso reter um cliente do que encontrar um novo. Pelo princípio de Pareto, verifica-se que 20% dos clientes de uma empresa geram 80% dos seus lucros e é 5 a 10 vezes mais caro adquirir um novo cliente do que fazer negócio com clientes atuais. Um aumento de 5% na retenção de clientes existentes traduz-se em mais de 25% de aumento na rentabilidade. Nesse sentido, a gestão de relacionamento com o cliente (CRM) é a definição dada ao processo de criação e manutenção de uma ligação com os clientes. É um processo complicado de identificação, atração, diferenciação e retenção de clientes e, por vezes, integra toda a cadeia de fornecimento de uma empresa de modo a dar mais valor ao cliente em cada etapa, seja através de maiores benefícios ou através da redução de custos. O resultado é um aumento no lucro através do aumento dos negócios, havendo uma coordenação perfeita entre vendas, serviço de atendimento ao cliente, marketing, entre outros. Em especial, E-CRM, representado na Figura 1, trata todas as formas de gerir as relações com os clientes fazendo uso das tecnologias da informação (TI), que têm vindo a crescer nas últimas duas décadas. Portanto, devido ao CRM, as empresas conhecem os seus clientes, compreendem as suas necessidades únicas e adaptam o seu serviço ou produto (Hassan et al., 2015).

Aprenderam, também, a necessidade de ser feita uma segmentação dos clientes, de forma a gerir os recursos de forma eficiente. Deste modo, é possível entender o valor do cliente e a

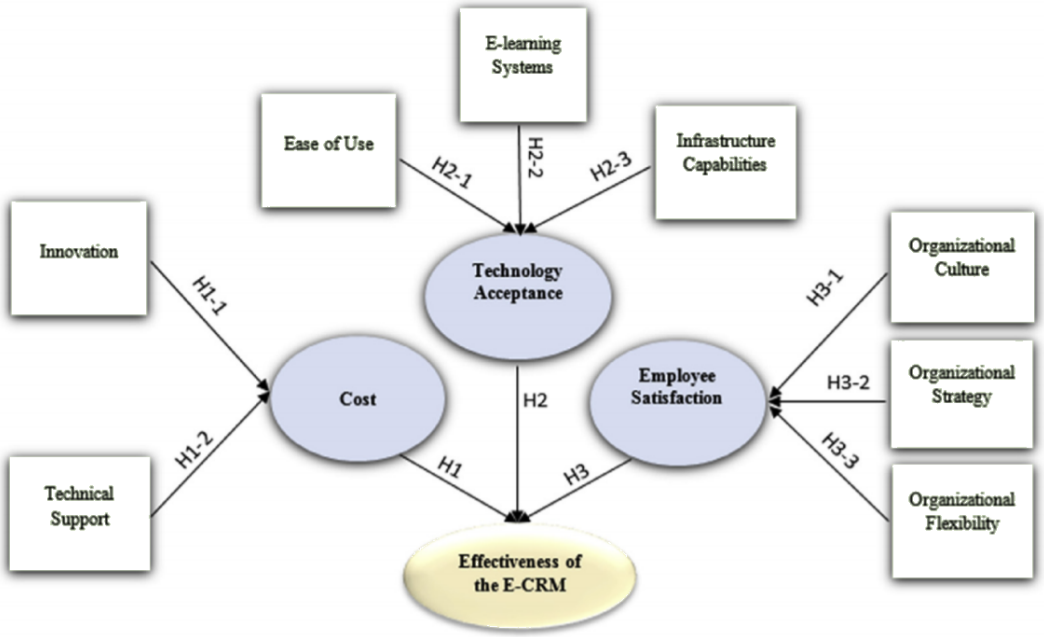


Figura 1: Modelo proposto sobre fatores determinantes para a eficácia do E-CRM (reproduzido de (Jafari Navimipour e Soltani, 2016)).

oportunidade. Concretizando, as características do cliente que devem determinar a forma como a empresa o trata são o seu valor e o seu custo. Portanto, clientes pouco valiosos devem ser redirecionados para um serviço de baixo custo e, consecutivamente, clientes de elite devem ser redirecionados para um bom serviço de apoio. Os clientes de alto valor (os que estão logo a seguir aos de elite) tendem a ser profissionais ricos que podem comprar mais se a empresa entender as suas necessidades e podem, por isso, ser um motor de crescimento potencial. O relacionamento com estes deve ser, também, próximo e cuidado (Genesys, 2014).

Os call-centers, que podem fornecer esse apoio, evoluíram bastante na última década. Mantêm-se algumas estratégias que, globalmente, melhoram os serviços, mas a realidade é que a tecnologia mudou drasticamente a forma como as centrais de atendimento podem funcionar. Hoje em dia, podem conseguir aproveitar todos os dados disponíveis de forma a impulsionar a interação de cada cliente. Conseguem saber que transações foram feitas pelo cliente na última hora, o que o cliente perguntou na última chamada, e muitas outras coisas. Além disso, podem também ter informações relativamente a satisfação do cliente, sentimento, propensão para *churn*, etc. Alavancar o contexto da conversa, por exemplo, é crucial porque permite que os operadores ofereçam uma conversa personalizada. Ou seja, usando dados históricos, é possível ter uma visão completa do cliente no sentido em que se pode verificar como é que o cliente já interagiu com a empresa. Usando dados em tempo real pode-se verificar questões relacionadas com experiência contínua. O contexto facilita o processo também para o cliente, já que este não precisa de se reexplicar múltiplas vezes. É, assim, um elemento chave para o encaminhamento das próprias chamadas ao levar o cliente certo ao agente certo, de forma rápida. As centrais de atendimento focam-se não só nos clientes, mas

também nos agentes e operadores de serviço. Cada agente faz-se comunicar de forma diferente e apela para um segmento diferente de clientes. É, por exemplo, evidente que alguns agentes não conseguem lidar com chamadas negativas, enquanto outros destacam-se em cenários do género (Fly, 2019).

A operadora, tal como todas as operadoras de telecomunicações, dispõe de um serviço de apoio ao cliente, disponível diariamente, com a vista a esclarecer dúvidas e a disponibilizar apoio às dificuldades técnicas ou avarias, resultantes de serviços contratados. Tem, portanto, um conjunto de centrais de chamadas munidas de operadores qualificados para tal.

Assim, há diversos contactos para os quais os clientes podem ligar, de forma a estabelecer apoio. Isto é, há, realmente, diferentes números de telefone associados a diferentes problemas. Contudo, por desconhecimento das pessoas, por incapacidade de o fazerem ou simplesmente porque as pessoas procuram uma ajuda da forma mais rápida possível, a linha primária, que deve ter a capacidade de fazer a triagem e a resolução de todos os problemas, é aquela que é mais contactada. Quando alguém contacta essa linha está a contactar os operadores da primeira linha que não são, necessariamente, especializados em todas as áreas e, portanto, quando não têm capacidade de resolver um problema, transferem a chamada para outro colega, agora na segunda linha. Essa transferência, contudo, não só traz custos à empresa porque implica uma chamada com maior duração e mais de dois operadores ocupados como, também, origina um aborrecimento do cliente porque, em todo o caso, tem que reexplicar o problema ao segundo operador.

Desta forma, procura-se melhorar o serviço fazendo uma previsão, prévia, em busca de se saber o porquê de o cliente querer ajuda e suporte. Isso possibilita à empresa fornecer uma assistência adequada, personalizada, rápida e confortável.

1.2 OBJETIVOS

O objetivo principal desta dissertação é a criação de um modelo de previsão que, a partir de dados históricos de clientes, fornecidos por uma empresa de telecomunicações nacional, consiga reencaminhar as chamadas dos clientes para o operador de call-center mais adequado para a resolução da questão em causa. Deste modo, é necessário o desenvolvimento de dois módulos, dependentes um do outro, que em conjunto permitem estabelecer uma solução para o problema apresentado. Ou seja, em primeiro lugar, há um foco na criação de um módulo que faz a previsão da razão da chamada do cliente, e por isso, tenta-se saber porque motivo o cliente está a contactar a empresa, com base em conceitos relacionados com inteligência artificial e *machine learning*. Na sua base estão algoritmos que permitem fazer previsões fiáveis, com base em dados passados e atuais dos sistemas. Em seguida, um segundo componente é desenvolvido de forma a se fazer uma alocação ao operador mais adequado, com base em técnicas de otimização e investigação operacional. Isto é, a chamada é atendida pelo operador disponível mais eficiente ou mais rápido

para o problema a resolver. É, por isso, pretendido que o produto final resulte numa melhoria do apoio ao cliente e num aumento do lucro da empresa, visto que os clientes são atendidos por pessoas mais especializadas na área em questão, o que origina uma resolução mais rápida e eficiente dos problemas.

Uma representação em alto nível da arquitetura a desenvolver é mostrada na Figura 2.

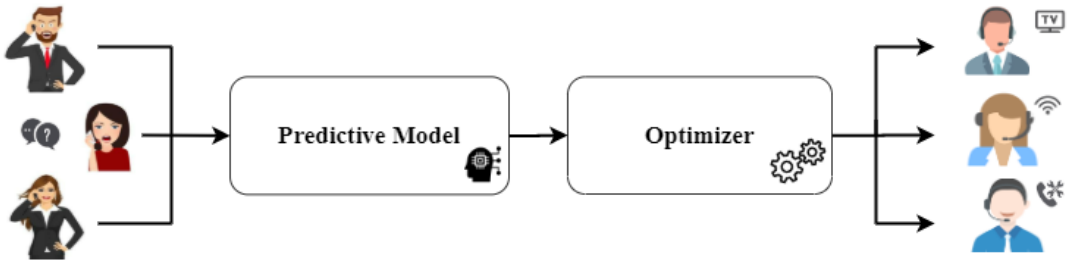


Figura 2: Arquitetura do sistema. As chamadas dos clientes são atribuídas ao operador do call-center mais apropriado, utilizando modelos preditivos e otimização.

1.3 CONTRIBUIÇÕES

Esta dissertação teve um artigo aceite, com revisão pelas partes para publicação no evento *21st International Conference on Intelligent Data Engineering and Automated Learning* - IDEAL 2020. O artigo está detalhado no Apêndice I.

- Jorge, S., Pereira, C. e Novais, P. (2020) 'Intelligent Call Routing for Telecommunications Call-Centers', *IDEAL 2020: 21st International Conference on Intelligent Data Engineering and Automated Learning*, Guimarães, 4-6 November 2020.

1.4 ESTRUTURA DA DISSERTAÇÃO

Esta dissertação está dividida em cinco capítulos. Primeiramente, faz-se uma introdução detalhada, no capítulo 1, que coloca o leitor em contacto com o problema e com os objetivos a alcançar. O capítulo 2, conceitos e tecnologias, descreve detalhadamente o contexto do problema, bem como as tecnologias utilizadas ao longo deste trabalho, necessárias à sua correta e sucedida implementação. Descreve, também, o trabalho relacionado, onde se procura informar sobre que literatura existe, atualmente, semelhante ao problema a resolver. O capítulo 3 descreve o módulo de previsão usado, em primeira instância, e é seguido do capítulo 4 que descreve o segundo módulo utilizado para otimizar e simular o sistema. As soluções alcançadas estão incluídas nessas secções. Finalmente, a dissertação termina com as conclusões, no capítulo 5, onde se resume tudo o que foi discutido e é apresentado o que poderá ser feito, num trabalho futuro, para melhorar os resultados obtidos.

2. CONCEITOS E TECNOLOGIAS

2.1 INTELIGÊNCIA ARTIFICIAL

Nos últimos anos, tem havido uma explosão de dados e, as telecomunicações, são umas das indústrias mais intensivas e poderosas no que a isso diz respeito. Existe uma grande oportunidade para os gestores de telecomunicações analisarem as grandes quantidades de dados que foram recolhidas, a fim de melhorar as operações de curto e longo prazo das suas organizações, ou seja, informações significativas podem ser extraídas de forma a tomar decisões apropriadas e chegar a conclusões que ajudem a indústria a crescer. A mineração de dados e *machine learning* têm sido usadas como ferramentas altamente eficazes na extração desta informação que está oculta nos grandes repositórios de dados. Alguns exemplos bem conhecidos disso, já explorados, são as análises e deteções de cancro de mama no sector biomédico, as pontuações de créditos no sector financeiro (Mishra e Rani, 2017), deteções de emoções (Costa et al., 2016), e até stress (Carneiro et al., 2015). Portanto, a evolução das novas tecnologias pode ter um papel muito importante, uma vez que podem tornar-se parte da solução para alguns dos problemas da sociedade (Novais et al., 2010).

Através de motores de pesquisa especializados em encontrar artigos científicos e académicos, tais como, *Google Scholar*, *IEEE*, *Web of Science* foi possível reunir alguma informação relativamente à forma como a comunidade tem visto, ao longo dos tempos, o que está relacionado com o tema em questão nesta tese. Usaram-se múltiplas palavras-chave tais como: 'predict customer behavior', 'machine learning customer', 'call routing customer', entre outras. Reuniram-se os resultados provenientes e foi possível verificar, como mostra a Figura 3, que tem havido um crescente interesse geral pela área relacionada com *machine learning*.

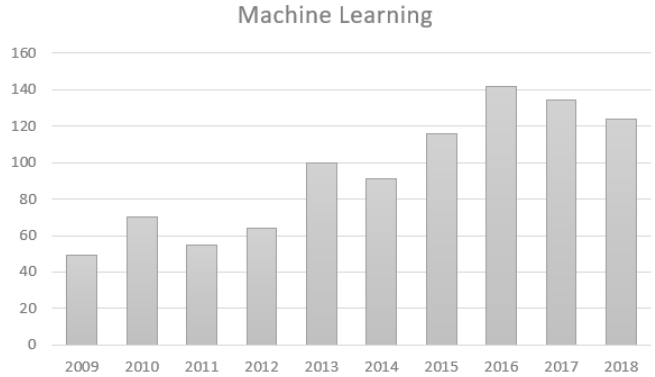


Figura 3: Análise ao crescente interesse da ciência por *machine learning*.

Verificou-se, também, como mostra a Figura 4, que há um crescente interesse em estudar e prever o comportamento dos clientes.

Em relação à determinação da satisfação das pessoas, alguns dados de telecomunicações sobre clientes, tais como detalhes da chamada e informações do cliente podem ser rentáveis se utilizados, na medida em que podem apoiar na determinação do comportamento do cliente e na

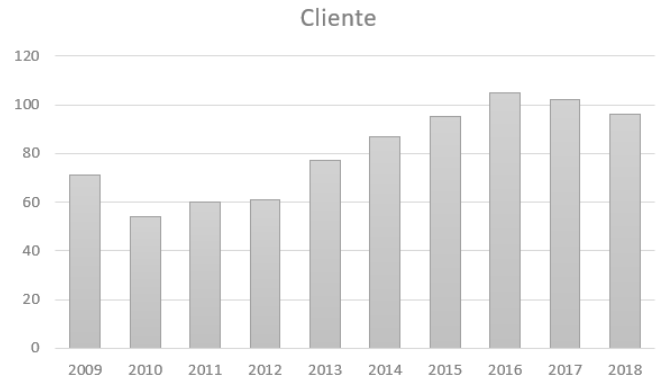


Figura 4: Análise ao volume de artigos relacionados com estudo do cliente.

identificação de oportunidades de apoio aos objetivos de expansão da base de clientes, reduzindo o *churn* (Tundjungsari, 2013).

Hoje em dia, existem, por exemplo, muitas abordagens ao maior desafio na indústria que é reter clientes. Portanto, os clientes podem mudar para outras operadoras por várias razões, tais como, melhores serviços, melhor viabilidade económica, melhor qualidade das chamadas, menos problemas na faturação, etc. Na indústria das telecomunicações, é muito comum este problema de clientes que mudam facilmente para outros concorrentes. Algumas propostas de solução para o problema, existentes no mercado, como a de (Mishra e Rani, 2017) mostram a necessidade de um bom classificador e a importância de um conjunto de dados pré-processado de forma apropriada, que remova dados desnecessários e redundantes. Isto é, as empresas de telecomunicações coletam informações sobre os seus clientes, e algumas delas não são relevantes para determinados contextos. Normalmente, estes dados têm um grande espaço de variáveis e uma distribuição de classes desequilibrada, uma vez que o número de *churners* é muito menor em comparação com os não *churners*. O conjunto de dados desequilibrado causa uma aprendizagem deficiente e fraca por parte de um classificador (Mishra e Rani, 2017).

Tem-se verificado, inclusivamente, que aplicações úteis não podem ser desenvolvidas sem compreender os vários dados utilizados nas indústrias de telecomunicações. Destaca-se, portanto, a aplicação de metodologias bem estabelecidas de maneira a criar aplicações e modelos inteligentes de alta qualidade e robustez (Preuveneers e Novais, 2012). Por isso, o primeiro passo no processo de *data mining* é compreender os dados. As diferentes categorias de dados utilizados nesta indústria estão agrupados principalmente em 3 tipos diferentes: dados detalhados da chamada com informações como data, hora e duração da chamada que podem gerar dados como duração média da chamada, número médio de chamadas originadas por dia, número médio de chamadas recebidas por dia, percentagem de chamadas durante o dia, percentagem de chamadas durante a semana, etc; dados da rede, isto é, as redes de *telecom* contêm milhares de componentes interligados que são capazes de gerar mensagens de erro ou de estado; dados do cliente como morada, histórico de pagamento, serviço contratado, entre outros. A informação pode ser usada para traçar

o perfil dos clientes e estes podem ser usados para fins de marketing e previsão. A ênfase da aplicação de marketing na indústria das telecomunicações passou da identificação de novos clientes para a medição do valor de clientes atuais e para um posterior tomar de medidas para os manter (Marwaha, 2014).

2.1.1 Definição

É a ciência que procura estudar e compreender o fenómeno da inteligência e, ao mesmo tempo, é também um ramo da engenharia porque tenta construir instrumentos que devem dar apoio à inteligência humana. Juntos, ciência e engenharia, pretendem construir máquinas que saibam realizar tarefas que, quando feitas por humanos, precisam do uso de inteligência.

Assim, busca o modo como os humanos pensam de forma a elaborar teorias e modelos da inteligência, em forma de programas de computador. Portanto, um sistema constituído por IA, deve ser capaz de adquirir dados e de os armazenar e deve manipular conhecimento deduzindo e inferindo, assim, novos conhecimentos a partir de conhecimento existente (Shalev-Shwartz e Ben-David, 2014). Existem vários ramos de estudo nesta área, como é possível verificar na figura seguinte.

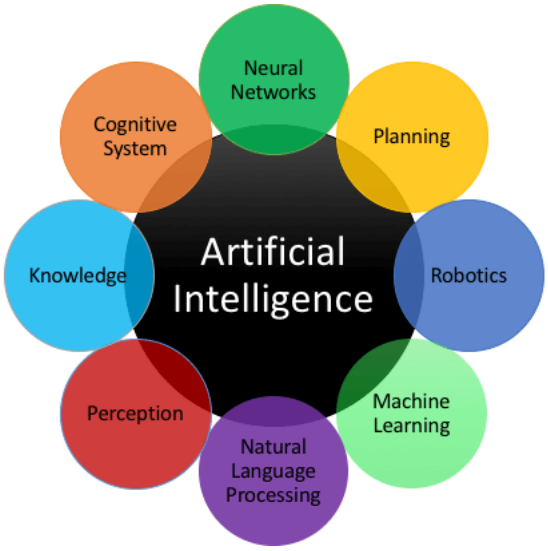


Figura 5: Ramos da inteligência artificial (reproduzido de (Guru, 2016)).

2.1.2 História da IA

Diz a história que, na Grécia Antiga, Aristóteles, professor de Alexandre (o Grande, rei da Macedónia), já pensava em como livrar os escravos dos seus afazeres. Entretanto, vários conceitos de IA foram criados e muita teoria se desenrolou. Contudo, sempre faltou poder de processamento (Messerly, 2015).

A verdade é que grande parte do sucesso dos aliados na Segunda Guerra Mundial deve-se às contribuições de Alan Turing, um génio matemático que hoje é considerado o pai da inteligência artificial e que construiu a 'Máquina de Turing' que possibilitou a descodificação e a quebra dos códigos secretos das comunicações alemãs.

Decorria o ano de 1956 quando, numa conferência no *campus* do Dartmouth College, a inteligência artificial foi definida como a ciência e engenharia de produção de máquinas inteligentes. Na década de 1960, os EUA, com o seu Departamento de Defesa, começaram a treinar computadores de forma a imitar o raciocínio humano. Em 1997, pela primeira vez na história, o Deep Blue, nome dado a um computador da IBM, venceu Garry Kasparov, o melhor jogador de xadrez de todos os tempos. Tal objetivo implica muito poder de processamento. Foi, assim, um grande marco na história da IA. Em 2003, muito antes de Siri, Alexa ou Cortana, a Defense Advanced Research Projects Agency (Darpa) criava assistentes pessoais (Benko e Sik Lányi, 2011).

Esses primeiros trabalhos prepararam o caminho para a automação e o raciocínio formal que se vê nos computadores de hoje, incluindo sistemas de apoio à decisão e sistemas inteligentes de pesquisa que podem ser projetados para complementar e expandir as capacidades humanas.

