LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO

MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

PREVISÃO DO SUCESSO DE LEADS – UMA ABORDAGEM EM MACHINE LEARNING

JOÃO PEDRO GIL CUSTÓDIO

OUTUBRO - 2020

LISBON
SCHOOL OF
ECONOMICS &
MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO

MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

PREVISÃO DO SUCESSO DE LEADS – UMA ABORDAGEM EM MACHINE LEARNING

JOÃO PEDRO GIL CUSTÓDIO

ORIENTAÇÃO:
PROF. CARLOS COSTA
DR. JOÃO PAULO CARVALHO

OUTUBRO - 2020

AGRADECIMENTOS

É sem qualquer margem de dúvida que começo por agradecer à minha família, ao meu pai, Jorge, à minha mãe, Maria João, e à minha irmã Beatriz. Agradeço por todos os esforços, por toda a disponibilidade, e por todo o amor, carinho, confiança e dedicação. Sou uma pessoa grata pela educação que me deram, pelos valores que me incutiram, e pelo conforto familiar que me permitiu chegar até aqui. Quero fazer e ser como vocês.

Agradeço à Quidgest S.A., em particular ao Dr. João Paulo Carvalho, pela oportunidade em poder realizar o meu Trabalho Final de Mestrado sobre um tema que tanto gosto, num contexto empresarial e real. Agradeço também ao departamento de Consultoria de Negócio, por toda a disponibilidade e simpatia.

Ao Prof. Carlos J. Costa, agradeço por toda a disponibilidade, simpatia e dedicação, bem como por todos os conselhos e sugestões. Agradeço também a oportunidade e confiança de este trabalho ter possibilitado a publicação de um artigo científico.

Agradeço à minha mô Helena Miranda. Nutro um carinho especial por este trabalho pelo facto de teres estado presente em todos os momentos, desde a mensagem de boa sorte na noite anterior a entrar no estágio, até ao dia de hoje, em que continuas a apoiar-me e a incentivar-me. Aprecio-te.

E por fim, e nunca menos importante, aos meus amigos, em particular ao Alexandre Gomes, que tem vindo a ser uma das pessoas mais fundamentais na minha vida, tanto em termos académicos como em termos pessoais. Obrigado por todas as dicas, críticas e auxílio. És um irmão para mim. À Mariana Aguiar, ao João Rodrigues e ao David Mendes, o meu obrigado pela vossa amizade e companheirismo.

"*In God we trust,*
all others bring data." –
Dr. William Edwards Deming

RESUMO

Este Trabalho Final de Mestrado advém de um estágio realizado na empresa Quidgest S.A., em parceria com o Instituto Superior de Economia e Gestão (ISEG), no âmbito do mestrado de Métodos Quantitativos para a Decisão Económica e Empresarial (MQDEE). Este trabalho resulta da necessidade em aumentar o sucesso de vendas, através da previsão do sucesso de *leads*, recorrendo a técnicas e algoritmos de *Machine Learning*. Neste domínio, foi realizada uma prévia análise e compreensão do cenário empresarial, para que a utilização dos algoritmos pudesse oferecer uma adequada compatibilidade com os padrões de negócio da empresa. Previamente, foram aplicadas técnicas de tratamento, aumento e limpeza de dados, nomeadamente: SMOTE; ADASYN e TOMEK. Relativamente aos algoritmos de *Machine Learning*, foram utilizados algoritmos assentes no âmbito da aprendizagem supervisionada, nomeadamente: *Support Vector Machines*; *Random Forest*; *Neural Networks*; *Adaboost* e C5.0.

As etapas deste trabalho foram alinhadas com a metodologia CRISP-DM, que é uma das mais conhecidas abordagens no que toca à logística de processos para a construção de sistemas computacionais. Os *softwares* utilizados foram: *software* R (com recurso ao *R Studio*); *Microsoft SQL Server* e o GENIO, que é o *software* próprio da Quidgest S.A.

PALAVRAS CHAVE: *Adaboost*; ADASYN; Aprendizagem Supervisionada; C5.0; CRISP-DM; GENIO; *Leads*; *Machine Learning*; *Neural Networks*; *Software* R; *Random Forest*; SMOTE; SQL; *Support Vector Machines*; TOMEK.

ABSTRACT

This Master's Final Work results from an internship performed at Quidgest S.A., in a partnership with Instituto Superior de Economia e Gestão (ISEG), within the scope of the Master of Quantitative Methods for Economic and Business Decision (MQDEE). The purpose of this work is to increase the company's sales success, by predicting the success of leads, using Machine Learning techniques and algorithms. In this domain, a previous analysis of the business scenario was studied and analyzed, to allow that the usage of the algorithms could be suitable with the company's standard business processes. Then, techniques for processing, increasing and cleaning the data were applied, such as: SMOTE; ADASYN and TOMEK, and then, algorithms within the scope of supervised learning were used, such as: Support Vector Machines; Random Forest; Neural Networks; Adaboost and C5.0.

The sequential steps of this work were aligned with the CRISP-DM methodology, which is one of the most used tools in terms of logistics for building computer systems. The used softwares were: R software (using R Studio); Microsoft SQL Server and GENIO, which is the owned and created software by Quidgest S.A.

KEYWORDS: Adaboost; ADASYN; C5.0; CRISP-DM; GENIO; Leads; Machine Learning; Neural Networks; Random Forest; R Software; SMOTE; SQL; Supervised Learning; Support Vector Machines; TOMEK.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ACRÓNIMOS E SIGLAS

ACC – Accuracy

ADASYN – Adaptive Synthetic Sampling Method for Imbalanced Data

ADB – Adaboost

AUC – Area Under the Curve

B2B – Business-to-Business

BACC – Balanced Accuracy

BD – Base de Dados

CN – Consultoria de Negócio

CRISP-DM – Cross-Industry Standard Process for Data Mining

CV – Cross Validation

IA – Inteligência Artificial

ML – Machine Learning

NN – Neural Networks

ODBC – Open Database Connectivity

RF – Random Forest

ROC – Receiver Operating Characteristic

SMOTE – Synthetic Minority Over Sampling Technique

SQL – Structured Query Language

SVM – Support Vector Machines

TFM – Trabalho Final de Mestrado

1. INTRODUÇÃO

1.1 Contexto

A competitividade de mercado entre empresas impõe cada vez mais a necessidade de usar novas abordagens e tecnologias. Joshi (2020) afirma que tanto o *Machine Learning* (ML) como a Inteligência Artificial (IA), têm vindo a demonstrar ser soluções adequadas ao mundo moderno e repleto de dados em que vivemos, contribuindo para a evolução tecnológica que revoluciona as vidas e negócios de todos nós. Imensas empresas de grande nome investem cada vez mais em ML para otimizarem os seus negócios. É o caso da Apple, com a finalidade de melhorar os seus produtos, ou para aumentar a segurança dos dados dos utilizadores, ou por exemplo a Tesla, que usa ML e IA para construir carros inteligentes, que por sua vez são mais seguros e menos poluentes (Marr, 2019).

O ano de 2020 está a ser marcado por um cenário pandémico devido ao Coronavírus (SARS–CoV–2), mais conhecido por COVID – 19. Este vírus provoca sintomas idênticos a outras doenças virais, podendo ser confundido com uma pneumonia. Em março de 2020 foi demonstrada a eficiência dos TAC's (Tomografia Axial Computorizada) como medida eficaz no diagnóstico da doença. Rapidamente foram surgindo aplicações de ML no combate ao COVID – 19, através da aplicação de algoritmos capazes de identificar a presença do vírus no organismo humano. Este processo consiste em analisar as imagens dos TAC's, sendo possível determinar com alta precisão se um paciente está ou não infetado com COVID – 19 (Barstugan *et al*., 2020), realçando assim a versatilidade e as inúmeras possíveis aplicações das técnicas e algoritmos de ML.