2.2 *MACHINE LEARNING*

2.2.1 Definição

É um método de análise de dados que automatiza a construção de modelos analíticos. É um ramo da inteligência artificial, baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana. Assim, permite que máquinas possam agir e tomar decisões com base em dados, ao invés de serem explicitamente programadas para tal. Os algoritmos analisam dados de entrada e preveem saídas, tentando diferentes abordagens antes de chegar a um resultado. Passam, também, por duas fases às quais se dá o nome de treino e teste. Na primeira, são alimentados por um conjunto de dados, rotulado ou não, para que sejam contextualizados com o problema. Na segunda fase, começam a deduzir caminhos a partir do que aprenderam na etapa anterior (Murphy, 2013).

2.2.2 Categorias de Aprendizagem

Aprendizagem Supervisionada

O supervisionamento é feito quando existe uma entidade que informa a resposta correta, ou seja, quando há um conjunto de dados rotulados previamente definido. Os rótulos podem ser de dois tipos: classificação quando há um conjunto finito de rótulos, aos quais se dá o nome de classes;

regressão quando se prevê valores reais. De seguida, deseja-se encontrar uma função ou uma forma de prever rótulos desconhecidos (Shalev-Shwartz e Ben-David, 2014).

Aprendizagem Não Supervisionada

Neste tipo de aprendizagem, o conjunto de dados não possui nenhum rótulo. Portanto, o objetivo da aprendizagem passa por descobrir similaridades entre os diferentes objetos e por descobrir padrões nos dados. Assim, um algoritmo deste tipo, cria agrupamentos em que cada grupo/*cluster* contém objetos com características semelhantes (Shalev-Shwartz e Ben-David, 2014).

Aprendizagem por Reforço

A aprendizagem por reforço é também uma categoria de aprendizagem, na qual, a máquina tenta aprender qual é a melhor ação a ser tomada, dependendo das circunstâncias na qual essa ação será executada. Não será usada nesta tese.

2.2.3 Metodologia

A quantidade de dados em crescente e a dificuldade de processamento dos mesmos levou à criação de processos padrão, na indústria, que ajudassem no processo. A metodologia CRISP-DM, que será usada nesta dissertação, foi criada há pouco mais de vinte anos e reúne as melhores práticas para que processos de *data mining* e de *machine learning* sejam o mais produtivo e eficiente possível (Pete et al., 2000).

Deste modo, tendo em conta o descrito anteriormente, o primeiro passo, num processo de ML, é a coleta dos dados que serão estudados e que serão usados para estudar comportamentos e para prever o futuro. É uma etapa muito importante porque a quantidade e a qualidade dos dados determina o quão bom ou quanto potencial tem o modelo a criar. De seguida, é conveniente fazer alguma visualização dos dados de forma a perceber se há algum relacionamento entre as diferentes *features*.

Na maioria das vezes, os dados que se coletam precisam de ajustes e manipulações. Por isso, o passo seguinte consiste em aplicar técnicas como correção de erros, normalização, transformação, entre outros.

Escolhem-se, depois, os modelos de ML mais adequados para resolver o problema em questão. Há, por exemplo, modelos mais adequados para *big data* do que outros e, até mesmo, há modelos que são mais convenientes em problemas de classificação do que em problemas de regressão, ou então, têm melhor desempenho na deteção de padrões em imagens do que na deteção de padrões em textos.

O fluxo acaba com a avaliação dos modelos escolhidos desenvolvidos e com a otimização dos hiperparâmetros dos mesmos.

2.2.4 Classificadores e Modelos de *Machine Learning*

Nesta secção apresentam-se alguns algoritmos de ML, próprios de tarefas de classificação.

Regressão Logística

A regressão linear é um algoritmo capaz de encontrar uma relação entre duas ou mais variáveis. O modelo ajusta uma equação linear entre os dados observados. A linha de melhor ajuste é conhecida como uma linha de regressão, representada na Figura 6.

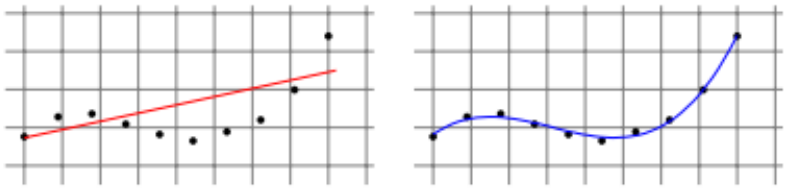


Figura 6: Ilustração da regressão linear (reproduzido de (Shalev-Shwartz e Ben-David, 2014)).

Numa regressão linear, tem-se variáveis de entrada, representadas pela variável x, e uma variável de saída, representada por y. Quando x é uma variável única, o modelo denomina-se por regressão linear simples e quando há múltiplas entradas, designa-se regressão linear múltipla.

A regressão linear, de certo modo, faz a mesma coisa que uma criança, se lhe for pedido que organize as pessoas da sua turma por ordem crescente de peso, sem perguntar às pessoas o seu peso real. Portanto, a criança faz uma análise visual no que diz respeito a altura e largura e ordena-as combinando esses parâmetros. A criança estima que a altura e a largura são variáveis que estão, de facto, correlacionadas com o peso.

A regressão linear é representada pela equação ilustrada na Figura 7.

$$Y = \beta_0 + \beta_{1X}$$

Figura 7: Regressão linear simples.

A regressão logística é semelhante ao modelo de regressão linear. No modelo logístico, a variável dependente y é discreta e admite dois ou mais valores, sendo que cada um destes valores representa uma categoria. Entende-se, então, como um modelo análogo à regressão linear para problemas de classificação. Prevê a probabilidade da ocorrência de um evento ajustando os dados

de uma função logística e, por isso, é conhecida como regressão logística. Os valores de saída esperados estão entre 0 e 1.

Partindo do pressuposto que um amigo dá a outro seu amigo um enigma para ser resolvido, então são esperados dois cenários: o enigma é resolvido e o enigma não é resolvido. Pense-se, também, num cenário em que há um amplo grupo de enigmas a serem resolvidos por um médico de profissão. Pode-se dizer, então, que, por exemplo, se lhe é dado um enigma relacionado com medicina, há uma probabilidade de 90% de o resolver e se lhe é dado um enigma relacionado com matemática, há uma probabilidade de 50% de o resolver. Essa probabilidade é o que a regressão logística fornece, ou seja, é a sua variável de saída. Aliás, a variável de saída da maioria dos algoritmos de ML é essa.

A variável dependente é calculada em relação a uma razão, denominada razão de chance. A razão de chance é uma razão entre a probabilidade de estar num grupo dividida pela probabilidade de estar num outro grupo.

Naïve Bayes

É um modelo probabilístico usado em tarefas de classificação. O cerne do classificador é baseado no teorema de Bayes.

O teorema de Bayes descreve que a probabilidade de um acontecimento A acontecer, sabendo que o acontecimento B aconteceu, corresponde à probabilidade de um acontecimento B acontecer, sabendo que A aconteceu a multiplicar pela probabilidade do acontecimento A acontecer e a dividir pela probabilidade de B acontecer.

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

Figura 8: Teorema de Bayes.

Partindo deste princípio, define-se o acontecimento B como a evidência e o acontecimento A como a hipótese e, assim, as *features* são independentes e a presença de uma não afeta a outra. Daí resulta o nome dado ao algoritmo, ingénuo (*naive*). O teorema pode ser reescrito como:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Figura 9: Teorema de Bayes transformado.

A variável y é a variável que se pretende prever (*target*) e a variável X representa o conjunto de *features*. Expandindo a variável X, obtém-se a fórmula representada na Figura 10 que ilustra, simplificadamente, o modelo.

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Figura 10: Fórmula simplificada do modelo Naive Bayes.

Redes Neuronais

Uma grande parte da investigação em RNAs, ao longo dos tempos, inspirou-se no sistema nervoso do ser humano, já que o cérebro humano é uma máquina altamente complexa capaz de processar grandes quantidades de informação em tempo extremamente reduzido. Apesar das vastas investigações no seu redor, o conhecimento em relação ao modo como este opera está longe de estar completo e sabe-se apenas que o cérebro adquire as suas próprias regras conforme as experiências vividas. Vários cientistas, ao longo dos anos, tentaram simular o funcionamento do cérebro, nomeadamente a forma como este aprende através da experiência, com o objetivo de criar sistemas inteligentes capazes de se comportarem de maneira similar.

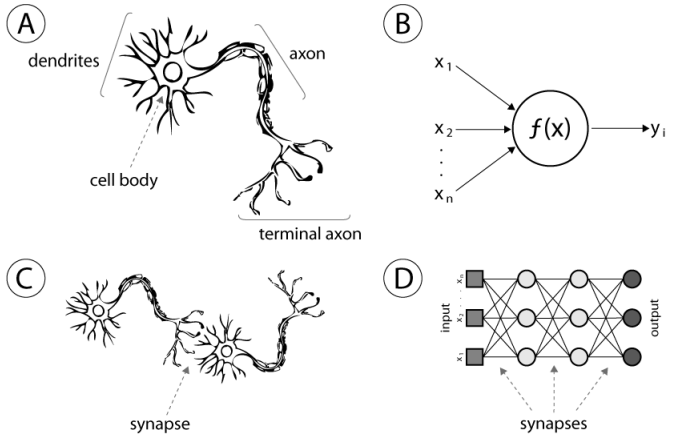


Figura 11: Comparação entre neurónios e neurónios artificiais (reproduzido de (Bhargavi e Jyothi, 2016)).

As redes neuronais artificiais (RNAs) são modelos computacionais baseados no sistema nervoso central do ser humano. São estruturas extremamente interconectadas de unidades computacionais, designadas neurónios ou nodos, com capacidade de aprendizagem. Estas redes, tal como o cérebro humano, apresentam a aptidão para adquirir conhecimento a partir do ambiente, com um processo de aprendizagem, e apresentam, também, a capacidade de armazenar o conhecimento nas conexões dos neurónios (Cortez e Neves, 2000).

Um neurónio é uma célula do sistema nervoso que responde a sinais eletroquímicos, sendo composto por um núcleo, por um conjunto de dendrites que recebem sinais de outros neurónios via sinapses e ainda um axónio que tem a função de transmitir um sinal a outros neurónios. Num cérebro humano existem estruturas complexas destas entidades interligadas de diferentes maneiras. Na informática um nodo, ou neurónio artificial, é a unidade mais importante no funcionamento de uma RNA, sendo composto por um conjunto de conexões indexadas com um peso (w_i), podendo

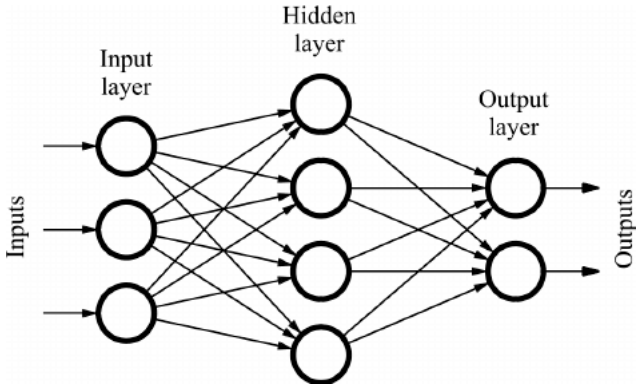


Figura 12: Rede neuronal artificial (RNA) (reproduzido de (Cortez e Neves, 2000)).

estas ter um papel excitatório ou inibitório, um integrador ou valor de ativação (P), que transforma os vários valores de entrada num único valor e ainda uma função de ativação (σ) que tem a função de condicionar o sinal de saída (Cortez e Neves, 2000).

A propriedade mais importante das RNAs é a sua capacidade de aprender a partir do seu ambiente. Assim, a RNA recebe estímulos do ambiente, isto é, recebe dados de treino; os pesos das sinapses são alterados em resultado deste estímulo; uma RNA é capaz de responder ao ambiente graças às alterações na sua estrutura interna. Esta aprendizagem é realizada consoante um algoritmo de treino selecionado previamente que depende da forma como a rede se relaciona com o meio, ou seja, depende do paradigma de aprendizagem utilizado (supervisionado, no caso desta tese). Dentro dos diversos paradigmas, existem diversos algoritmos de treino. O mais usado, atualmente, é o *backpropagation* que faz parte dos paradigmas supervisionados e baseia-se na alteração dos pesos das sinapses. Este algoritmo utiliza dois passos: **Em frente**, onde o vetor de entrada é propagado ao longo da rede até à última camada e no fim é calculado um erro em função do resultado obtido e o desejado. Nesta fase os pesos não são alterados; **Retro propagação**, onde o erro é propagado desde a última camada até à inicial segundo algumas funções de alteração dos pesos das sinapses (Cortez e Neves, 2000).

Árvores de Decisão

É um método supervisionado não-paramétrico. Baseia-se em árvores que, de modo geral, na computação, são estruturas de dados formadas por conjuntos de elementos que armazenam informações, aos quais se dá o nome de nós. Todas as árvores possuem um nó, denominada raiz, que possui o maior nível hierárquico e que constitui o ponto de partida para todos os outros elementos, denominados filhos. Esses filhos podem também possuir filhos. Os nós que não possuem filhos são conhecidos como nós folha.

Uma árvore de decisão é uma árvore que armazena testes numa variável em cada nó de decisão, possíveis valores de uma variável nos ramos descendentes e classes nos nós folha. Portanto, cada percurso na variável corresponde a uma regra de classificação. Deste modo, a utilizar uma

estratégia de dividir para conquistar, consegue representar um mapa dos possíveis resultados de uma série de escolhas tomadas. A escolha de qual variável selecionar em cada momento, ou seja, para dividir a população em grupos diferentes e heterogéneos, é feita usando técnicas como a de Gini, cálculos de ganho de informação, qui-quadrado e entropia.

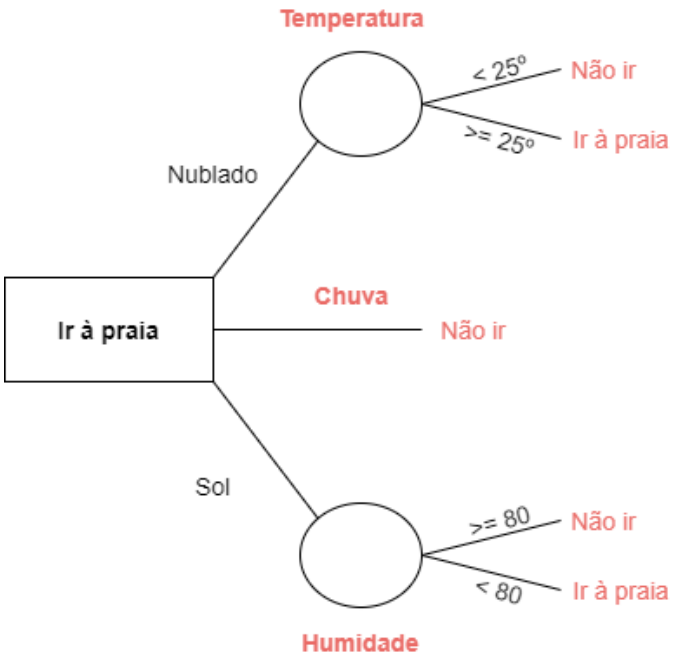


Figura 13: Exemplo ilustrativo de uma árvore de decisão.

Modelos *Ensemble*

A aprendizagem *ensemble* ou em conjunto combina as decisões de múltiplos modelos, de forma a melhorar o desempenho geral de uma classificação. Buscam, deste modo, minimizar as principais causas de erros (ver Figura 14) nos modelos de aprendizagem, ou seja, o ruído, o *bias* e a variância. As técnicas usadas são:

- ***Bagging***: reduz a variância e, por isso, pode conseguir contornar problemas relacionados com *overfitting*. Os modelos de árvores de decisão, por exemplo, introduzem uma variância elevada. De modo a contornar isso, usam-se pequenos conjuntos do *dataset*, e para cada um, treina-se um modelo do mesmo tipo. As previsões finais resultam da agregação ou votação de todas as sub-previsões.

  - Random Forest: é um algoritmo baseado em árvores de decisão que aplica *bagging*. Portanto, muito simplificadamente, constrói múltiplas árvores de decisão e, no fim, junta todas, de forma a obter uma previsão mais estável e eficaz (Breiman, 2001).

  Neste tipo de modelos, alguns hiperparâmetros que se costumam ter em conta são, por exemplo, o número de estimadores, ou seja, o número de árvores a construir (quanto maior

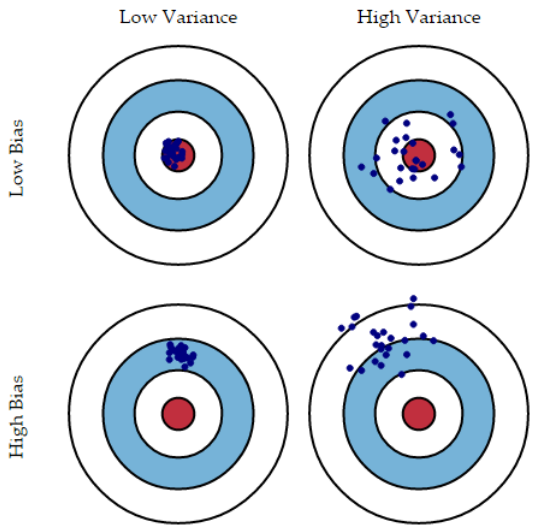


Figura 14: Erros comuns em tarefas de ML (reproduzido de (Bhalla, 2017)).

for o valor, melhor), ou a percentagem de variáveis a usar, já que nem sempre usar todas as variáveis é recomendado, ora por questões de perda de diversidade, ora por decréscimo na velocidade de treino dos algoritmos.

- ***Boosting***: reduz o *bias* superando problemas de *underfitting*. É uma técnica iterativa que ajusta o peso de uma observação a partir da última classificação atribuída. Portanto, se uma observação foi classificada erradamente, é-lhe atribuído um peso maior e vice-versa. Contudo, pode originar problemas de *overfitting*.

  - LightGBM: (*Light Gradient Boosting Model*) é um algoritmo baseado em árvores de decisão que aplica um *boosting* de gradiente. As vantagens que traz face a modelos semelhantes são a velocidade e eficiência no processo de treino, pouco uso de memória, suporte para *big data* e bons resultados (Ke et al., 2017).
  - XGBoost: (*EXtreme Gradient Boosting*) é também um algoritmo que se baseia em árvores de decisão onde é aplicado um *boosting* de gradiente. Ao contrário do anterior, não é tão rápido e tão eficiente ao nível da memória. Contudo, consegue ser mais regular no que a resultados diz respeito (Chen e Guestrin, 2016).
  - AdaBoost: (*Adaptative Boosting*) é um algoritmo onde se aplica *boosting*, usando árvores de decisão (Schapire, 1999).

Em modelos de *boosting*, os parâmetros importantes são o número de iterações, que corresponde ao número de árvores a construir, a profundidade máxima de cada árvore, e o número de folhas. Quanto mais elevado for a definição dos três parâmetros referidos, maior é o desempenho, embora que possa haver *overfit* e um aumento no tempo de treino. Mais

uma vez, a fração de variáveis/*features* a usar é também uma definição e um parâmetro importante.

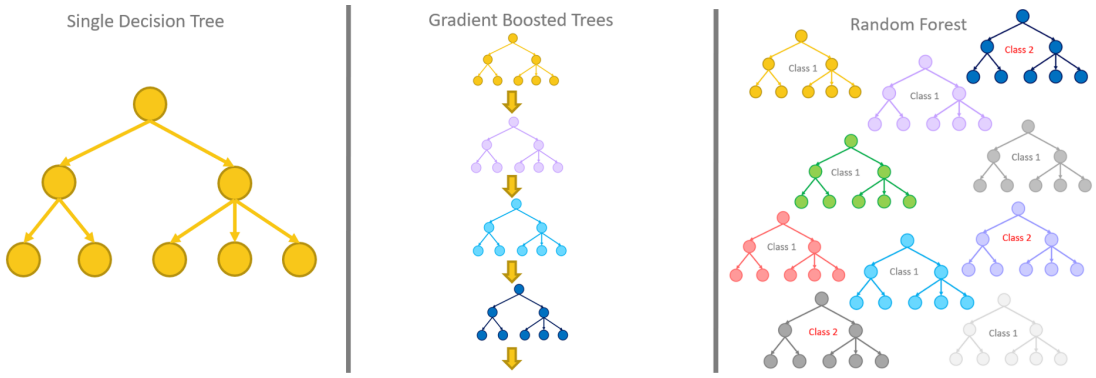


Figura 15: Exemplo gráfico de árvore de decisão, *boosting* e *bagging* (reproduzido de (Yaswanth, 2019)).

- ***Voting***: o *voting classifier* é uma forma simples de combinar as previsões de múltiplos algoritmos de *machine learning*, através de um sistema de votos. Para o fazer, múltiplos modelos são treinados e as probabilidades retornadas por cada modelo individual são combinadas. Supondo que três modelos originam as probabilidades $P(x) = [0.45, 0.45, 0.90]$, a combinação pode ser feita de duas maneiras:

  - *Hard-Voting* - Há 1 voto a favor (0.90) e 2 votos contra (0.45 e 0.45), por isso, resulta em $1/3$ (33%) de probabilidade, classificando a observação como negativa;
  - *Soft-Voting* - Calcula a média das probabilidades, o que resulta em 0.6 (60%), classificando a observação como positiva.

- ***Stacking***: o *stacking classifier* envolve, também, a combinação das previsões de múltiplos modelos de *machine learning*. Aborda a questão relativa a qual modelo usar, em determinadas situações, visto que alguns são hábeis num problema e outros são hábeis noutro problema. Por isso, utiliza-se um outro modelo de *machine learning* que aprende quando utilizar ou confiar em cada modelo. Para isso, definem-se dois níveis de modelos:

  - Modelos de Base - modelos que são treinados individualmente, e dos quais são extraídas as probabilidades de cada classe, para cada observação;
  - Meta-Modelo - modelo que aprende a melhor forma de combinar as previsões previamente calculadas por cada modelo.

  O meta-modelo é treinado com as previsões feitas pelos modelos de base e com os resultados esperados.

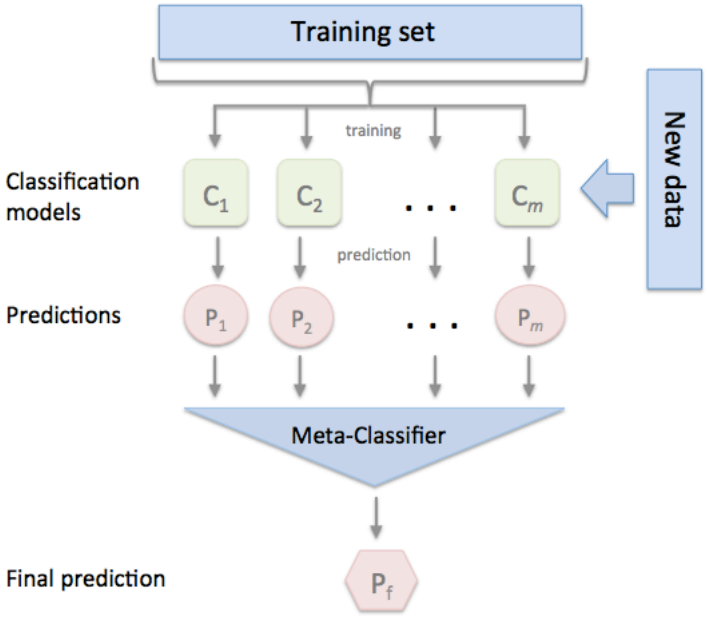


Figura 16: Arquitetura do *Stacking Classifier* (reproduzido de (Raschka, 2018)).

2.2.5 Avaliação de Modelos e Diferentes Métricas

A capacidade de avaliação dos diferentes modelos pressupõe um prévio conhecimento de alguns conceitos estatísticos, que serão explicados na presente secção.

Quando o modelo prevê um caso positivo corretamente, tem-se um caso verdadeiro positivo (TP) e quando o modelo prevê um caso positivo incorretamente, tem-se um caso falso positivo (FP), também conhecido como erro de tipo I. O oposto pode também ocorrer e quando o modelo prevê um caso negativo corretamente, tem-se um caso verdadeiro negativo (TN). Finalmente, quando prevê um caso negativo incorretamente, denomina-se falso negativo (FN), que corresponde a um erro de tipo II.

Tabela de Confusão

É uma matriz de confusão, geralmente com o tamanho de 2x2 (pode ser maior dependendo do número de classes). A métrica é voltada para modelos de classificação e tem como objetivo a visualização, essencialmente, do número de verdadeiros positivos e negativos e de falsos positivos e negativos.

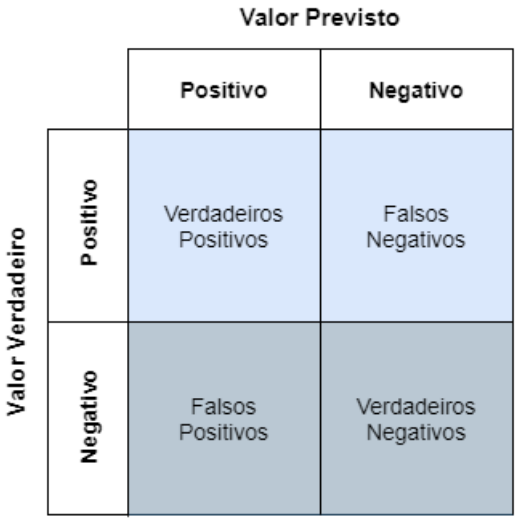


Figura 17: Matriz de confusão.

Accuracy

É o rácio entre o número de previsões corretas (verdadeiros positivos e verdadeiros negativos) e o número total de observações, como mostra na Figura 18.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Figura 18: Fórmula da *accuracy*.

Contudo, só é uma boa métrica se há um número igual de amostras para todas as classes, ou seja, só deve ser usada perante um conjunto de dados balanceado. Por exemplo, considerando que se quer construir um modelo que classifica pessoas, num aeroporto, em normais ou terroristas. Considerando-se, também, que há 98% de amostras para a classe normais e 2% para a classe terroristas num conjunto de treino qualquer. Então, o modelo conseguirá obter os 98% de *accuracy* em treino simplesmente a prever que todas as pessoas são normais. Conseguirá, também, obter os 60% de *accuracy* em teste, se o conjunto de teste tiver 60% de amostras para a classe normais e 40% para a classe terroristas. Ou seja, apesar de não parecer, porque a métrica não indica, um modelo que tenha estes resultados, é um mau modelo. Em situações de desbalanceamento, pode dar uma falsa sensação que o modelo é bom quando poderá não o ser. O problema torna-se grave, ainda mais, quando se está perante situações em que o custo de uma classificação incorreta das amostras da classe minoritária é muito alto. Ou seja, associando os 98% de *accuracy* a uma pessoa saudável e os 2% a uma pessoa com uma doença rara e fatal, o custo de não diagnosticar

a pessoa doente corretamente, é maior que o custo de enviar uma pessoa saudável para testes mais detalhados.

Precisão

A precisão é o rácio entre positivos verdadeiros e a soma entre positivos verdadeiros e positivos falsos. É, por isso, uma métrica que se foca nos erros relacionados com falsos positivos. Portanto, a métrica diz quantas observações estão realmente certas, daquelas que foram classificadas como certas.

Sensibilidade

O *recall*, também conhecido como sensibilidade, representa a frequência com que o classificador encontra exemplos de uma classe. É o rácio entre positivos verdadeiros e a soma entre verdadeiros positivos e falsos negativos. A métrica interpreta-se como a fração de observações positivas que foi corretamente prevista como positiva. Assim, foca-se nos erros relacionados com falsos negativos.

F1

É a média harmónica entre a precisão e a sensibilidade. Fornece uma medida mais correta dos casos classificados incorretamente do que a métrica de *accuracy*. Perante um dataset desbalanceado, o seu uso é, por isso, recomendável.

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision} + \text{Recall})}$$

Figura 19: Formula de cálculo da métrica F1.

A F1 pode, também, ser distinguida entre *macro*-F1 e *micro*-F1. Quando é *macro*, cada classe tem o mesmo peso na avaliação final. Quando é *micro*, cada classe tem o seu peso definido pela distribuição.

Curva de ROC

A curva AUC-ROC é uma medida de desempenho para problemas de classificação em várias definições de *threshold*. ROC (*Receiver Operating Characteristic*) é uma curva de probabilidade e AUC representa o grau ou medida de separabilidade, ou seja, informa o quanto o modelo pode fazer distinção entre as diversas classes. Quanto maior for o valor de AUC, melhor o modelo está a prever os positivos como positivos e os negativos como negativos. A curva é traçada no sentido em

que o rácio de positivos verdadeiros corresponde ao eixo yy e o rácio de positivos falsos corresponde ao eixo xx (Murphy, 2013).

Curva PR

A curva *precision-recall* (PR) é semelhante à curva de ROC, com a diferença de ser o resultado do gráfico entre a precisão e o *recall*. Também nesta curva é possível obter o AUC-PR, ou seja, a área debaixo da curva, de forma a avaliar e comparar o desempenho do modelo.

Validação Cruzada

É uma técnica que mostra de que forma é que o modelo generaliza, ou seja, como é que se comporta perante dados que nunca viu. É uma alternativa ao método de retirada (*holdout set*), no qual se remove uma parte dos dados de treino e usam-se, estes, para testar os diferentes modelos. Este método sofre demasiada variância porque não é certo que dados acabam por ficar no conjunto de treino e, para conjuntos diferentes, o desempenho do modelo pode ser totalmente diferente. Ou seja, corre-se o risco de perder padrões e tendências importantes o que pode introduzir *bias*. Portanto, relativamente à validação cruzada, há dois grandes tipos:

- ***K-Fold***: os dados são divididos em k subconjuntos e o método de validação é repetido k vezes, de modo que, a cada iteração, um dos subconjuntos k é usado como conjunto de teste e os restantes subconjuntos são reunidos para formar um conjunto de treino. Reduz significativamente o *bias* porque usa-se a maioria dos dados;

- ***Stratified K-Fold***: o *K-Fold* não tem em conta o *target* quando divide o conjunto de dados em diferentes subconjuntos. Assim, neste método, há uma pequena variação relativamente ao anterior, na medida em que se certifica que cada subconjunto contém a mesma percentagem de observações de cada classe;

2.2.6 Otimização dos Hiperparâmetros

Nos modelos de ML há dois tipos de parâmetros: os do modelo que são aprendidos por este na fase de treino, como é o caso dos pesos, e os hiperparâmetros que são definidos pelo cientista de dados. Ou seja, podem ser vistos como configurações que precisam de ser ajustadas de maneira diferente para cada problema já que o melhor hiperparâmetro de modelo para um conjunto de dados em particular não é o melhor em todos os conjuntos de dados. Em suma, a otimização de hiperparâmetros consiste em encontrar a melhor combinação de valores de configuração para um modelo de ML.

Grid Search

O *grid search* pode ser descrito como um método de adivinhação e verificação exaustiva. Define-se um conjunto de valores (domínio) que são mapeados numa grelha hiperparamétrica. O método tenta todas as combinações de valores no domínio. A melhor combinação de valores nem sempre é encontrada através do *grid search* já que só são usados os valores contidos no domínio. Tem uma limitação que é o facto de ser caro em termos computacionais, ao exigir uma validação cruzada a cada combinação de hiperparâmetros. Isto quer dizer que o domínio usado não pode ser grande, pois exige muito tempo para ser executado.

Random Search

O *random search* é, geralmente, mais eficiente em comparação com o *grid search*. Isto é, embora o *grid search* consiga encontrar o melhor conjunto de hiperparâmetros (se estiverem no domínio), o método aleatório encontra um valor também muito próximo desse, em muito menos iterações. Isto acontece porque a pesquisa em grelha perde demasiado tempo em regiões não muito promissoras do espaço de pesquisa, já que tem de avaliar todas as combinações. O *random search*, ao contrário, consegue fazer uma exploração mais eficiente.

Manual Search

É um método que se baseia na seleção dos diferentes hiperparâmetros com base na intuição e na experiência do programador. Melhor dizendo, treina-se um modelo com os hiperparâmetros que se julgam estar corretos e verifica-se o desempenho. O processo é repetido até o cientista de dados ficar satisfeito com os resultados obtidos.

Otimização *Bayesiana*

A otimização *bayesiana* ajusta os hiperparâmetros com base num problema de regressão. Assim, dado um conjunto de hiperparâmetros, de forma a resolver o problema faz-se suposições sobre quais combinações têm mais probabilidade de obter os melhores resultados. O modelo é, então, executado, e a regressão é usada para escolher o próximo conjunto de hiperparâmetros a testar, usando tudo o que sabe sobre o problema até ao momento. A próxima combinação, por vezes, é próxima daquela que resultou no melhor resultado até ao momento e, por vezes, consegue-se melhorar o desempenho de forma incremental. Outras vezes, escolhe valores bem diferentes daqueles que tentou anteriormente, o que permite explorar todo o intervalo possível de valores (Snoek et al., 2012).

2.3 PRÉ-PROCESSAMENTO DOS DADOS

A preparação dos dados consiste em transformar os conjuntos de dados, para que a informação neles contida esteja adequadamente exposta às ferramentas de ML e de extração de conhecimento. Assim, os dados devem ser formatados às ferramentas, já que se foram recolhidos do mundo real, podem estar incompletos, podem conter lixo e podem, sobretudo, ter inconsistências.

2.3.1 Visualização dos Dados

O mundo é inerentemente visual. Aprende-se, desde muito cedo, que as imagens valem e falam mais alto do que as palavras. E, hoje em dia, as empresas coletam enormes quantidades de dados que, posteriormente, são alvo de tratamento por parte de projetos de *big data* e por projetos relacionados com análise de dados. Uma forma de o fazer com eficácia e com rapidez é compreender e explicar os dados que têm em posse de uma maneira que faça sentido. Portanto, a visualização de dados, como uma técnica que permite colocar dados num contexto visual, como um mapa ou um gráfico, facilita a compreensão de dados grandes pelo cérebro humano e facilita na colocação de um significado. Permite, também, ajudar na deteção de padrões, tendências e *outliers*. A visualização de dados é, de facto, importante em todas as áreas, desde professores que tentam entender os resultados dos testes dos seus alunos até aos cientistas da computação que tentam desenvolver algum projeto em inteligência artificial.

Alguns dos principais gráficos usados são:

CountPlot gráfico de barras que mostra a contagem dos diferentes valores que surgem numa coluna;

BarPlot gráfico de barras com eixo xx e eixo yy. Permite analisar a relação entre duas variáveis;

DistPlot gráfico importante para entender a distribuição dos dados, através de um histograma que é um gráfico de barras para variáveis quantitativas, que são divididas em intervalos. Mostra a frequência de dados que se tem em cada intervalo;

BoxPlot mais conhecido por caixa de bigodes. É um tipo de gráfico com muita informação sobre os dados, já que permite ver o limite inferior e superior, os quartis, a mediana e os possíveis *outliers*;

2.3.2 Limpeza dos Dados

Na etapa de limpeza de dados realizam-se diversas tarefas que se revelam de imensa importância para um bom desempenho dos modelos a executar posteriormente. É comum, portanto, observar alguns problemas nos dados, tais como:

- Inconsistências;
- Dados não registados e nulos;
- Dados registados de forma errada;
- Análise errada;

É, principalmente a ausência de dados que gera os maiores problemas de resolução no sentido em que nem sempre é óbvio qual a decisão a tomar. Ou seja, face a registos onde faltam dados, pode-se ignorá-los, não sendo aconselhável se a quantidade de dados em falta em cada atributo for elevada; pode-se ignorar as variáveis onde faltam os dados, não sendo aconselhável se as variáveis onde tal acontece forem de extrema importância; pode-se preencher os dados em falta, manualmente; pode-se criar tendências nos dados, preenchendo com o valor médio, por exemplo, não havendo um grande impacto negativo se o desvio padrão não for grande; pode-se preencher com o valor mais frequente do atributo. Como é de esperar, quantos mais valores são inventados, maior é o desvio dos dados relativamente à realidade que o problema ilustra.

2.3.3 Integração dos Dados

De forma a desenvolver um modelo que seja provido e dotado da maior quantidade de dados possível, é comum reunir informações armazenadas em diferentes fontes de dados. Esta etapa constitui, portanto, a integração dos dados de diferentes *datasets* num só. Apesar de tudo, é uma etapa que requer a deteção e a solução de conflitos entre os dados. Além disso, exige conhecimento do negócio.

2.3.4 Transformação dos Dados

A transformação dos dados relaciona-se com:

1. **Alisamento**
    - *Clustering* - Também conhecida como análise de agrupamento de dados, é o conjunto de técnicas que visa fazer agrupamentos automáticos de dados, a partir de um grau de semelhança. A cada conjunto de dados resultante deste processo dá-se o nome de grupo, agrupamento ou *cluster*;
    - *Binning* - A técnica consiste em fazer discretização ou enumeração dos dados, ou seja, reduzir o número de valores de um atributo contínuo, dividindo-o em intervalos. Pode-se fazer usando discretização de igual largura, que divide a gama de valores em N intervalos de igual largura, resultando numa grelha uniforme, mas que pode resultar numa distribuição diferente dos dados e pode-se fazer usando discretização

de igual altura que divide a gama de valores em N intervalos, contendo, cada um, aproximadamente a mesma quantidade de valores. A discretização de igual altura é, quase sempre preferível em detrimento da discretização de igual largura porque evita o amontoar de valores e impede a dispersão de valores frequentes por diferentes intervalos.

2. **Agregação**

    - Juntar dados em grupos (ex: vendas trimestrais, vendas nos últimos 5 anos, etc);

3. **Generalização**

    - Usar categorias superiores em dados hierarquizados (ex: usar cidade em vez da rua);

4. **Codificação de variáveis categóricas**

    - *Ordinal Encoding* - a cada valor único da variável é atribuído um valor inteiro.
    - *One-Hot Encoding* - garante que os modelos não pressuponham que os números mais altos são mais importantes, o que acontece no *ordinal encoding*. Neste tipo de codificação, a cada valor único da variável é atribuída uma nova coluna que pode assumir o valor 0 ou 1.
    - *Target Encoding* - para cada valor único na variável, calcula-se a média dos valores correspondentes na variável target (y). É atribuído, então, esse valor calculado. Como é dependente da variável y, algum *overfit* pode surgir.

2.3.5 Normalização e Padronização

As duas técnicas, normalização (*MinMaxScaler* no *sklearn*) e padronização (*StandardScaler* no *sklearn*), têm como objetivo transformar todas as variáveis para a mesma ordem de grandeza (Buitinck et al., 2013).

Assim, normalizar as variáveis, também conhecido como *feature scaling*, coloca todos os valores dentro de um intervalo de 0 a 1. Por outro lado, padronizar os dados significa fazer uma transformação que resulta numa média igual a zero e um desvio padrão igual a um.

Se a distribuição dos dados não é gaussiana ou quando o desvio padrão é muito pequeno, normalizar os dados é uma boa escolha a ser tomada. A padronização dos dados origina perda dos *outliers* e gera uma mudança que pode ser significativa na distribuição dos dados.

O *RobustScaler* do *sklearn* é também uma opção bastante usada no momento, visto que é um método similar à normalização de dados, mas fazendo uso de uma amplitude interquartil em vez de uma amplitude 'min-max'. Desta forma, torna-se robusto para *outliers* e o seu uso pode ser apropriado quando os há num determinado conjunto de dados (Buitinck et al., 2013).

2.3.6 Balanceamento dos Dados

O balanceamento dos dados é uma tarefa bastante valorizada na área, na medida em que permite ajudar os modelos a tomarem uma decisão que não enviesada. Ou seja, perante um conjunto de dados que tem mais observações de uma classe do que de outra, os modelos de ML são influenciados a escolherem pela classe que tem mais peso e predominância. Algumas das técnicas usadas que fazem o equilíbrio dos dados são:

SMOTE A abordagem adotada neste método é a de gerar observações sintéticas e artificiais para a classe que se pretende aumentar, a partir das já existentes. A geração é feita na vizinhança de cada caso da classe minoritária e, com isto, o espaço de decisão da classe cresce e o poder de generalização dos classificadores pode também aumentar (Chawla et al., 2002);

Random Over-Sampling Duplicação de observações aleatórias da classe minoritária;

Random Under-Sampling Remoção de observações aleatórias da classe maioritária.

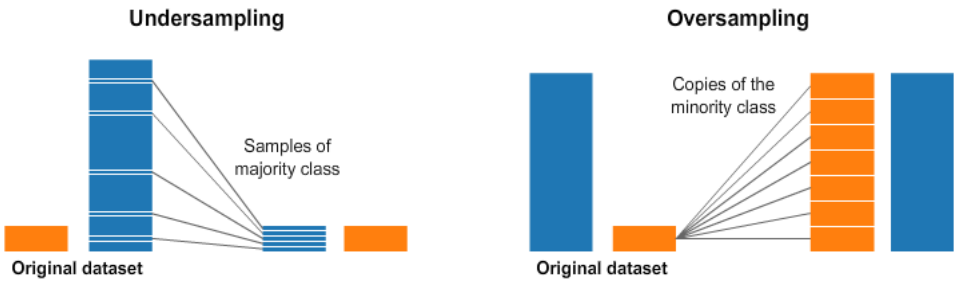


Figura 20: *Random Over-Sampling* e *Random Under-Sampling* (reproduzido de (Alencar, 2017)).

2.3.7 Variáveis

Seleção de Variáveis

A seleção de variáveis ou *features* ocupa um papel muito importante em ML. Há, de facto, colunas irrelevantes ou menos importantes, nos conjuntos de dados, que não contribuem em larga medida para o *target*, a variável a prever. Isso pode afetar negativamente o desempenho do modelo. Algumas das vantagens em o fazer são:

- Ajuda a evitar o *overfitting*, porque resulta em menos dados redundantes;
- Pode aumentar a *performance* dos modelos, já que a aprendizagem se concentra apenas nos dados com significado;
- Reduz o tempo de execução e o uso de memória, já que há menos dados para processar;

Há três métodos diferentes de seleção de *features*, que se distinguem pela quantidade de computação usada e pela forma como fazem a filtragem que leva à escolha das *features* mais significativas para o problema a resolver. Os métodos por filtro baseiam-se nas propriedades intrínsecas e nas relevâncias das *features*. Portanto, fazem uso de métricas estatísticas.