A elaboração deste Trabalho Final de Mestrado (TFM) advém da realização de um estágio em parceria com a empresa Quidgest S.A., e com o Instituto Superior de Economia e Gestão (ISEG), no âmbito do mestrado de Métodos Quantitativos para a Decisão Económica e Empresarial (MQDEE). Teve a duração de 3 meses, principiou no início de novembro de 2019, e foi concluído em janeiro de 2020. Este estágio decorreu em conjunto com o departamento de Consultoria de Negócio (CN), o departamento responsável pelas *leads* (propostas a concursos públicos). A finalidade do estágio incidiu no desenvolvimento de um *software*, recorrendo a algoritmos de ML, capaz de prever o sucesso das *leads* da empresa.

Este trabalho possibilitou ainda a publicação de um artigo científico intitulado: *Success Prediction of Leads – A Machine Learning Approach*, apresentado na 15º Conferência Ibérica de Sistemas e Tecnologias de Informação (CISTI), realizada entre 24 a 27 de junho de 2020, em Sevilha, Espanha (Custódio *et al*., 2020).

1.2 Contexto organizacional: Quidgest S.A.

A Quidgest S.A. é uma empresa tecnológica de origem portuguesa, fundada em 1988. É pioneira na geração automática de *software* e aposta no desenvolvimento sustentável e na internacionalização de soluções e serviços. A empresa conta com vários certificados de qualidade ISO, nomeadamente: ISO 9001, ISO 14001 e ISO 27001, sendo também uma PME Excelência que integra as PMEs inovadoras da COTEC (Quidgest, 2019).

O GENIO, é um *software* de soluções *low code*, e é o principal meio de negócio da Quidgest. Descreve-se como uma plataforma de modelação e geração de sistemas de informação integrados, com diferentes arquiteturas e linguagens, tais como: MVC *(Model View Controller*), HTML 5, Java, C ++ e C #. É um *software* que se baseia nos conceitos de MDD (*Model Driven Development*), RAD (*Rapid Application Development*) e *DevOps*, sendo útil para projetos complexos, únicos e urgentes. Uma das premissas do GENIO é o facto de ser possível construir um solução à medida para cada negócio, tornando-se até 10 vezes mais rápido do que um *software* idêntico programado à mão, sendo também de fácil manutenção e customização (Quidgest, 2020).

Parte da estrutura de negócio da Quidgest S.A. passa por realizar propostas a concursos públicos com a finalidade de ganhar projetos e clientes. A empresa vende o seu *software*, o GENIO, e utiliza o mesmo para desenvolver soluções à medida do cliente. Entende-se por *lead*, uma oportunidade de negócio que advém do interesse de um indivíduo ou de uma organização, nos produtos ou serviços que uma empresa comercializa. Torna-se então fulcral prever o sucesso dos novos *leads*, para melhorar a eficácia das propostas a concursos públicos, possibilitando assim um aumento nas vendas.

O departamento de CN é um departamento essencial dentro da empresa. Os seus elementos têm um papel fundamental na procura de novas oportunidades de negócio, na manutenção dos clientes e dos negócios já existentes, e é ainda o departamento responsável pela elaboração de propostas a *leads* de concursos públicos.

Várias empresas recorrem ao mesmo modelo de negócio, tornando-se assim empresas concorrentes à Quidgest. Cada empresa realiza as suas propostas, e no final de cada concurso, são divulgados os relatórios preliminares onde constam os resultados sobre o vencedor, bem como as pontuações que cada empresa obteve em cada critério de adjudicação. Os critérios de adjudicação, são parâmetros que ajudam na transparência dos concursos, e são utilizados como método de avaliação das propostas. O preço, a qualidade técnica, ou o prazo de entrega, são exemplos comuns de critérios de adjudicação.