Por outro lado, os métodos de *wrapper* medem a utilidade das *features* com base no desempenho dos classificadores.

Por último, nos métodos *embedded*, o algoritmo de selecção de variáveis é integrado como parte dos algoritmos de aprendizagem.

Extração de Variáveis

Uma característica dos grandes conjuntos de dados é haver um grande número de variáveis que requerem muitos recursos computacionais para serem processadas. Ao fazer-se a extração das variáveis, selecionam-se e combinam-se variáveis ou múltiplas variáveis de forma a reduzir efetivamente a quantidade bruta de dados, mas, ao mesmo tempo, mantêm-se os dados que ainda assim descrevem de forma precisa e completa o conjunto original de dados. Isso permite:

- Melhorias na *performance* dos modelos;
- Redução do risco de *overfitting*;
- Treino mais rápido;
- Visualização de dados facilitada;
- Aumento da explicabilidade do modelo.

2.4 OTIMIZAÇÃO E FILAS DE ESPERA

As filas de espera são um fenómeno que ocorre, frequentemente, no dia-a-dia. Verifica-se a aplicação de filas, principalmente, em prestações de serviços, mas também em fluxo de tráfego, seja ele relacionado com transporte de pessoas ou até de comunicações.

Existem, portanto, clientes que pretendem prestações de serviços e, que para isso, podem ter de esperar em filas físicas ou concetuais, e servidores que prestam serviços. Os clientes chegam à fila com uma determinada taxa de chegada e o servidor demora algum tempo a atender cada cliente, o denominado tempo de serviço.

O congestionamento que acontece nas filas e que leva os clientes a esperarem demasiado tempo só é aceitável quando o custo do servidor é maior que o custo de espera do cliente. Isso pode ser

desejável em relação, por exemplo, a bombeiros, mas é indesejável, por exemplo, num supermercado. Então, a teoria das filas de espera otimiza o funcionamento já que permite encontrar soluções equilibradas entre os dois extremos.

A teoria das filas de espera otimiza o funcionamento das filas, já que encontra soluções equilibradas entre os dois extremos.

Um sistema de filas de espera é constituído por cinco elementos fundamentais: a população, a fila de espera, o serviço, a capacidade do sistema e a própria disciplina usada no atendimento. Portanto, para se estabelecer um sistema deste tipo é preciso ter em conta a dimensão da população, que pode ser finita ou infinita, a dimensão da chegada que caracteriza se os clientes chegam um a um ou em grupo, o controlo das chegadas que define se as chegadas são planeadas ou não previamente, a distribuição das chegadas, a taxa de chegada e a atitude dos clientes, que pode ser paciente ou não.

Segundo (Muller, 2007), em 1953, o matemático inglês David George Kendall (1918 – 2007) propôs a seguinte notação para representar uma fila de espera: $A/S/m/K/N/Q$ onde:

- A é a distribuição dos tempos entre as chegadas (processo de chegadas);
- S é a distribuição dos tempos de serviço (processo de atendimento);
- m é o número de servidores ($m \in N$);
- K é a capacidade do sistema ($K \in N \bigcup +\infty$);
- N é o tamanho da população ($K \in N \bigcup +\infty$);
- Q é a disciplina de atendimento.

A distribuição exponencial é, normalmente, representada por M (de Memoryless), a distribuição de Erlang é representada por Er e a distribuição genérica é representada por G. Muitas vezes, os três últimos símbolos são omitidos. Nestes casos, assume-se capacidade ilimitada, população infinita e disciplina de atendimento FCFS (Pereira, 2009).

Nesta tese, de agora em diante, será usada a notação anteriormente descrita para referir modelos de filas de espera.

O apoio ao cliente da empresa e, portanto, o seu call-center, tipicamente, pode ser representado por um sistema de filas de espera, próprio da teoria de filas de espera. Assim, alguns dos seus constituintes mostram-se na Tabela 1.

Resumidamente, não há uma distribuição inteligente das chamadas e, qualquer operador pode ser confrontado com todos os problemas. Ou seja, assumindo capacidade ilimitada, população infinita e disciplina de atendimento FCFS/FIFO, pode-se denominar o sistema por fila M/M/S.

Nesta tese propõe-se uma solução que permita melhorar o serviço e/ou reduzir custos oferecendo o mesmo préstimo. Essa melhoria pode ser conseguida reduzindo o tempo médio de serviço e, até mesmo, reduzindo o número de servidores necessários em operação a cada momento.

Chegadas	
Dimensão da População	Infinita, porque podem ligar pessoas que não são clientes
Dimensão da Chegada	Unitária
Controlo das Chegadas	Incontrolável
Distribuição das Chegadas	Aleatórias
Atitude dos Clientes	Impaciente
Fila de Espera	
Número de Filas	Única simples
Comprimento da Fila	Infinito
Disciplina da Fila	Cliente entra na fila e os seguintes ficam atrás de si
Serviço	
Dimensão de Serviço	Simples, porque cada servidor atende uma pessoa de vez
Distribuição do Tempo de Serviço	Aleatório
Capacidade do Sistema	Infinita
Disciplina do Atendimento	FIFO

Tabela 1: Informação detalhada do call-center da empresa.

2.5 TRABALHOS RELACIONADOS

(Ali, 2011) referiu que, dentro da temática dos call-centers, não é possível treinar todos os operadores de chamada de maneira a que consigam lidar e dar resposta, tecnicamente ou não, a todo o tipo de chamadas. Neste sentido, diz o autor que, normalmente e de forma a combater-se isso, aplica-se a técnica *skill based routing* (SBR), na qual os atendedores de chamadas são divididos em grupos, com conhecimentos distintos, aos quais chegam chamadas com problemas associados e aos quais sabem dar uma resposta transparente e eficiente. Defende, contudo, uma solução melhor que consiga reduzir a complexidade inserida pela solução clássica referida anteriormente e que permita, de igual forma, aumentar vendas, melhorar a satisfação do cliente e acima de tudo, reduzir a média de duração das chamadas.

Na solução proposta, as chamadas para o apoio ao cliente vão ao encontro de vendas por telefone e, pretende-se maximizar um *score*, que é diretamente relacionado com uma venda, ou seja, se o produto é vendido a um preço alto e com uma boa satisfação do cliente, então o *score* é positivo. Começa por definir uma metodologia constituída por dois componentes principais, o *Switch* e o *Mapper*. O primeiro é o responsável por identificar os clientes que estão à espera na fila e os operadores disponíveis para atender a chamada e o segundo trata de extrair todas as variáveis relacionadas com os clientes e com os operadores das diferentes bases de dados. Ao *Mapper*, é também atribuída uma função de *Predictor* e uma função de *Optimizer*.

Inicialmente, a partir de dados históricos e de informações relativas a cliente e a operador, constrói um conjunto de dados que será usado para treinar um conjunto de modelos de forma *offline*. Esse conjunto de dados passa, depois, por um pré-processamento onde se usa *feature scaling*, reduções de dimensionalidade, eliminação de variáveis inúteis para o problema, eliminação de *outliers* e transformações de variáveis. Finalmente, usa diversos modelos e verifica qual o modelo com melhor desempenho a prever o *target* em questão. Depois de obtido o melhor modelo, usa o mesmo de forma a fazer *scoring online*. Por isso, a cada momento, são reunidos todos os clientes que estão à espera de atendimento e todos os operadores disponíveis. Ou seja, se há 15 clientes à espera e 10 operadores disponíveis, cria um *dataset* com 150 linhas, as quais são sujeitas ao modelo anteriormente desenvolvido de forma a prever um *score*. Assim, por fim, o *Optimizer*, para cada cliente, verifica qual é o operador que resulta num maior *score*. É, portanto, esse o operador que é escolhido para o atendimento.

A abordagem tomada no trabalho descrito torna-se aceitável, num contexto como o de venda de produtos. Apesar de não ser referido, a quantidade de chamadas a cada momento não é e nem pode ser grande, isto é, todo o modelo que encaminha clientes para operadores é computacionalmente dispendioso. Deste modo, embora a solução encontrada por Ali funcione otimamente em relação ao problema em causa, não parece ser, ainda assim, a melhor maneira de o resolver.

(Mehrbod et al., 2018) abordou questões relacionadas com dados e centrais de chamadas.

Nesse sentido, procurou responder a como é que dados demográficos e históricos podem ser utilizados nos call-centers de forma a melhorar a satisfação dos clientes. Por isso, fez uso de três conjuntos de dados diferentes relacionados com dados biográficos de clientes como idade, género, estado civil e profissão, informações dos operadores como idade, género, qualificação, *score* de avaliação recente feita por supervisor e seis meses de dados históricos com informações acerca de chamadas *inbound*. Além disso, definiu o *target* como o resultado do desfecho da chamada (positivo ou negativo).

O autor começou por aplicar algum pré-processamento de dados, onde removeu dados nulos. De seguida, tratou o desbalanceamento das classes, visto que a classe referente a chamadas positivas é maioritária relativamente a chamadas negativas. Deste modo, usou a técnica de *undersampling*, na qual manteve as observações relativas à classe minoritária e removeu, aleatoriamente, observações relativas à classe maioritária. Usou, também, a técnica de *oversampling*, com recurso a SMOTE, na qual gerou observações relativas à classe minoritária. Usou algoritmos de RNA, Random Forest e regressão logística. Conseguiu resultados satisfatórios já que no Random Forest, obteve uma *accuracy* de 0.92 e um valor de AUC-ROC correspondente a 0.90.

Neste trabalho verifica-se que o problema a tratar é muito semelhante ao que esta tese se propõe a resolver. Também neste são usados dados históricos de clientes e informações relativas a estes e alguns dados estatísticos relacionados com operadores de chamadas. Contudo, torna-se semelhante ao trabalho anteriormente falado, no que diz respeito à solução, já que junta os N operadores disponíveis com os M clientes em espera. O *target* definido, porém, não parece suficiente porque um resultado positivo ou negativo em relação a uma chamada não revela, apesar de tudo, tudo o que seria de bom prever. O sucesso na resposta a uma chamada não significa, de todo, que o atendimento foi o máximo que a empresa podia fornecer, ao cliente, naquele momento.

3. MODELO DE PREVISÃO

Esta secção descreve todos os conjuntos de dados usados para treinar os modelos, bem como todo o fluxo de trabalho que leva aos resultados finais. Assim, descreve-se o módulo de previsão, ilustrado na Figura 2.

3.1 MATERIAIS

O objetivo desta tese, nesta fase, é o desenvolvimento de um modelo de *machine learning* para prever a razão da chamada de um cliente, para o serviço de apoio ao cliente de uma empresa de telecomunicações. De forma a tornar isso possível, é necessário ter o histórico completo de problemas e dos esforços feitos nas suas resoluções. Além disso, são também necessários os dados relacionados com os clientes já que, cada um, atua de forma diferente de acordo com fatores como idade, necessidade do serviço e outras variáveis. Os diagnósticos técnicos e reparações técnicos podem, da mesma forma, serem necessários e devem ser considerados no momento da chegada de uma nova chamada. O conjunto completo de dados resumirá as interações entre cliente e serviço de apoio ao cliente (SAC), todas as análises técnicas e todas as intervenções técnicas e algumas informações sobre o perfil do cliente e sobre o uso que este dá ao conjunto de todos os serviços que tem subscritos. A linguagem *Python* será usada como a principal linguagem de programação a conduzir o trabalho porque possui muitas bibliotecas e *frameworks* que suportam, com facilidade, o desenvolvimento de modelos de *machine learning*.

3.1.1 JupyterLab

Para tornar esta tese possível e de forma a desenvolver o modelo preditivo, foi necessário um IDE para fornecer um local onde o código necessário pudesse ser escrito e testado. Contudo, em vez de se usar um IDE, usou-se o Jupyter Notebook. O Jupyter Notebook é baseado em navegador, open-source, e suporta fluxos de trabalho, código e visualização de dados (Randles et al., 2017).

A ferramenta possibilita texto, em forma de comentário (*markdown*) e código, assim como o seu *output*. Os ficheiros criados, aos quais se dá o nome de notebooks, podem ser exportados nos mais diversos formatos, inclusive até em formatos como PDF.

A ferramenta, possibilita ainda que processos longos sejam divididos em pedaços menores que são executados independentemente. Dá, deste modo, a possibilidade de executar apenas parte do código que não funcionou ou que se quer que funcione, ao invés de perder tempo valioso executando todo o código novamente, como acontece em outras ferramentas e outros ambientes de desenvolvimento de *software*.

Em vez da versão simples do Jupyter Notebook, utilizou-se o Jupyter Lab nesta tese, que corresponde a uma versão melhorada. Oferece todos os blocos de construção e todas as vantagens

familiares e associadas ao clássico Jupyter e oferece algumas vantagens no que diz respeito à interface de utilização (Pandey, 2019).

3.1.2 Pacotes Utilizados

Nesta fase do projeto, para além do ambiente descrito acima, usaram-se múltiplas bibliotecas disponíveis em Python, que são comummente usadas em problemas do género. Portanto, de forma a carregar, a trabalhar e a manipular conjuntos de dados, usou-se a biblioteca pandas que oferece estruturas e operações de dados para manipulação de tabelas. Para trabalhar com as estruturas multidimensionais da forma mais eficiente possível, dado o tamanho dos conjuntos de dados, usou-se a biblioteca numpy que possui uma larga coleção de funções matemáticas para trabalhar com estruturas do género. Para visualizar os dados em interfaces atrativas, informativas e dinâmicas usou-se a biblioteca seaborn. Por fim, de forma a aplicar todos os métodos e abordagens relacionadas com ML, deu-se uso à biblioteca scikit-learn que é uma biblioteca de *machine learning*, e a algumas bibliotecas auxiliares disponíveis para esse efeito.

3.2 DESCRIÇÃO DOS DADOS

No contexto do problema a solucionar, o conjunto de dados base será constituído, essencialmente pelo SAC, ou seja, tudo aquilo que se sabe relacionado com o serviço de atendimento ao cliente e consumidor. Mais tarde, a ele são adicionados dados adicionais provenientes de múltiplas fontes de dados como informações adicionais do cliente e informações do serviço associado.

3.2.1 *Dataset* do SAC

Nos dados do serviço de apoio ao cliente, cada linha, representa uma interação de um operador com o sistema, relativamente a um cliente. Essa ação não é necessariamente uma chamada, mas resulta, normalmente, de uma chamada para o serviço de apoio ao cliente, ou da deslocação de um cliente a uma loja.

Cada ticket tem, a si associado, colunas/variáveis que representam informação relacionada com os mais diversos assuntos, tais como:

- Administrativos: sistemas associados ao ticket;
- Dados gerais do cliente: chave/id, *bundle*/pacote contratado, datas de permanência, reincidências, entre outros;
- Dados do ticket: chave/id, chave/id do serviço a tratar, tipificações do problema a tratar, equipa ou departamento responsável, operador responsável, entre outros.

3.2.2 *Dataset* de Idades Estimadas

Cada linha descreve a idade estimada de um cliente, representado pela sua chave/id. A idade é obtida previamente pelo sistema e estimada por este a partir do número de identificação fiscal fornecido pelos clientes no ato da subscrição dos serviços.

3.2.3 *Dataset* de Clientes

Cada linha descreve as informações úteis que a empresa tem relativamente a cada cliente. É constituído por atributos como região do país onde o cliente habita, valor que o cliente paga mensalmente, sexo, se tem a fatura eletrónica ativa ou não, se cumpre os prazos de pagamento das faturas e, também, em que ciclo de faturação é que está inserido. Esta informação relativa a cada cliente é passível de alteração ao longo do tempo e, por isso, o atributo 'válido desde' permite saber a informação válida, numa dada data.

3.2.4 *Dataset* de Despistes

Este conjunto de dados representa todos os despistes técnicos que foram realizados, remotamente, aos clientes. Portanto, quando surge um cliente com um problema técnico nos seus serviços, tenta-se resolver o problema por chamada telefónica em vez de agendar de imediato uma intervenção pessoal na casa deste, a qual tem um custo adicional para a empresa. Este *dataset* é, portanto, constituído por variáveis como problema e sintoma descrito, alguns passos realizados que tentam ir de encontro à solução (página de saída), o que resulta do despiste, entre outros.

3.2.5 *Dataset* de Instalações

Representa todas as intervenções técnicas pessoais realizadas por técnicos, ou seja, deslocações à casa dos clientes no sentido de efectuar instalações, manutenções, alterações de serviço ou restabelecimentos.

3.2.6 *Dataset* de Equipamentos

Cada linha descreve um equipamento que um cliente teve em determinada altura. É constituído pela identificação do equipamento, pelas datas de início e fim do aluguer.

3.2.7 *Dataset* de Consumos

Cada linha descreve alguma informação relativa à utilização dos diversos serviços, por parte de um cliente, em determinado mês.

3.3 PRÉ-PROCESSAMENTO DOS DADOS

3.3.1 Integração dos Dados

Como se mostrou na secção 3.2, os diferentes dados disponíveis para utilização estão altamente distribuídos, ou seja, estão armazenados e separados em lugares diferentes. Por isso, o primeiro passo dado, no sentido de construir a solução para a problemática, foi o de juntar todos os dados, ou todos os *datasets*, num só *dataset*. Para isso, foi necessário perceber de que forma é que os diferentes *datasets* se podiam unir.

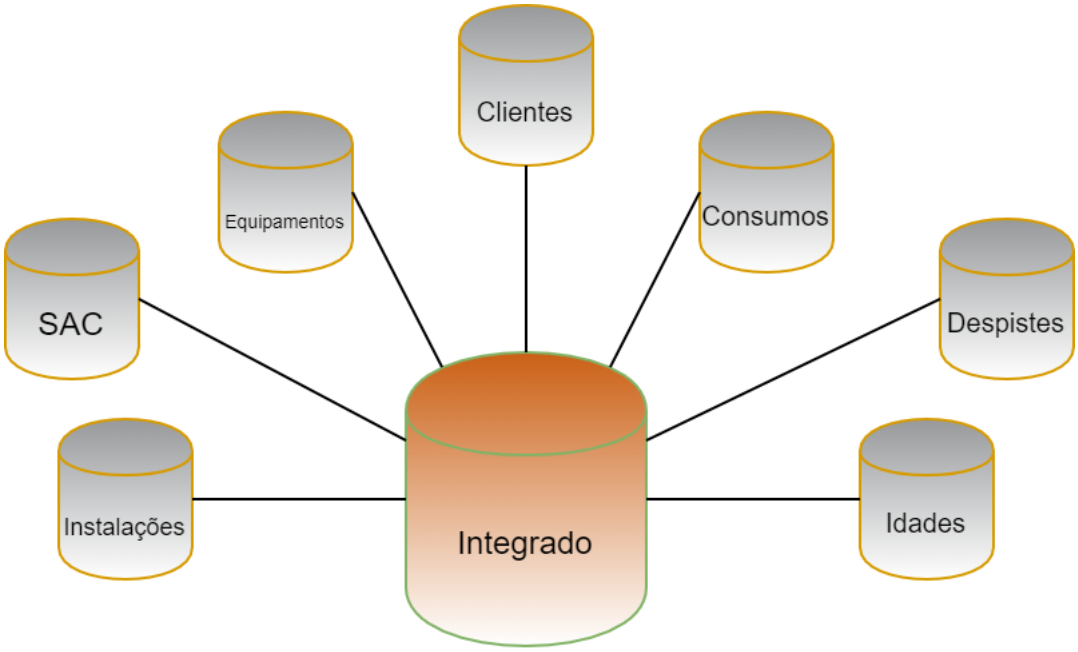


Figura 21: Esquema do processo de integração dos dados.

Deste modo, e tendo como ponto de partida o *dataset* do serviço de apoio ao cliente, começou-se por ligar a este, a informação relativa aos clientes. Isso foi possível graças à chave/ID de cliente, que é o ponto comum entre os dois, e a uma operação de *right join* com *asof*, que permite alcançar a informação válida à data.

Através de uma análise profunda das instalações, verificou-se que os dados aí presentes não eram úteis para o desenvolvimento do modelo pretendido. Contudo, por exemplo, o número de instalações que um cliente já teve, ou o número de alterações de serviço, podem indicar um cliente novo, ou um serviço recente. De igual forma, o número de manutenções a que foi sujeito pode indicar um serviço com tendência para problemas técnicos. Portanto, estes dados foram sujeitos a extração de algumas dessas variáveis, que conseguem resumir toda a informação útil que esses dados podem dar para o problema a resolver. A união com o SAC foi, mais uma vez, a partir da chave/ID do cliente e de uma operação de *right join* com *asof*, que permite obter a informação válida aquando da data do ticket do serviço de apoio ao cliente.

Os despistes, como já foi esclarecido nesta tese, são intervenções virtuais que tentam minimizar o número de intervenções físicas e presenciais, que têm um custo mais elevado. Portanto, cada ticket do serviço de apoio ao cliente, com algum caráter técnico, eventualmente, terá resultado em despiste. De forma a fazer-se a união, usa-se o que os dois têm em comum, ou seja, a chave/ID do ticket.

A junção do SAC com as idades faz-se a partir da chave/ID do cliente e de uma operação de *right join*.

A união do conjunto de dados de consumos e do conjunto de dados relacionados com os equipamentos com o SAC foi, também, a partir da chave/ID do cliente e de uma operação de *right join* com *asof*, que permite obter a informação válida aquando da data do ticket do serviço de apoio ao cliente.

3.3.2 Construção da Variável *Target*

Quando uma chamada é feita para o serviço de apoio ao cliente, é tipificada pelo operador que atendeu e fica registada no sistema. A tipificação, ou seja, a descrição do problema, é feita de forma hierárquica e, por isso, três categorias são responsáveis por descrever a razão da chamada, e, em cada uma delas, o problema vai sendo cada vez mais aprofundado.

A construção da coluna *target*, ou seja, aquilo que se quer prever, é conseguida à custa de um mapeamento feito a partir das duas primeiras colunas hierarquizadas anteriormente descritas. Devido a se querer prever a razão para uma chamada por parte do cliente, o problema é multiclasse, em vez de binário, que é mais comum em soluções típicas de ML.

Por conseguinte, quando a primeira coluna descreve uma falha de serviço, uma análise à segunda é feita no sentido de perceber de que tipo foi a falha. O mapeamento para as classes **Problema - TV**, **Problema - TLM**, **Problema - NET** e **Problema - TLF** é, então, feito analisando a presença (ou não) de palavras relacionadas, como mostra na Tabela 2.

Tabela 2: Mapeamento realizado para atribuição de classes.

Classe	Palavras Relacionadas
Problema - TV	TV, BOX, VIDEOCLUBE, GRAVAÇÃO
Problema - TLM	VOZ MÓVEL, MENSAGENS, ROAMING
Problema - NET	INTERNET
Problema - TLF	VOZ FIXA

Nos restantes problemas, a informação está duplicada entre as duas colunas e, por isso, o mapeamento faz-se acedendo apenas à primeira coluna. Isso permite distinguir as restantes razões mais comuns para os clientes ligarem e algumas classes são criadas, tais como: **Desativações** que correspondem à vontade de o cliente querer desistir dos serviços da empresa, **Faturação** quando há problemas ou dúvidas relacionados com faturas, **Pagamentos** quando há dúvidas

em relação ao ato de pagamento, **Serviços** quando o cliente quer aderir ou alterar um serviço, **Configurações - Apoio Téc** se o cliente procura ajuda com o manuseio dos serviços, **Apoio Cliente** relativamente a assuntos relacionados com transferências de morada, documentação, portabilidades e alterações de titularidade, **Equipamentos** quando há dúvidas relativas a boxes e a equipamentos, **Informações** relativas a dúvidas gerais e **Intervenções** quando é para discutir sobre manutenções ou instalações técnicas presenciais. Os restantes problemas, mais incomuns e, portanto, mais difíceis de prever, são agregados na classe **Outros**.

Mais tarde, a variável foi sujeita a uma técnica de codificação (*Ordinal Encoding*), de forma a transformar os problemas, representados por palavras, em números, como mostra a Tabela 3. Também, as classes relativas a informações, configurações, apoio a cliente, por serem menos previsíveis e menos frequentes, foram agregadas na classe **Outros**.

Tabela 3: Identificador atribuído a cada classe.

Classe	Identificador
Desativações	0
Equipamentos	1
Faturação	2
Outros	3
Pagamentos	4
Problema de Internet	5
Problema de Telefone	6
Problema de Telemóvel	7
Problema de Televisão	8
Serviços	9

3.3.3 Limpeza dos Dados

Num contexto empresarial e numa era digital como a presente, é de esperar que os dados não sejam perfeitos, como é habitual acontecer em ambientes académicos. Portanto, a limpeza dos dados revelou ser, em especial nesta dissertação, uma etapa essencial com busca a melhores desempenhos, principalmente no que diz respeito aos dados do serviço de apoio ao cliente. Inicialmente, através de uma análise exploratória e exaustiva dos dados foi, até, desde logo possível perceber que alguns dados deveriam ser removidos.

Em primeiro lugar, removeram-se todos os assuntos que se encontravam pendentes e abertos. Também se apagaram as linhas relativas a clientes empresariais, visto que o objetivo é melhorar o serviço de apoio a particulares, pois constituem a grande fatia. Observaram-se, também, linhas que não estavam associadas a nenhum cliente e a nenhum serviço. Ou seja, com certeza estão relacionadas com pessoas que ligam para o serviço de apoio ao cliente, mas que nunca são identificadas, ora porque não o querem ser, ora porque não são, efetivamente, clientes da empresa.

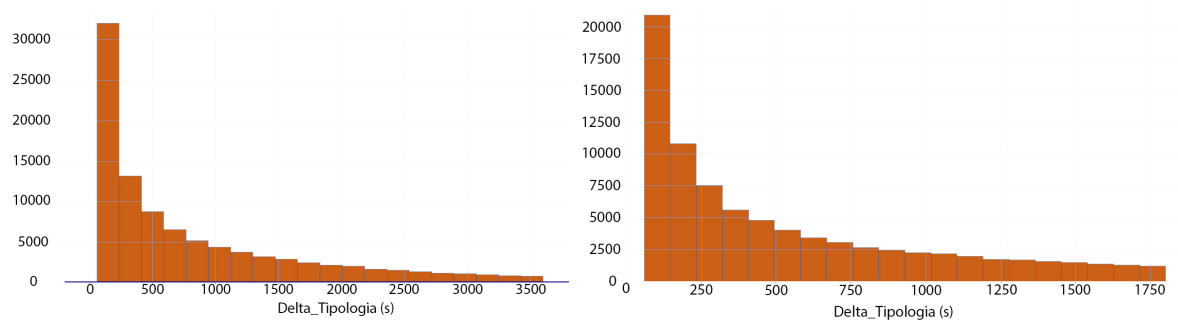
De qualquer das formas, a chamada e o acontecimento ficam registados. Estas linhas devem ser removidas já que não é possível estabelecer um histórico nem é possível adquirir informações sobre a pessoa que ligou.

Tabela 4: *Target* e sua distribuição.

Classe	**Antes de Limpeza**	**Depois de Limpeza**
0 - Desativações	1100659 (19%)	160523 (7%)
1 - Equipamentos	707821 (12%)	60506 (3%)
2 - Faturação	290786 (5%)	163695 (7%)
3 - Outros	1263669 (21%)	463111 (19%)
4 - Pagamentos	188551 (3%)	144350 (6%)
5 - Internet	583164 (10%)	409821 (17%)
6 - Telefone	115551 (2%)	82726 (3%)
7 - Telemóvel	45168 (1%)	33157 (1%)
8 - Televisão	799240 (14%)	630134 (26%)
9 - Serviços	789823 (13%)	254384 (11%)
TOTAL	5884432 (100%)	2402407 (100%)

Verificou-se que a empresa está a migrar de um sistema de informação antigo para um novo, e nesse, os dados estão estruturados de forma distinta. A maioria dos dados, ainda assim, residem, na sua maior parte, no sistema antigo. Tomou-se a decisão de remover linhas referentes ao sistema novo, de forma a minimizar a perda de informação e de histórico de cliente. Houve a necessidade de escolher entre um dos sistemas, dado que as diferenças entre estes não permitem a concretização de uma solução compatível com os dois.

Alguns dados são tipificados como duplicados e com erro. Além disso, alguma informação relativa à equipa/departamento que lidou com o problema está indisponível e, torna-se impossível rastrear e perceber um problema com esse desconhecimento. A decisão tomada foi a de remover linhas com esse tipo de característica. Também o sistema, por vezes, gera informação duplicada, seja por questões de distribuição de dados ou por outra razão desconhecida. O próprio operador pode, por engano, registar a chamada (e o problema) mais do que uma vez. Portanto, de forma a contornar isso, apagaram-se todas as linhas posteriores a uma chamada, relativas ao mesmo cliente, num intervalo de tempo inferior a 5 segundos. Isto é, um problema é tipificado e registado no sistema e, se mais algum problema surgir nos próximos 5 segundos, é apagado porque supõe-se que o operador não tem tempo suficiente para o fazer e, além disso, nenhum cliente consegue reportar num intervalo tão pequeno de tempo. Inclusive, por análise da Figura 22, conclui-se que a reincidência de chamadas para problemas do mesmo tipo é mais ou menos quadrática até à primeira meia hora. A partir daí, fica linear, como indica a Figura 23. Portanto, reincidências para o mesmo problema num intervalo inferior a 30 minutos foram também apagadas, e são tratadas como se fossem duplicadas.



(a) 1 minuto a 1 hora

(b) 1 minuto a 30 minutos

Figura 22: Volume de reincidência de chamadas para o mesmo problema em curtos intervalos de tempo.

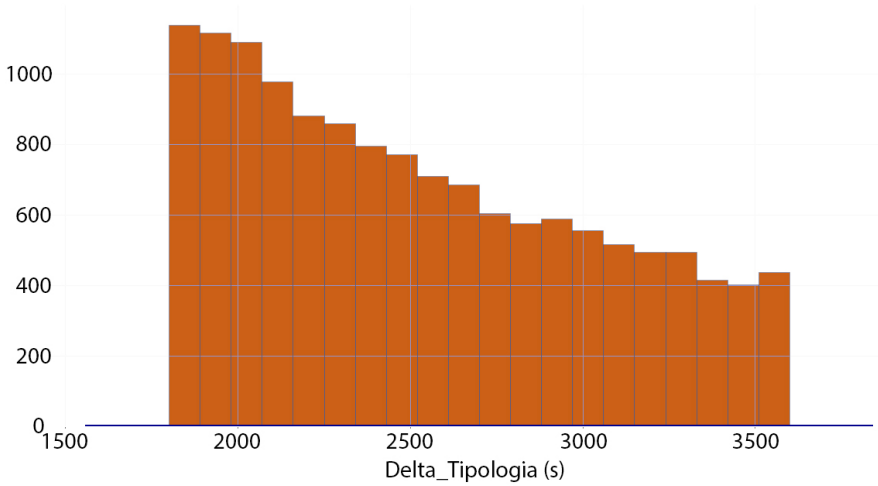


Figura 23: Volume de reincidência de chamadas para o mesmo problema entre a primeira meia hora e a primeira hora.

Os dados relativos ao serviço de apoio ao cliente contêm algumas colunas administrativas. Essas informações foram, também, retiradas. Além disso, verificou-se que, muitas das vezes, a maioria dessas colunas têm uma cardinalidade muito próxima de um, ou seja, não acrescentam informação relevante. Retiraram-se identificadores de assuntos ou de tickets, variáveis de tempo diretamente correlacionadas com as datas completas, variáveis relacionadas com campos adicionais que nunca são preenchidos e variáveis relacionadas com os sistemas internos.

Quando um problema não fica resolvido no contacto, ou quando é relatado pelo cliente numa loja física, de forma a solucioná-lo, a empresa liga ao cliente, ou seja, o contacto deixa de ser *inbound call*, mas sim *outbound call*. Essas chamadas que têm origem nos call-centers da empresa ficam registadas no sistema. Como tal, essas linhas, bem como as que têm origem nas lojas foram removidas, uma vez que a aprendizagem deve ser feita com chamadas *inbound*.

3.3.4 Visualização dos Dados

Alguma visualização de dados foi feita no sentido de estudar e prever a importância de algumas variáveis e seus comportamentos, face à variável a prever (*target*).

A Figura 24 demonstra a relação entre a variável idade e a variável *target*. Desta forma, permite verificar se a variação nas idades origina diferentes problemas. Analisando a figura, conclui-se, por exemplo, que os problemas de telefone estão mais ligados a pessoas com mais idade e, problemas com pagamentos, são mais comuns em clientes mais jovens. Os problemas de televisão também vão aumentando com a idade.

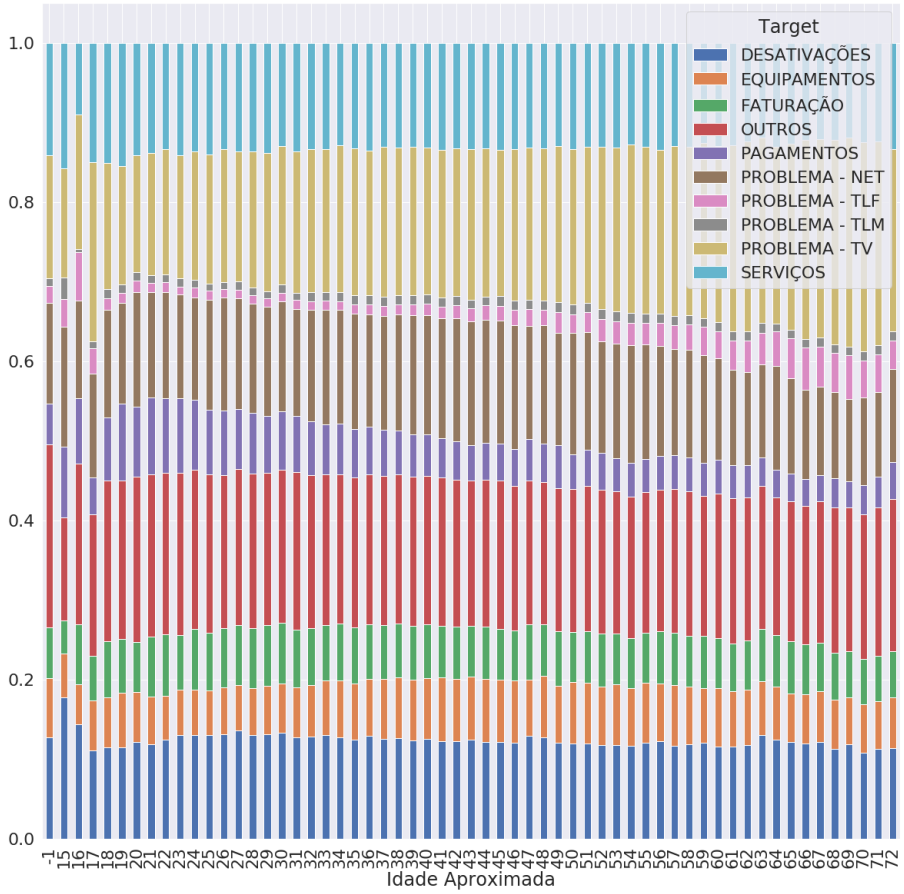


Figura 24: Gráfico de problemas em função da idade dos clientes.

A Figura 25 ilustra a relação entre a variável mensalidade (preço que o cliente paga pelo serviço) e a variável que se pretende prever, *target*. Permite concluir se o tipo de problema varia com a mensalidade. Analisando, verifica-se, por exemplo, que as questões relacionadas com serviços surgem mais em preços mais elevados e os problemas de televisão são mais comuns em mensalidades mais baixas.

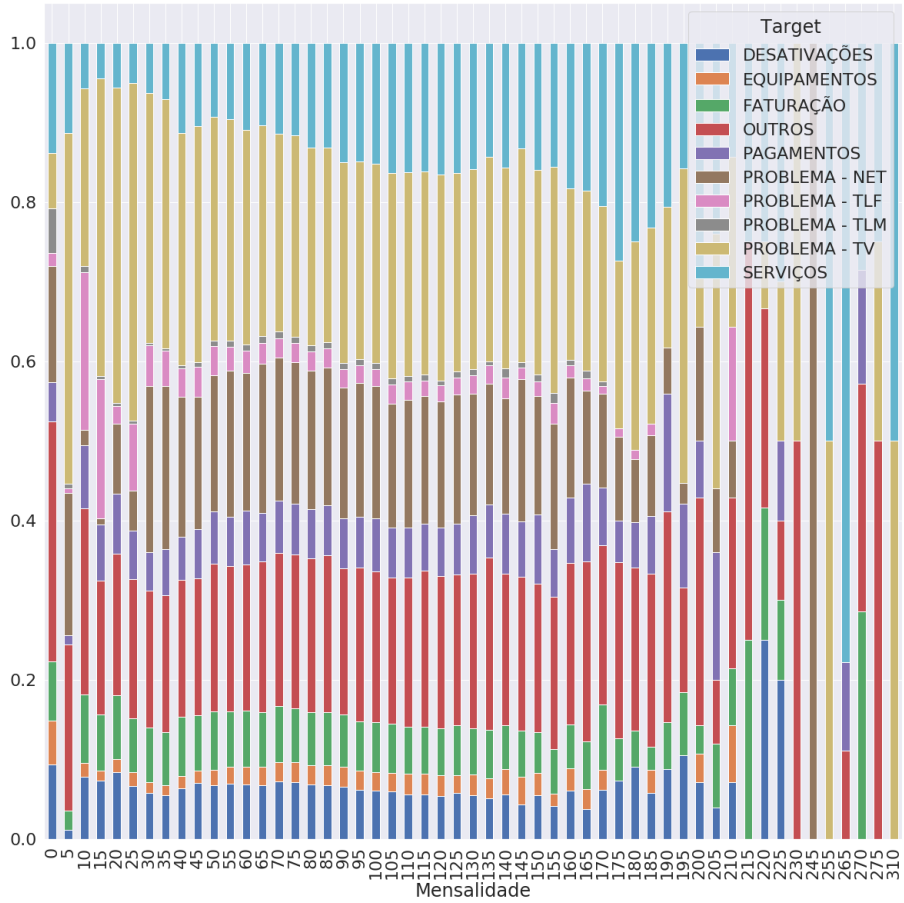


Figura 25: Gráfico de problemas em função da mensalidade paga pelos clientes.

3.3.5 Seleção de Variáveis

Nesta tese, e uma vez que o uso de modelos de árvores de decisão foi generalizado, a seleção de variáveis *a priori* deixou de fazer sentido. Os modelos *ensemble* como o Random Forest, o LightGBM e o XGBoost têm um hiperparâmetro que permite escolher a fração de variáveis mais importantes a usar, ou seja, faz-se, por isso, uma seleção de variáveis de forma *embedded*.

3.3.6 Extração de Variáveis

Esta etapa revelou ser bastante importante porque permitiu extrair muita informação escondida nos dados. Nesse sentido, como já foi dito antes, começou-se por obter o número de manutenções e o número de instalações de cada cliente, até ao momento da chamada. Isso é conseguido contando as intervenções técnicas que a pessoa teve até esse momento. Também, a partir da data de fim do contrato e da data de criação da conta do cliente, foram determinados os números de meses restantes até ao fim do contrato e há quanto tempo é que o cliente está ligado à empresa.

Como foi dito na secção 3.3.3, algumas linhas foram removidas e suprimidas por diversas razões, tais como idas a lojas ou chamadas *outbound*. Contudo, algumas dessas informações foram

retidas em variáveis como o número de vezes que o cliente foi à loja até aquele momento, o número de pedidos de desligamento, o número de vezes que foi contactado pela retenção da empresa (equipa responsável por evitar a desistência dos serviços por parte do cliente), número de vezes em que foi contactado pela provedoria devido a uma reclamação, entre outros.

O concelho de residência do cliente pode, em determinados momentos, sofrer de problemas técnicos generalizados ou alguns picos de utilização. Por isso, construíram-se variáveis que permitem compreender a incidência de problemas no concelho nos últimos 15 dias. Ou seja, se na zona de residência do cliente estiverem a acontecer imensos problemas de internet, é provável que a queixa do cliente vá também nesse sentido. Essas e mais algumas variáveis que foram extraídas são descritas e sumariadas em seguida:

- **#Manutenções** - número de manutenções presenciais;
- **#Instalações** - número de instalações presenciais;
- **#OTAlteraServico** - número de alterações de serviço presenciais;
- **#OTRestabelecimento** - número de restabelecimentos presenciais;
- **Contrato Restante SA** - meses restantes para o fim do contrato do serviço;
- **Contrato Restante CA** - meses restantes para o fim do contrato do cliente;
- **Idade Ativação (d)** - há quanto tempo a conta do cliente foi ativada;
- **Idade da Conta (d)** - há quanto tempo o cliente está na empresa;
- **Duração Ticket (min)** - média de duração dos despistes técnicos que o cliente teve, até à data da chamada;
- **#Loja** - número de vezes que o cliente já se deslocou à loja;
- **#Retenção** - número de vezes que o cliente quis desistir dos serviços da empresa;
- **#Despistes** - número de despistes técnicos feitos ao cliente;
- **#Reclamações** - número de reclamações/provedorias;
- **Tempo desde Último Contacto** - há quanto tempo o cliente não tem um problema;

Sendo Problema $\epsilon [Problema{-}NET, Problema{-}TV, Problema{-}TLM, Problema{-}TLF, Pagamentos, Faturação, Desativações, Equipamentos, Serviços]$, construíram-se 50 variáveis (5 x 10 classes):

- **Tempo desde Problema** - há quanto tempo o cliente não tem um problema do mesmo género;
- **%Problema Concelho vs País 15D** - percentagem de incidência do problema no concelho, em relação ao resto do país, nos últimos 15 dias;
- **%Problema Concelho 15D** - percentagem de incidência do problema no concelho, nos últimos 15 dias;
- **#Problema 30D** - número de vezes que o cliente teve um problema do mesmo género, nos últimos 30 dias;
- **#Problema** - número de vezes que o cliente já teve um problema do mesmo tipo.