1.3 Objetivos do estágio

Este estágio tem como principal objetivo propor uma solução para a previsão do sucesso de *leads*. Mais detalhadamente, os objetivos são: identificação do contexto empresarial e dos dados a utilizar; identificar as soluções mais adequadas de tratamento, aumento e limpeza de dados; identificar os melhores algoritmos de ML que permitam prever com assertividade o sucesso das *leads*, avaliando e comparando a qualidade dos mesmos; e finalmente, desenvolver um *software* que incorpora a solução proposta.

1.4 A estrutura do TFM

A estrutura deste TFM consiste em: no 2º capítulo é realizada a revisão da literatura, onde é demonstrada a importância do ML nas organizações, e onde é feita uma detalhada descrição de todas as metodologias, algoritmos e técnicas que foram utilizadas no tratamento de dados e na estimação dos modelos preditivos. O 3º capítulo aborda a metodologia utilizada (CRISP-DM), onde é demonstrado o alinhamento entre as várias etapas da metodologia, e os passos que foram realizados no processo de estimação de modelos, bem como no processo de construção do *software* de previsão de *leads*. O 4º capítulo diz respeito ao desenvolvimento prático, onde são detalhadas todas as etapas realizadas, bem como os resultados que foram sendo obtidos, os quais permitiram conduzir o processo desde a recolha de dados, ao desenvolvimento da Base de Dados (BD), à aplicação e seleção dos algoritmos de ML, e é ainda apresentado o *layout* da ferramenta final. Finalmente, no 5º e último capítulo, é feita a discussão dos resultados e são apresentadas as conclusões, bem como as propostas para trabalhos futuros.

2. REVISÃO DA LITERATURA

"*Se o teu principal concorrente está a apressar-se a desenvolver ferramentas de IA e tu não, isso irá esmagar a tua empresa*." – Elon Musk

Em: US News (2017)

2.1 Introdução

A revisão da literatura tem como finalidade enquadrar o objetivo, com as ferramentas que devem ser usadas para o alcançar. No caso, o objetivo passa por desenvolver um *software*, usando dados reais e algoritmos de ML, com a finalidade de obter um modelo de previsão assertivo para o sucesso de *leads*. Para tal, foi necessário estudar sobre as contribuições do ML nesta temática, bem como sobre quais são as melhores práticas no que toca ao tratamento de dados, quais os melhores e mais adequados algoritmos, e quais as formas mais adequadas para avaliar e selecionar o melhor, entre todos os modelos.

2.2 A importância do *Machine Learning* no B2B

B2B, *Business-to-Business* em inglês, ou Serviço de Empresas para Empresas em português, é uma das grandes áreas de foco relativamente a aplicações de ML. São geralmente empresas privadas que desenvolvem ferramentas tecnológicas de previsão, com o intuito de uso próprio e interno. Algoritmos baseados em árvores de decisão, especialmente o *Random Forest* (RF), têm mostrado bons resultados na assertividade de previsões em negócios B2B (Mortensen *et al*., 2019).

A dinâmica de negócio empresarial é tradicionalmente baseada em modelos mentais e subjetivos, que advêm da experiência dos profissionais. No entanto, estudos têm demonstrado que as empresas são mais eficientes quando tomam decisões com base em modelos estatísticos, que são obtidos através de dados reais (Bohanec *et al*., 2017). Isto tem incentivado à adoção de algoritmos inteligentes, como por exemplo: *Support Vector Machines* (SVM); RF; *Neural Networks* (NN) e modelos com o paradigma *Boosting* (que será explicado no subcapítulo 2.11), que são capazes de prever tendências e comportamentos dos elementos envolvidos num negócio B2B (Bohanec *et al*., 2017).

Num outro estudo dedicado à implementação de ML para previsão de vendas em B2B, algoritmos baseados em árvores de decisão, nomeadamente o algoritmo RF, mostraram boa capacidade preditiva (Bohanec *et al*., 2015), reforçando assim a pertinente utilidade deste tipo de algoritmos, no contexto deste trabalho.

Existem várias metodologias possíveis para a realização de cada projeto, mas alguns passos são considerados obrigatórios e fundamentais em todos eles, tais como: a recolha, o tratamento e a limpeza de dados; a normalização das variáveis; a *feature selection* e a análise de correlações das mesmas.