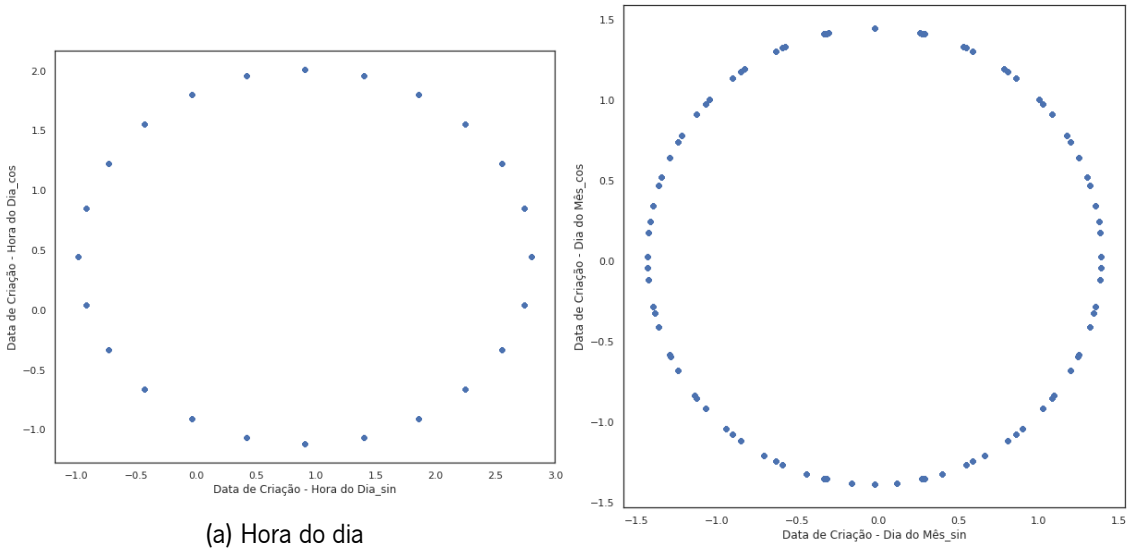
Todas as variáveis anteriormente descritas são calculadas com informação anterior à data da chamada.

De forma a adaptar as datas de início de chamada aos modelos, estas foram convertidas em múltiplas variáveis cíclicas (ver Figura 26), recorrendo ao círculo trigonométrico e a funções matemáticas de seno e cosseno. Desta forma, os modelos conseguem perceber o carácter cíclico da informação temporal visto que, por exemplo, as 23h da noite estão próximas das 2h da manhã. Por isso, a cada um dos pontos abaixo referido, correspondem duas variáveis relativas a seno e cosseno.

- **Hora do dia** - hora do dia em que a chamada foi feita (0-24h);
- **Época** - época do ano em que a chamada foi feita (divisão do ano em quatro);
- **Semana** - semana do ano em que a chamada foi realizada;
- **Dia da Semana** - dia da semana em que a chamada foi feita (ex: domingo);
- **Dia do Mês** - dia do mês em que a chamada foi feita (um mês pode ter 28, 29, 30 ou 31 dias);
- **Semana do Mês** - em que semana do mês é que a chamada foi feita (um mês pode ter entre 4 a 6 semanas);
- **Feriado** - se a chamada foi feita ou não num feriado nacional.

Por último, algumas variáveis de atraso (*lag features*) foram criadas, no sentido de compreender o contexto das últimas chamadas. São descritas e sumariadas em seguida:

- **Tipificação II, III e IV, dos últimos 5 contactos/chamadas**, para se perceber mais aprofundadamente o tipo de problemas que o cliente reporta;



(a) Hora do dia

(b) Dia do mês

Figura 26: Resultado obtido da transformação das variáveis temporais em temporais cíclicas.

- **Problema/Classe** relatada na **última chamada**;
- **Reincidência**, ou seja, se a **última chamada** era uma reincidência de um problema prévio. Poderá indicar uma insatisfação acumulada;
- **Resolvido no contacto**, isto é, se o descrito no **último contacto** ficou ou não resolvido. Se não ficou resolvido, é provável que o cliente ligue para resolver o mesmo problema;
- **Houve despiste**, se houve despiste técnico ou não na **última chamada**. Se não houve, o problema poderá não ter ficado resolvido;
- **Equipas** de criação de ticket e fecho de ticket do **último ticket/contacto**;
- **Percentagem de reincidência** do **operador anterior**, em problemas como aquele que foi relatado;
- **Percentagem de resolvido no contacto** do **operador anterior**;
- **Percentagem de resolvido** em relação à **página de saída**, ou seja, no que diz respeito ao que foi feito no despiste técnico, no caso de este ter acontecido, na **última chamada**;
- **Média de contactos para o problema anteriormente** relatado.

3.3.7 Transformação dos Dados

Alguma transformação de dados, ao nível do tratamento dos *outliers* foi necessária. Na verdade, em muitos casos, os dados discrepantes têm bastante significado e, de facto, nesta tese quer-se que os modelos estejam cientes da sua existência e da sua probabilidade de existirem e de

ocorrerem. Todavia, verifica-se que existem dados que são mesmo resultantes de erros de *input* ou de anomalias. Achou-se importante remover por serem erros claros.

Quanto aos restantes dados que, embora possam ser *outliers*, não são erros claros de *input*, mais à frente, tenta-se avaliar o impacto que estes poderão ter no desempenho dos modelos. Por isso, avaliam-se cenários em que são completamente descartados, com o uso por exemplo do *RobustScaler*.

A variável que indica a idade da conta de um cliente, e tendo em atenção que a empresa não tem mais do que 25 anos de existência, nunca pode assumir um valor superior a isso. Portanto, dado que a variável está numa escala diária, definiu-se que todos os valores acima de 10000 dias (27.4 anos) seriam *outliers*. Portanto, valores superiores a esse limite foram alterados para 10000 dias, que passou a ser o teto máximo.

Também em relação às variáveis de contrato restante, tanto do cliente, como do próprio serviço, analisou-se a existência de valores que superavam o teto máximo de 24 meses. Efetivamente, não se fazem contratos superiores a dois anos e, se algum valor for superior a isso, é um *outlier* e um erro. Portanto, à semelhança das idades das contas, valores superiores a dois anos foram alterados para 24 meses, que passou a ser o limite máximo para meses de contrato restantes.

A respeito da mediana da duração do total das chamadas dos últimos 7 dias e dos últimos 30 dias, verificaram-se também algumas durações fora do comum, como chamadas com durações superiores a 4 horas. Nesse sentido, todas as durações superiores ao percentil 95, foram reduzidas para este.

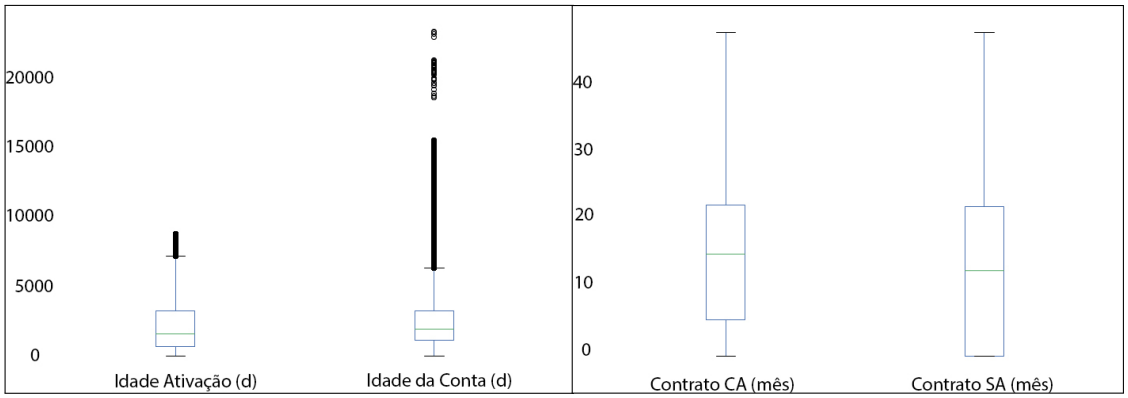
Finalmente, fez-se uma última verificação em todas as variáveis, no sentido de perceber se ainda poderiam haver nulos e infinitos, os quais não são suportados pela maioria dos modelos de ML. Assim, estes foram substituídos em alguns casos pelo valor -1 e noutros casos pelo valor 0.

Clustering

Algumas variáveis estão relacionadas com informações escolhidas pelo operador de chamada de maneira semiestruturada. Portanto, foi necessário usar algoritmos de incorporação de palavras para extrair o contexto da chamada e o problema relatado. O FastText (Bojanowski et al., 2016), uma extensão criada pelo Facebook a partir do famoso Word2Vec (Mikolov et al., 2013), foi escolhida. Transforma palavras e, consecutivamente, frases em vetores, que posteriormente são o *input* num modelo de agrupamento, o K-Means. O uso desse modelo não supervisionado dividiu as frases (vetores) em diferentes grupos.

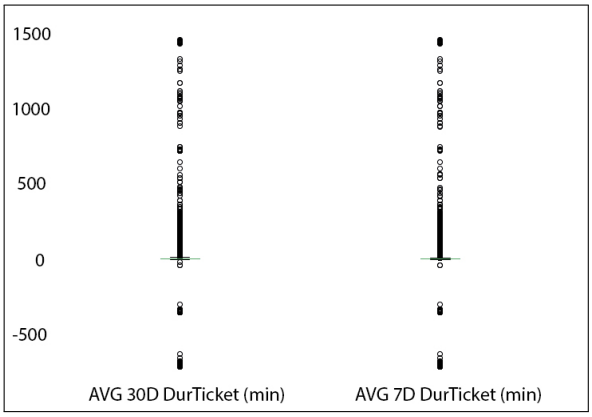
As variáveis têm, cada uma, mais de 500 valores únicos. Contudo, verificou-se que, pouco mais de uma centena deles, são usados 95% das vezes pelos operadores. Portanto, as tipificações não comuns foram substituídas pelo texto 'Tipificação - Outros'.

O uso do algoritmo pressupôs a utilização de técnicas de tratamento de texto como remoção de



(a) Idade de Conta

(b) Contrato



(c) Duração de Ticket

Figura 27: *Outliers* das variáveis em análise.

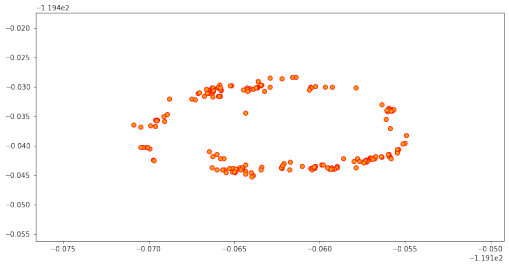
caracteres especiais, uso de stemização (reduzir as palavras ao seu radical) e remoção de palavras vazias, ou seja, palavras que não adicionam informação relevante, tais como advérbios, conjunções, pronomes, entre outros.

De forma a facilitar a visualização do *output* do modelo de FastText, foram executados os algoritmos PCA (*Principal Component Analysis*) e TSNE (*t-Distributed Stochastic Neighbor Embedding*) de redução de dimensionalidade. Deste modo, foi possível mostrar os *clusters* em formatos 2D e 3D.

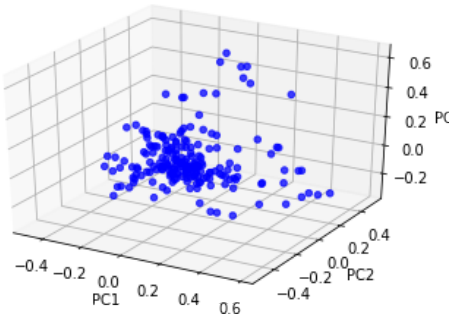
A primeira variável a sofrer esta agregação é onde os operadores tipificam o segundo nível do problema. Ou seja, a partir do que lhes é dito pelo cliente, detalham o problema a uma profundidade definida como nível 2. Foram usados os 150 valores mais comuns.

A segunda variável a ser submetida ao algoritmo de incorporação de palavras foi a que está responsável de armazenar a tipificação nível 3 do problema, onde o problema é descrito ao mais baixo nível, e com alguma exatidão. Foram usados os 150 valores mais comuns.

A terceira variável a ser submetida ao algoritmo de incorporação de palavras foi a que está responsável de armazenar a resolução do operador, ou seja, o que resultou da chamada e do contacto entre operador e cliente. Foram usados os 120 valores mais comuns.

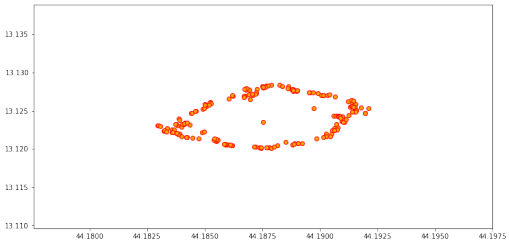


(a) TSNE

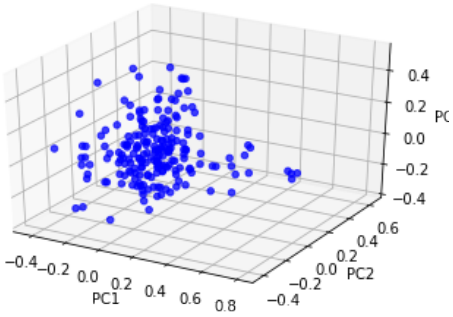


(b) PCA

Figura 28: Representação gráfica dos *clusters* obtidos para a variável tipificação nível 2.

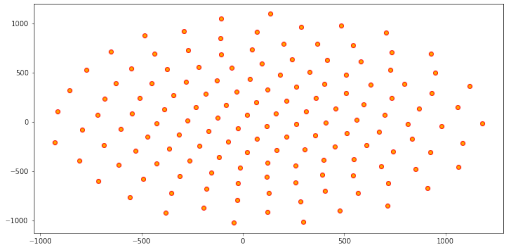


(a) TSNE

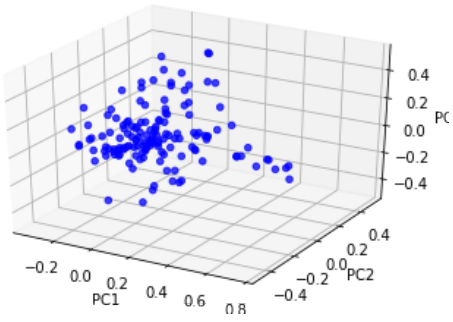


(b) PCA

Figura 29: Representação gráfica dos *clusters* obtidos para a variável tipificação nível 3.



(a) TSNE



(b) PCA

Figura 30: Representação gráfica dos *clusters* obtidos para a variável tipificação nível 4.

A última etapa deste processo foi atribuir um número de *clusters* a cada variável. A decisão foi suportada pela análise gráfica (ver Figuras 28, 29, 30) e pelo método do cotovelo, que pode ser visualizado na Figura 31. Para a primeira variável o número ótimo escolhido foi 22 *clusters*, para a segunda variável foi 19 *clusters* e para a última variável escolheram-se 25 *clusters*.

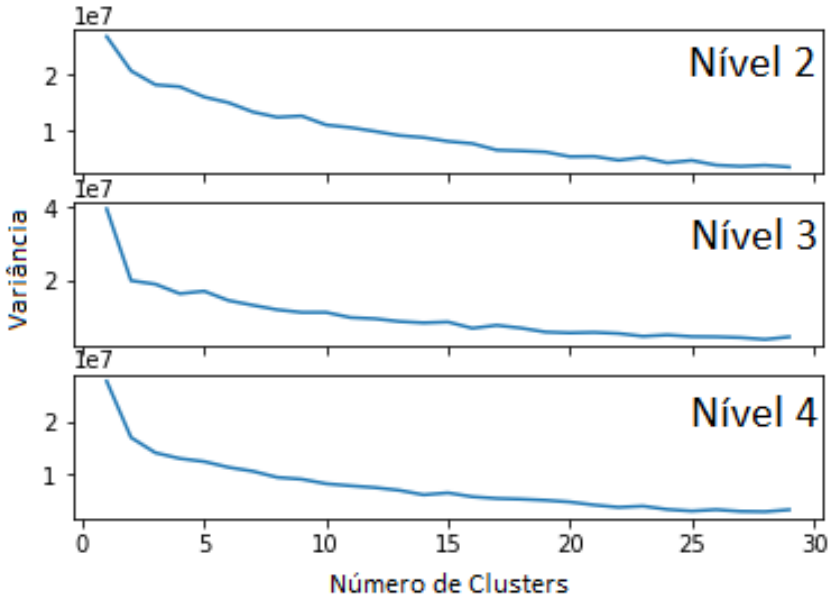


Figura 31: Método do cotovelo aplicado às 3 variáveis.

Encoding

O *encoding* revelou ser uma etapa importante neste processo devido à imensa informação que existe em formato textual. Os modelos não são capazes de lidar com informação nesse formato e há a necessidade de a converter num formato numérico. A estratégia usada foi o *target encoding* com adição de ruído, uma técnica bastante usada que permite evitar o *overfitting* que costuma ocorrer na técnica original.

Inicialmente, a técnica de *one-hot encoding* foi explorada e implementada e, por isso, cada diferente valor de uma variável originava uma coluna diferente. Contudo, devido à já grande dimensionalidade do conjunto de dados, este *encoding* foi deixado de parte de forma a não aumentar ainda mais o tamanho do *dataset* e, consecutivamente, o tempo de treino dos modelos.

3.4 MODELAÇÃO

Esta secção explora o modo como os diferentes modelos de ML são usados, depois de concluídas as etapas de processamento e transformação de dados.

Primeiramente, o problema foi definido como classificação. Decidiu-se avaliar o uso do *StandardScaler*, mas também do *RobustScaler*, de forma a ter um especial cuidado com a presença de possíveis *outliers*. Também, devido ao desbalanceamento das classes verificado na Tabela 4, o balanceamento de classes foi considerado. O *dataset* em uso tem uma grande dimensionalidade e, por isso, tornou-se impossível executar algoritmos de *oversampling* sintéticos como o SMOTE, devido à quantidade de tempo que isso levaria. Contudo, o *random over-sampling* (ROS) foi apli-

cado às duas maiores classes (Problemas de TV e Outros) e o *random under-sampling* (RUS) foi aplicado às restantes classes.

Também uma rede neuronal foi desenvolvida. A arquitetura está representada na Figura 32 e é constituída por uma camada de input (196 colunas), 8 camadas escondidas e 1 camada de output (10 classes).

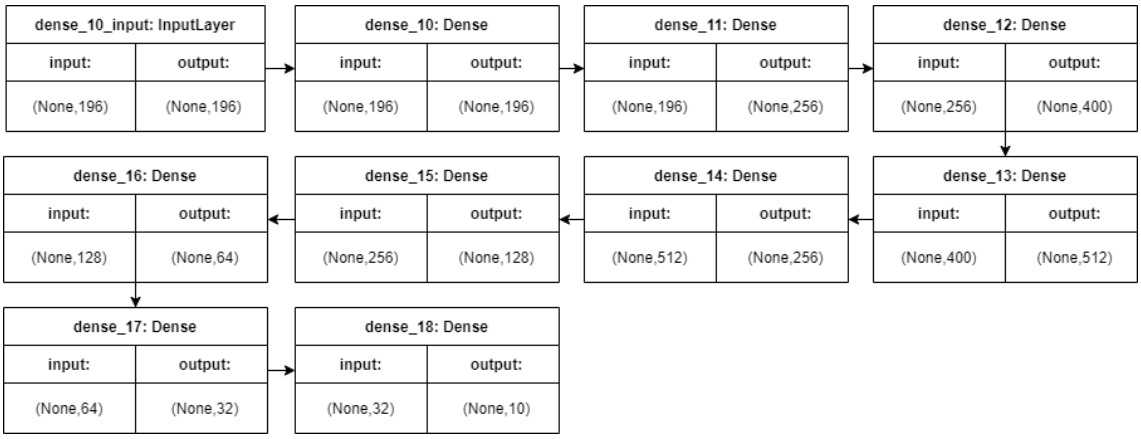


Figura 32: Arquitetura da rede neuronal criada (MLP).

A Tabela 5 resume as combinações que foram testadas, no sentido de encontrar o melhor modelo para os dados em questão.

Tabela 5: Combinação de alternativas para a avaliação.

Scalers	**Balanceamentos**	**Estimadores**
StandardScaler *RobustScaler*	ROS (10%) + RUS (30%) ROS (10%) + RUS (50%) ROS (20%) + RUS (30%) ROS (20%) + RUS (50%) Balanceamento do peso das classes Nenhum	Regressão Logística Naive Bayes Random Forest Redes Neuronais AdaBoost XGBoost LightGBM

O conjunto de dados final foi dividido em conjuntos de treino (70%), para treino do modelo, validação (15%), para seleção e otimização do melhor modelo, e teste (15%) para geração dos resultados finais evitando *bias*. Foi essencial, também, ter cuidado com a forma como a divisão era feita, de forma a que não provocasse *time data leakage*, ou seja, não treinar os modelos com dados que são futuros aqueles que estão alocados a validação ou teste. O conjunto de todas as várias alternativas apresentadas acima, levou à execução de 46 modelos diferentes. A técnica de validação cruzada, bastante usada em problemas de ML foi deixada de parte, face à grande quantidade de dados disponível, que permite recorrer a *holdout* sets, ou seja, dados que são deixados de parte, de forma a medir a performance dos modelos posteriormente.

Tabela 6: Melhores modelos obtidos.

Estimador	***Scaler***	**Balanceamento**	**micro-F1**
Random Forest	*RobustScaler*	ROS (10%) + RUS (30%)	0.372
LightGBM	*StandardScaler*	ROS (10%) + RUS (30%)	0.374
Random Forest	*RobustScaler*	Nenhum	0.382
Random Forest	*StandardScaler*	Nenhum	0.383
XGBoost	*RobustScaler*	Nenhum	0.384
Redes Neuronais	*RobustScaler*	Nenhum	0.385
Redes Neuronais	*StandardScaler*	Nenhum	0.385
XGBoost	*StandardScaler*	Nenhum	0.385
LightGBM	*RobustScaler*	Nenhum	0.398
LightGBM	***StandardScaler***	Nenhum	**0.398**

A escolha de uma métrica de avaliação adequada para modelos gerados a partir de dados desequilibrados é uma tarefa crítica. A *accuracy* não é uma métrica adequada quando se trabalha com um conjunto de dados desequilibrado, como já se viu anteriormente. Portanto, utilizou-se o micro-F1 porque é adequado para classificações desequilibradas.

A Tabela 6 apresenta os resultados que tiveram os melhores desempenhos para todas as combinações que foram avaliadas, sem otimização dos hiperparâmetros, relativamente ao conjunto de validação. Portanto, combinando diferentes escaladores com diferentes tipos de técnicas e modelos de balanceamento (ver Tabela 5) verifica-se o desempenho no conjunto de validação.

A partir dos resultados, observa-se que o melhor modelo foi o LightGBM. Por conseguinte, foi o selecionado para a otimização de hiperparâmetros a fim de o adaptar ainda mais aos dados em questão.

Alguns resultados adquiridos (ver Tabela 7) permitem, também, concluir alguns pressupostos que poderão ser úteis em futuros desenvolvimentos nesta área. Fixar um estimador/classificador permite analisar o impacto das diferentes técnicas de modelação. Portanto, abordagens semelhantes a balanceamento não sintético de classes não originam bons resultados, bem como, preocupação desmedida com os *outliers*. Aconselha-se, no entanto, o uso de modelos com *boosting*, como o LightGBM ou o XGBoost.

Tabela 7: Impacto das diferentes técnicas.

Estimador	***Scaler***	**Balanceamento**	**micro-F1**
LightGBM	*RobustScaler*	ROS (20%) + RUS (50%)	0.334
LightGBM	*StandardScaler*	ROS (20%) + RUS (50%)	0.335
LightGBM	*RobustScaler*	ROS (10%) + RUS (50%)	0.341
LightGBM	*StandardScaler*	ROS (10%) + RUS (50%)	0.342
LightGBM	*RobustScaler*	ROS (20%) + RUS (30%)	0.358
LightGBM	*StandardScaler*	ROS (20%) + RUS (30%)	0.359
LightGBM	*RobustScaler*	ROS (10%) + RUS (30%)	0.370
LightGBM	*StandardScaler*	ROS (10%) + RUS (30%)	0.371
LightGBM	*RobustScaler*	Nenhum	0.397
LightGBM	***StandardScaler***	Nenhum	**0.398**

3.5 OTIMIZAÇÃO DOS HIPERPARÂMETROS

A hiperparametrização do melhor modelo (LightGBM com *StandardScaler* e sem balanceamento) foi conseguida através da técnica de Otimização *Bayesiana* (Nogueira, 14), em detrimento de outros algoritmos, devido à enorme quantidade de dados e ao longo tempo de treino. A seguir, ilustra-se o intervalo de valores em que os hiperparâmetros foram variados. A métrica escolhida para a otimização foi a maximização da métrica micro-F1.

Tabela 8: Intervalo de valores dos hiperparâmetros.

Hiperparâmetro	**Intervalo definido**
num_leaves	[30, 500]
feature_fraction	[0.5, 1]
max_depth	[10, 400]
lambda_l1	[0, 5]
lambda_l2	[0, 3]
min_split_gain	[0.001, 0.1]
min_child_weight	[1, 50]
bagging_freq	[1, 10]
learning_rate	[0.0001, 0.1]

No LightGBM, usar um valor alto para o hiperparâmetro num_leaves é o recomendado para quando se quer uma *accuracy* máxima, visto que possibilita o controlo da complexidade das árvores e, por isso, do modelo. No entanto, definiu-se um valor máximo de 500 porque valores altos induzem, ao mesmo tempo, possibilidade de *overfitting*. O mesmo raciocínio foi feito para o max_depth. Relativamente à fração de variáveis a usar, estabeleceu-se, apenas, o limite mínimo de 0.5 (50%) porque se acreditou que o modelo não conseguisse ter um bom desempenho com menos de metade das variáveis. O min_split_gain, o lambda_l1 e o lambda_l2 foram usados com

vista a evitar/reduzir o overfit e, nesse sentido, exploraram-se intervalos comummente usados por artigos académicos e científicos da mesma área. O bagging_freq foi utilizado de forma a aumentar a velocidade de treino do modelo. Por fim, relativamente ao *learning_rate*, a preocupação foi a de criar um intervalo grande o suficiente para que o modelo fosse testado com um vasto conjunto de diferentes valores.

O desempenho obtido foi ligeiramente melhorado, tendo sido atingido o micro-F1 de 0.41 ao fim de 40 iterações. Os parâmetros ótimos definidos pelo modelo de otimização foram os seguintes:

```
num_leaves = 100,
feature_fraction = 0.88,
max_depth = 46,
lambda_l1 = 5,
lambda_l2 = 2,
min_split_gain = 0.01
min_child_weight = 40,
bagging_freq = 2,
learning_rate = 0.08
```

3.6 RESULTADOS

O modelo otimizado LightGBM foi finalmente testado com o conjunto de teste, tendo sido obtido o mesmo resultado verificado no conjunto de validação, ou seja, micro-F1 igual a 0.41.

As 10 variáveis mais importantes (ver Figura 33) basearam-se principalmente em dados relacionados com o cliente, tais como o tempo desde o início do contrato, a idade, a localização e o tempo desde o último contacto. Estas variáveis podem servir de orientação para que outras empresas implementem também este sistema.

Para complementar este resultado, a curva ROC para o modelo otimizado LightGBM é apresentada na Figura 35, onde é possível verificar o desempenho do modelo na previsão de cada classe, conforme apresentado na Tabela 4, verificando-se que está bem acima da diagonal. Pela análise das curvas, e olhando especialmente para a curva relativa à classe 'Outros', verifica-se que é a que o modelo prevê com menos eficácia. Aliás, também olhando para os resultados gerados pelo relatório, isso se verifica. Isto acontece porque é a classe que agrega todos os outros problemas que a empresa tem que resolver, e por isso, por causa da agregação, a diversidade e a variância é grande. Portanto, é previsível e expectável que não seja tão fácil prever esses tipos de problemas. As curvas PR, na Figura 36, destacam o desempenho razoável do modelo no que diz respeito a prever problemas de televisão e de internet.

De forma a contornar o problema relatado, relacionado com a classe 'Outros' e com vista a melhorar o desempenho, algumas abordagens foram testadas e são descritas a seguir.

Figura 33: Variáveis com mais influência na decisão.

	precision	recall	f1-score	support
DESATIVAÇÕES	0.38	0.16	0.23	23948
EQUIPAMENTOS	0.58	0.08	0.15	9622
FATURAÇÃO	0.27	0.07	0.11	19521
OUTROS	0.31	0.44	0.36	64662
PAGAMENTOS	0.32	0.23	0.27	16553
PROBLEMA - NET	0.42	0.46	0.44	53049
PROBLEMA - TLF	0.58	0.19	0.28	13200
PROBLEMA - TLM	0.45	0.07	0.13	4131
PROBLEMA - TV	0.47	0.70	0.56	89053
SERVIÇOS	0.38	0.20	0.26	40165
accuracy			0.41	333904
macro avg	0.42	0.26	0.28	333904
weighted avg	0.40	0.41	0.38	333904

Figura 34: Relatório de classificação do modelo criado.

A primeira opção tomada foi a de não treinar o modelo com a classe 'Outros' e associar a ela uma chamada, medindo a incerteza do modelo na escolha das restantes classes. De acordo com (Settles, 2012), a incerteza de um modelo pode definir-se de três formas diferentes:

- Aplicando a entropia de Shannon às probabilidades retornadas pelo modelo para cada classe. Quanto maior for a entropia, maior é a incerteza do modelo relativamente a qual é o problema do cliente;
- Verificar o quão alto é o valor relativo à classe com maior probabilidade. Ou seja, por exem-

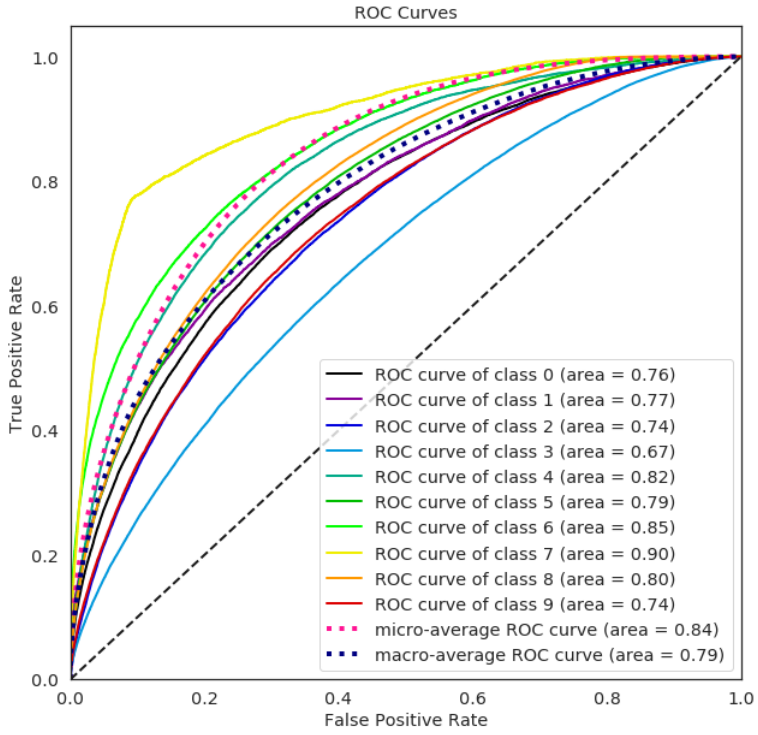


Figura 35: Curvas de ROC.

plo, se for 40%, provavelmente o modelo tem mais dúvidas do que se fosse 80%;

- Comparando o valor relativo à classe com maior probabilidade com o valor relativo à classe com a segunda maior probabilidade. Ou seja, olha-se para a distância entre os dois valores. Uma diferença pouco significativa pode revelar um nível grau de incerteza.

As três técnicas foram usadas e o micro-F1 obtido nunca superou o 0.41 obtido anteriormente.

O bom resultado indicado pelas curvas de ROC (AUROC = 0.84), conjugado com o micro-F1 obtido a partir do modelo de LightGBM otimizado, mostra que o modelo se comporta melhor em diferentes *thresholds* daqueles que são usado para o micro-F1, no qual cada observação é mapeada na classe com a maior probabilidade. Nesse sentido, uma análise do gráfico da Figura 35 foi feita no sentido de encontrar os *thresholds* ótimos, ou seja, os pontos das curvas onde o rácio entre verdadeiros positivos e falsos positivos é maior. Os *thresholds* atribuídos estão representados na Tabela 9. Ou seja, depois de definidos os *thresholds*, uma chamada só é atribuída a uma classe, se a probabilidade dada pelo modelo for superior ao *threshold* definido para essa classe. Esta é uma técnica que fornece bons resultados, no que diz respeito a otimizar o desempenho dos modelos. Neste caso, e eventualmente por ser um problema multiclasse, o micro-F1 também nunca superou o valor de 0.41 obtido anteriormente.

Tentou-se prever a classe 'Outros' através de uma deteção de anomalias, usando *Isolation Forest*, um modelo de árvores de decisão semelhante ao *Random Forest*, mas com foco na deteção de *outliers*. Os resultados não foram satisfatórios.

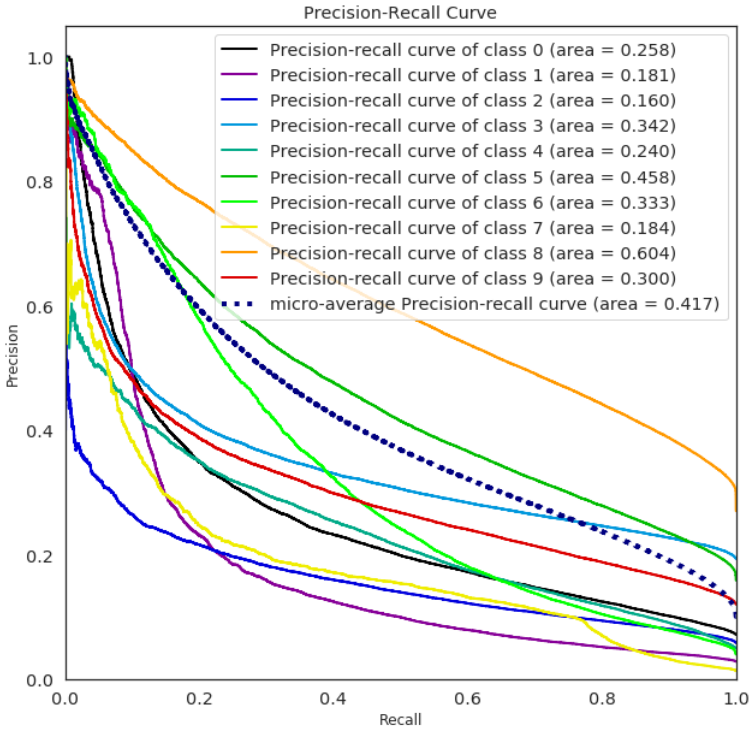


Figura 36: Curvas de PR.

Tabela 9: *Thresholds* definidos para cada uma das classes.

Classe	***Threshold* definido**
0 - Desativações	0.16
1 - Equipamentos	0.08
2 - Faturação	0.12
3 - Outros	0.21
4 - Pagamentos	0.85
5 - Internet	0.24
6 - Telefone	0.15
7 - Telemóvel	0.73
8 - Televisão	0.29
9 - Serviços	0.16

Por fim, a abordagem seguida foi a de combinar alguns modelos, anteriormente explorados, através de *voting classifier* (VC) e *stacking classifier* (SC). Os dois, como foi explicado no capítulo anterior, constituem abordagens algo distintas, o que levou a uma tomada de decisões divergente. Se por um lado, o VC, por ser baseado num sistema de votos, exige a combinação dos melhores modelos para estes dados, por outro, o SC exige uma gama diversificada de modelos, a fazerem diferentes suposições naquela que é a tarefa de previsão.

Assim, para o VC, escolheu-se combinar os dois melhores modelos. Como ficou provado na Tabela 6, estes são o LightGBM e o XGBoost. O LightGBM foi otimizado na secção 3.5, mas o XGBoost não. Portanto, o passo seguinte foi o de otimizá-lo utilizando uma abordagem semelhante à

utilizada com o LightGBM, ao nível de escolha de hiperparâmetros e da técnica utilizada (Otimização *Bayesiana*). Depois de 40 iterações, os hiperparâmetros encontrados para o modelo XGBoost foram os seguintes:

```
feature_fraction = 0.91,
colsample_bytree = 0.50,
subsample = 0.96,
max_depth = 2,
max_delta_step = 9,
gamma = 5,
min_child_weight = 14.3
```

O modelo de VC foi criado em modo *soft*, que como foi explicado anteriormente, faz a média das probabilidades dos modelos base. O resultado obtido para a métrica micro-f1 foi de 0.40.

Relativamente ao SC, utilizar uma gama diversificada de modelos, que fazem diferentes suposições sobre a previsão, é o recomendado. É, também, comum o uso de um meta-modelo simples, como por exemplo, um modelo linear (regressão linear ou logística), já que a própria tarefa de combinar os diferentes modelos não é complexa. Por isso, os modelos escolhidos para a combinação foram o LightGBM e o Random Forest. O meta-modelo escolhido foi a regressão logística. O resultado obtido para a métrica micro-f1 foi de 0.39.

Portanto, apesar do esforço feito, especialmente o computacional, a combinação dos diversos modelos, tanto a partir de VC, como de SC, não evidenciaram um resultado superior ao que já tinha sido alcançado aquando da otimização do modelo simples de LightGBM. É, por isso esse modelo que fica como o que consegue obter o melhor desempenho nestes dados e neste trabalho.

4. ALOCAÇÃO AO OPERADOR

Uma vez conhecida a razão da chamada do cliente, pode ser feita uma otimização para atribuir a chamada ao operador mais adequado para resolver o problema. Algumas técnicas de simulação são utilizadas para recriar um cenário passado. A seguir, descreve-se o módulo de otimização, como ilustrado na Figura 2.

4.1 MÉTODO

Como já foi visto anteriormente, o call-center da empresa pode ser representado por um sistema de fila de espera, próprio da teoria de filas de espera, no qual não existe uma distribuição inteligente de chamadas e qualquer operador pode ser confrontado com todos os problemas. Assim, assumindo capacidade ilimitada, população infinita, disciplina de atendimento FIFO, podemos representar o sistema por uma fila M/M/S, no qual S operadores podem atender em qualquer momento. Os clientes são impacientes porque podem cansar-se de esperar e cada operador só pode atender uma pessoa de cada vez.

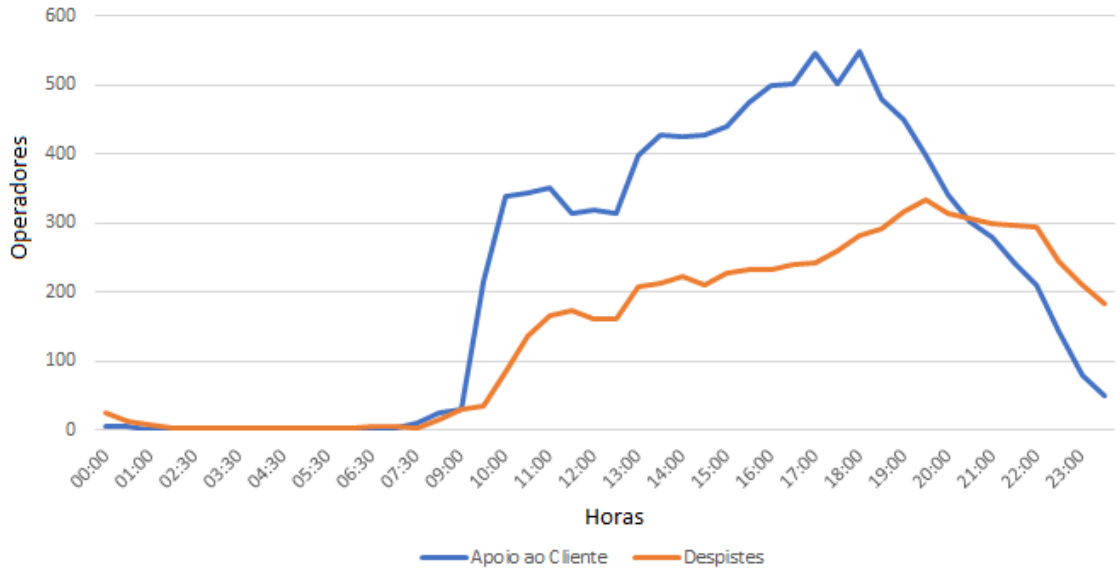


Figura 37: Operadores disponíveis para atender, por hora.