2.3 Os tipos de aprendizagem em *Machine Learning*

Tom Mitchell, é um cientista da computação (*alma mater* – MIT), reconhecido como um dos mais importantes nomes dentro do ramo do ML. Tom Mitchell (1997), define o ML como um conjunto de técnicas e algoritmos em que:

"Um programa aprende com a experiência E, em relação a uma determinada tarefa T, com uma medida de desempenho P, se o seu desempenho na tarefa T, medido por P, melhora com a experiência E".

Em: Mitchell (1997), p. 2.

Existem vários tipos de aprendizagens na temática do ML, entre elas: a supervisionada; a não supervisionada; a semi-supervisionada; a *reinforcement* e uma das mais recentes, a *self-learing*.

Este estágio insere-se no ramo da aprendizagem supervisionada. Na aprendizagem supervisionada, cada observação de um conjunto de dados tem uma *label* que descreve cada registo. O objetivo da utilização de algoritmos de ML neste tipo de aprendizagem, tem como finalidade a previsão dessa *label*. Um possível exemplo inserido num cenário de resposta binária, pode dar-se pela necessidade de prever se uma determinada observação diz respeito a um negócio que foi concretizado ou não, sendo por norma o valor 1 associado à *label* dos casos positivos, e o valor 0, associado à *label* dos casos negativos. Por outro lado, a aprendizagem não supervisionada não possuí essa *label*, e tais conjuntos de dados são mais adequados para análises descritivas, recorrendo por exemplo a algoritmos de *clustering*, ou a técnicas de redução de dados, tais como: Análise em Componentes Principais ou Análise Fatorial (Burkov, 2019; Hendrycks *et al*., 2019).

2.4 *Cross Validation K – Fold*

O *Cross Validation* (CV) *K – Fold* é uma técnica estatística, usada para estimar a capacidade de generalização de modelos preditivos. É também útil para determinar quais as variáveis mais relevantes de um determinado *data set*, permitindo assim alcançar estimações mais assertivas, e permitindo ainda evitar o problema do *overfitting*[1] (Berrar, 2018; Hastie *et al*., 2009). O CV *K – Fold* é um dos métodos geralmente mais adotados, e é especialmente fácil de executar em pequenos conjuntos de dados, pois esta técnica tem altos requisitos computacionais (Kuhn & Johnson, 2013).

O CV *K – Fold* permite selecionar aleatoriamente ***k*** subconjuntos com dimensão uniforme. Um dos ***k*** subconjuntos é deixado de fora em todas as iterações, pois o CV *K – Fold* irá usar esse subconjunto (conjunto de teste) para avaliar o modelo, que é estimado com os dados dos restantes ***k*** – 1 subconjuntos (conjunto de treino). Assim, irão obter-se as métricas de desempenho obtidas ao treinar os ***k*** cenários, sendo importante relembrar que todos esses cenários são estimados com o mesmo modelo estatístico.

Assim, a seleção do melhor modelo não será apenas com base numa única repartição aleatória dos dados, mas sim com base em ***k*** repartições. Portanto, os resultados de performance de cada modelo serão a média de cada uma das respetivas métricas, calculadas em cada um dos ***k*** cenários (Torgo, 2016) (Fig.1), permitindo assim obter uma avaliação mais realista do quão bem cada modelo se consegue adaptar aos dados.