Para o desenvolvimento de um sistema de alocação inteligente é, de facto, necessário saber-se o quão eficientes e rápidos foram os operadores no passado. Ou seja, é necessário o registo de quanto duraram as chamadas. O que acontece, de facto, é que a empresa não tem essa informação guardada e torna-se inviável desenvolver este sistema para o serviço geral de apoio ao cliente. Todavia, a informação está disponível no que aos despistes técnicos diz respeito. Portanto, sabe-se, efetivamente, quando uma chamada para um despiste inicia, e quando termina. Deste modo, a alocação ao operador mais rápido é feita para os operadores técnicos. Relativamente ao serviço de apoio ao cliente, uma minimização do risco de reincidência será feita, ou seja, as chamadas

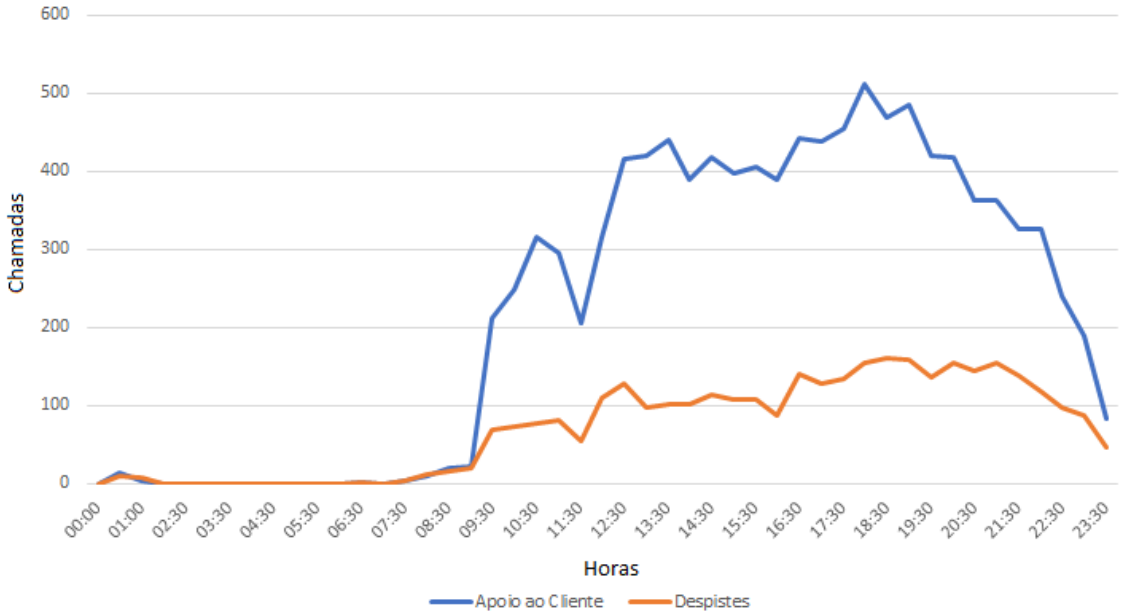


Figura 38: Taxa de chegada de clientes ao sistema, por hora.

serão reencaminhadas para os operadores que são mais eficientes a resolver uma chamada, no sentido de o problema relatado não reincidir nos quinze dias seguintes à chamada.

Para simular este sistema, foi necessário definir os períodos de trabalho dos operadores e recriar as chamadas dos clientes. Para o tornar fiável, os clientes podem desistir quando estão à espera há mais de 4 minutos, e os operadores fazem pequenas pausas, não inferiores a 30 segundos, entre diferentes chamadas. Para proceder à otimização, utilizou-se a biblioteca Simpy (SimPy, 2002) disponível em Python, que permite a simulação de eventos de uma forma básica, simples e intuitiva. As duas classes criadas, Operador e Cliente, estão ilustradas na Figura 39. Cada problema tem a si associado uma variável de tempo de serviço e uma percentagem de reincidência. Na Figura estão ilustradas as variáveis relativas a problemas de televisão e a problemas de internet. As variáveis relativas aos demais problemas foram ocultadas do diagrama.

Inicialmente foi necessário estimar o número de operadores disponíveis em cada momento. Esta estimativa foi extraída dos dados, ou seja, do que realmente aconteceu. Por exemplo, se um operador atendeu uma chamada às 16:00 horas de um determinado dia, isso significa que o trabalhador estava no seu posto de trabalho nessa altura. Assim, o horário de cada operador foi calculado. O passo seguinte tomado foi o de calcular o tempo de serviço e a taxa de reincidência de cada operador, com base nas chamadas atendidas no passado por estes, para cada um dos problemas. Ou seja, para um determinado dia, o tempo de serviço e a taxa de reincidência do João em problemas de televisão é calculado pela duração média das chamadas e pelas chamadas que reincidiram, relacionadas com a televisão e que ele atendeu até esse dia.

A seguir, apresenta-se algum pseudocódigo, onde se mostra, para os problemas de televisão e

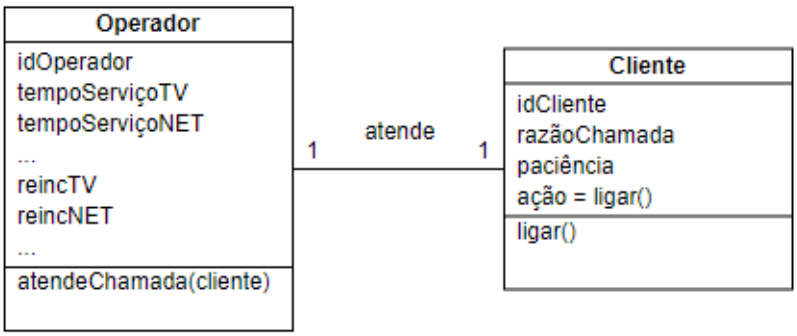


Figura 39: Diagrama de classes da simulação realizada.

de internet, o modo como a simulação é feita (ver Figuras 40 e 41).

```
# 1, o sistema minimiza tempos
# 0, o sistema minimiza reincidências
modo_sistema = 1

# cria lista de operadores disponíveis para atender
operadores_disponiveis = []

# inicialização da contagem de reincidências a 0
reincidências = 0

função atendeChamada(cliente):
    se cliente.razãoChamada igual a TV:
        timeout(tempoServiçoTV) # duração da chamada
    se cliente.razãoChamada igual a NET:
        timeout(tempoServiçoNET) # duração da chamada
    ...

    se (modo_sistema igual a 0):
        se cliente.razãoChamada igual a TV:
            se valor_aleatório igual a 1:
                incrementa reincidências
        se cliente.razãoChamada igual a NET:
            se valor_aleatório igual a 1:
                incrementa reincidências
        ...

    operador faz pausas entre 30 a 100 segundos
    operador fica novamente disponível

função ligar():
    requisita operador e espera

    se obtém operador:
        operador atende chamada
    se espera maior do que paciência:
        cliente desiste e desliga chamada
```

Figura 40: Pseudocódigo dos principais métodos da simulação.

A verificação das pessoas a trabalharem é feita a cada 30 minutos, e permite uma aproximação fidedigna, relativamente aos trabalhadores que estão ou estavam efetivamente disponíveis.

```
se (cliente.razãoChamada igual a TV):
    se (modo_sistema igual a 1):
        melhor_operador = min em tempoServiçoTV na lista operadores_disponiveis
    senão:
        melhor_operador = min em reincTV na lista operadores_disponiveis

se (cliente.razãoChamada igual a NET):
    se (modo_sistema igual a 1):
        melhor_operador = min em tempoServiçoNET na lista
            operadores_disponiveis
    senão:
        melhor_operador = min em reincNET na lista operadores_disponiveis

...
```

Figura 41: Pseudocódigo da escolha do melhor operador para atender.

```
a cada 30 minutos:
    para cada operador a trabalhar:
        sai do serviço se os próximos 30 minutos não estiverem no seu horário

    para cada operador que não está a trabalhar:
        entra no serviço se tem os próximos 30 minutos no seu horário
```

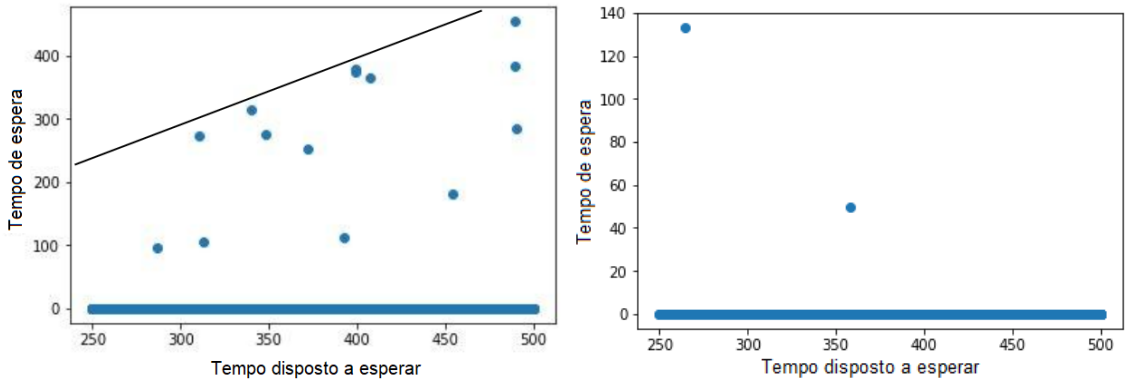
Figura 42: Pseudocódigo da gestão dos turnos dos operadores.

Depois de toda a informação estar disponível, a simulação foi executada, e os clientes foram recebidos no sistema, sabendo-se agora porque motivo estão a telefonar. Entre os operadores disponíveis, e de acordo com a opção definida, o mais rápido ou o mais eficaz a resolver problemas semelhantes no passado é aquele a quem a chamada é atribuída. No final da chamada, o operador torna-se novamente disponível para responder a uma nova chamada. Comparou-se o tempo de serviço simulado com a abordagem proposta e o número de reincidências geradas com o sistema em funcionamento agora.

4.2 RESULTADOS

A solução para este problema aumenta a satisfação do cliente. Na prática, isso acontece porque um cliente quer ser atendido o mais rapidamente possível (tempo é dinheiro), e um serviço eficiente mostra ao cliente o profissionalismo da empresa. Também, do lado da empresa, permite a redução de custos porque é possível atender mais chamadas no mesmo tempo ou com menos operadores. Os operadores ficam também mais satisfeitos em desempenhar as suas funções porque se sentem mais úteis ao atenderem chamadas nas quais estão preparados para ajudar da melhor forma.

A Figura 43 ilustra, também, a eficácia do sistema, visto que o tempo de espera é, na maior parte das vezes, nulo. Indica, também, que o número de operadores disponíveis é sempre superior às chamadas recebidas, ou seja, há excesso de mão de obra (excesso de servidores). Por outro lado, permite perceber os clientes que desistem de esperar (pontos acima da diagonal) e, quanto a isso, verifica-se que a grande maioria dos clientes não tem que esperar muito tempo até que um operador atenda.



(a) SAC

(b) Despistes

Figura 43: Tempo de espera, no sistema, em função do tempo que os clientes estão dispostos a esperar.

As Tabelas 10 e 11 mostram as comparações entre as soluções propostas neste documento e as soluções atuais, em que um cliente é atribuído a um operador aleatório. Podemos observar que é possível alcançar uma melhoria. Estes resultados, contudo, dependem consideravelmente da eficácia do modelo preditivo. A simulação mostra que, no caso da otimização ao operador mais rápido, a solução proposta pode reduzir o tempo de serviço e a carga de trabalho dos operadores em cerca de 40%. A simulação mostra também, no que diz respeito à otimização ao operador que gera menos reincidência, que é possível evitar que 52% dos clientes voltem a ter o mesmo problema. As duas soluções permitem não só uma melhor gestão dos recursos, mas também uma maior satisfação do cliente.

Tabela 10: Comparação entre as duas soluções para a alocação ao operador mais rápido.

Modelo	Tempo de Serviço Médio	Utilização dos Servidores
Solução Atual	7.9 minutos	15%
Solução Proposta	4.7 minutos	9%

Tabela 11: Comparação entre as duas soluções para a alocação ao operador que gera menos reincidência.

Modelo	Número de Reincidências Diárias
Solução Atual	2389
Solução Proposta	1236

5. CONCLUSÃO

Esta secção final conclui a dissertação resumindo o que foi abordado. Em seguida, são discutidos os resultados obtidos, bem como possíveis ideias para o melhorar, no futuro.

5.1 CONCLUSÃO

O mercado das telecomunicações tem registado um enorme crescimento nos últimos anos, atraindo vários clientes. Por esta razão, a competitividade é enorme, e as empresas tentam investir na sua infraestrutura para melhorar a relação com os seus clientes. Como as aplicações de IA estão a crescer na indústria e os muitos dados que têm estão disponíveis para utilização em qualquer altura e em qualquer lugar, espera-se que a aplicação de tais avanços tecnológicos conduza a uma melhor relação com os clientes e à sua satisfação. A maior parte dos esforços dos últimos anos tem-se centrado na previsão da desistência dos serviços, que é uma fase tardia de insatisfação dos clientes. Nesta tese, é feita uma tentativa de prever a razão da potencial insatisfação de um cliente ao ligar para o apoio ao cliente, no sentido de melhorar as operações internas e externas.

Assim, elaborou-se um modelo preditivo que antecipa o problema para o qual o cliente está a ligar, o que permite conhecer a tipologia da dificuldade antes do cliente a reportar. Para isso, foi necessário um foco em tarefas de tratamento e limpeza de dados e, também, na construção de novas variáveis. Seguiu-se a execução de diversos modelos de *machine learning*. Recorrendo-se, depois, a procedimentos de otimização, as chamadas são recriadas, sabendo-se agora o motivo para a razão desta, e os clientes são reencaminhados para o operador ou funcionário de serviço mais adequado, e que permite oferecer as melhores condições de assistência à pessoa naquele momento, para o tipo de chamada.

Os resultados do modelo de previsão foram satisfatórios, com a micro-F1 a 0.41 e o AUC-ROC a 0.84. Dão alguma confiança na utilização deste tipo de modelos e dados no processo de suporte. A otimização num cenário simulado indicou uma possível melhoria entre os 40% e os 50%.

Os benefícios resultantes desta solução são um serviço mais eficiente, ora porque os clientes são atendidos mais rápido, ora porque são atendidos de melhor forma, dado que a probabilidade de reincidência é menor. Espera-se que a produtividade dos trabalhadores saia também aumentada, visto que conseguem se sentir úteis e a desempenhar o seu trabalho com sucesso. A satisfação de um cliente, depois de atendido e ajudado, é maior o que não deixa de ser um fator relevante e importante num mercado e num setor tão concorrido como o da empresa em questão. Há também uma redução de custos porque pode-se atender mais pessoas, com menos operadores/funcionários, podendo estes ser alocados a outro tipo de serviços e trabalhos na empresa.

5.2 TRABALHO FUTURO

Apesar de tudo, algumas melhorias podem ser alcançadas através da recolha de mais dados e com alguma otimização adicional do melhor modelo obtido (LightGBM). Também a limitação em *hardware* não possibilitou uma exploração maior de modelos de redes neuronais que, possivelmente, poderiam originar resultados superiores. Tendo em conta os diversos dados disponíveis, outras variáveis importantes poderão ser criadas, levando a uma potencial melhoria nos resultados, ainda que ligeiramente. A exploração de outros modelos de *machine learning* é também uma das tarefas a fazer.

Alguns trabalhos de integração de dados e extração de variáveis foram necessários e computacionalmente dispendiosos para obter os resultados aqui mostrados. Por conseguinte, a tarefa de colocar esta solução em produção, última fase do CRISP-DM, é ainda uma tarefa árdua e seria o próximo passo para se verificar e provar se a melhoria e se os resultados aqui mostrados se podem de facto obter.

REFERÊNCIAS

Alencar, R. (2017). https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets#t1. Acedido em 5 de julho de 2020.

Ali, A. R. (2011). Intelligent call routing: Optimizing contact center throughput. *Proceedings of the 11th International Workshop on Multimedia Data Mining, MDMKDD'11 - Held in Conjunction with SIGKDD'11*, pages 34–43.

Benko, A. e Sik Lányi, C. (2011). History of Artificial Intelligence. [Acedido em 2019-12-27].

Bhalla, D. (2017). Understanding Bias-Variance Tradeoff. https://www.listendata.com/2017/02/bias-variance-tradeoff.html. Acedido em 20 de janeiro de 2020.

Bhargavi, K. e Jyothi, S. (2016). Classification of DNA Sequence Using Soft Computing Techniques: A Survey. *Indian Journal of Science and Technology*, 9(47):1–7.

Bojanowski, P., Grave, E., Joulin, A., e Mikolov, T. (2016). Enriching word vectors with subword information.

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., e Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Carneiro, D., Novais, P., Pêgo, J. M., Sousa, N., e Neves, J. (2015). Using mouse dynamics to assess stress during online exams. In Onieva, E., Santos, I., Osaba, E., Quintián, H., e Corchado, E., editors, *Hybrid Artificial Intelligent Systems*, pages 345–356, Cham. Springer International Publishing.

Chawla, N. V., Bowyer, K. W., Hall, L. O., e Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

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

Cortez, P. e Neves, J. (2000). Redes neuronais artificiais. -.

Costa, Â., Rincon, J., Carrascosa, C., Julián, V., e Novais, P. (2016). Emotions detection on an ambient intelligent system using wearable devices. *Future Gener. Comput. Syst.*, 92:479–489.

Fly, A. (2019). 5 Best Practices for AI- and Data-Driven Call Centers. https://towardsdatascience.com/5-best-practices-for-ai-and-data-driven-call-centers-647406b4234b. Acedido em 9 de junho de 2020.

Genesys (2014). Best Practices for Contact Center Routing. *Business white paper*.

Guru, B. T. (2016). What is artificial intelligence (ai)? https://www.besttechguru.com/future-technology-explained-what-is-artificial-intelligence/. Acedido em 16 de julho de 2020.

Hassan, R. S., Nawaz, A., Lashari, M. N., e Zafar, F. (2015). Effect of Customer Relationship Management on Customer Satisfaction. *Procedia Economics and Finance*, 23(October 2014):563–567.

Jafari Navimipour, N. e Soltani, Z. (2016). The impact of cost, technology acceptance and employees' satisfaction on the effectiveness of the electronic customer relationship management systems. *Computers in Human Behavior*, 55:1052–1066.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., e Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., e Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 3146–3154. Curran Associates, Inc.

Marwaha, R. (2014). Data Mining Techniques and Applications in Telecommunication Industry. *Artificial Intelligence in Medicine*, 16(1):1–2.

Mehrbod, N., Grilo, A., e Zutshi, A. (2018). Caller-Agent Pairing in Call Centers Using Machine Learning Techniques with Imbalanced Data. *2018 IEEE International Conference on Engineering, Technology and Innovation, ICE/ITMC 2018 - Proceedings*.

Messerly, J. G. (2015). Aristotle, Robot Slaves, and a New Economic System. https://ieet.org/index.php/IEET2/more/messerly20150527. Acedido em 5 de agosto de 2020.

Mikolov, T., Chen, K., Corrado, G. S., e Dean, J. (2013). Efficient estimation of word representations in vector space.

Mishra, K. e Rani, R. (2017). Churn prediction in telecommunication using machine learning. *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 2252–2257.

Muller, D. (2007). *Processos Estocásticos E Aplicações*. EDIÇOES 70 - BRASIL.

Murphy, K. P. (2013). *Machine learning : a probabilistic perspective*. MIT Press, Cambridge, Mass. [u.a.].

Nogueira, F. (2014–). Bayesian Optimization: Open source constrained global optimization tool for Python. `https://github.com/fmfn/BayesianOptimization`. Acedido em 9 de junho de 2020.

Novais, P., Costa, R., Carneiro, D., e Neves, J. (2010). Inter-organization cooperation for ambient assisted living. *J. Ambient Intell. Smart Environ.*, 2(2):179–195.

Pandey, P. (2019). Jupyter Lab: Evolution of the Jupyter Notebook.

Pereira, C. R. V. (2009). Uma Introdução Às Filas De Espera. -.

Pete, C., Julian, C., Randy, K., Thomas, K., Thomas, R., Colin, S., e Wirth, R. (2000). Crisp-Dm 1.0. *CRISP-DM Consortium*, page 76.

Preuveneers, D. e Novais, P. (2012). A survey of software engineering best practices for the development of smart applications in ambient intelligence. *J. Ambient Intell. Smart Environ.*, 4:149–162.

Randles, B. M., Pasquetto, I. V., Golshan, M. S., e Borgman, C. L. (2017). Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–2. IEEE.

Raschka, S. (2018). Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24).

Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, page 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Settles, B. (2012). *Active Learning*. Morgan & Claypool Publishers.

Shalev-Shwartz, S. e Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA.

SimPy (2002). SimPy: Discrete-Event Simulation for Python. `https://simpy.readthedocs.io/en/latest/`. Acedido em 9 de junho de 2020.

Snoek, J., Larochelle, H., e Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA. Curran Associates Inc.

Tundjungsari, V. (2013). Business Intelligence with Social Media and Data Mining to Support Customer Satisfaction in Telecommunication Industry. *International Journal of Computer Science and Electronics Engineering*, 1(1):1–4.

Yaswanth, M. (2019). `https://medium.com/@maniyaswanth123/bagging-and-boosting-a0b39e312117`. Acedido em 4 de julho de 2020.

APÊNDICES

Apêndice I - Publicações

Intelligent Call Routing for Telecommunications Call-Centers

Autores: Sérgio Jorge, Carlos Pereira e Paulo Novais

Título: Intelligent Call Routing for Telecommunications Call-Centers

Conferência: Intelligent Data Engineering and Automated Learning - IDEAL 2020

Ano de Publicação: 2020, aceite para publicação

Resumo: At telecommunications companies, call-centers have the highest interaction with customers, and the operators' performance is vital because an excellent service satisfies the customer and helps a better operation. Therefore, attempts are made to use customer data, call operator data, and historical service data to improve support. Pairing a customer with an operator who is comfortable with the problem to solve helps companies reducing costs, improves customer service, and increases employee productivity. In this article, we propose an approach based on machine learning and optimization, which predicts the problem for which the customer is calling and routes the call and the customer to the most appropriate call operator. The results show that using large amounts of business data along with innovative algorithms such as LightGBM can improve the customer support performance.