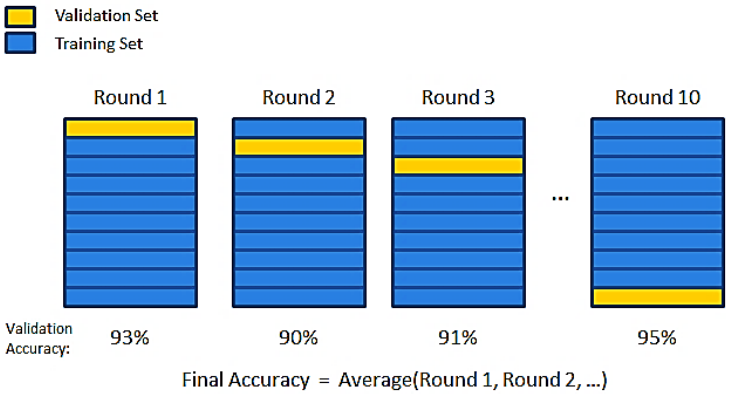


Figura 1 – Exemplo de *Cross Validation K – Fold*
Fonte: (Kaza, 2018)

[1] ***Overfitting*** é o cenário em que um modelo explica demasiado bem um determinado fenómeno durante o processo de treino, mas apresenta resultados pobres no processo de teste. Genericamente, este problema advém de uma incorreta limpeza de dados, ou de uma incorreta *feature selection* (Lantz, 2013).

2.5 SMOTE

O algoritmo SMOTE (*Synthetic Minority Over Sampling Technique*), é uma técnica de aumento da dimensão dos dados para a amostra de treino. Além de permitir aumentar o volume de dados no seu todo, permite também aumentar o volume de dados pertencentes à classe minoritária, o que possibilita uma melhor perceção dessa classe por parte dos algoritmos de ML, alcançando assim modelos com maior qualidade preditiva.

O processo de funcionamento deste algoritmo baseia-se na identificação de cada observação da amostra de treino, pertencente à classe minoritária, e, de seguida, o SMOTE gera amostras sintéticas (dados artificiais) no espaço dimensional que corresponde ao vetor que une cada registo da classe minoritária, aos ***k*** vizinhos mais próximos da respetiva observação (Fig.2) (Chawla *et al*., 2002).

O SMOTE é uma técnica de *oversampling*, sendo particularmente útil em casos de dados desbalanceados. No entanto, no processo de criação de observações artificiais, o algoritmo por vezes provoca uma expansão do *cluster* da classe minoritária, introduzindo observações artificiais da classe minoritária, no espaço dimensional da classe maioritária, podendo assim provocar *overfitting* (Cruz-Jesus *et al*., 2020). Uma possível solução para este problema consiste na utilização do algoritmo TOMEK, que será explicado no subcapítulo 2.7.

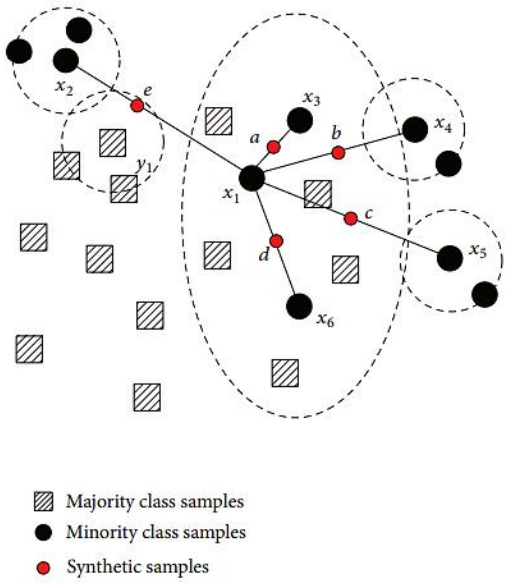


Figura 2 – Exemplo de aplicação – SMOTE
Fonte: (Hu & Li, 2013)

2.6 ADASYN

O algoritmo ADASYN (*Adaptive Synthetic Sampling Method for Imbalanced Data*) deriva do algoritmo SMOTE. Este algoritmo foi desenhado para reduzir a quantidade de dados desbalanceados entre as classes, e para ajustar os limites de classificação (Aditsania, 2017). O conceito deste algoritmo debruça-se na utilização de uma função de distribuição de densidade, como critério de seleção automático do número de observações artificiais, que vão ser criadas para cada observação da classe minoritária (Fig.4). Cada observação original da classe minoritária tem um *weigth* (peso) associado, que advém do nível de dificuldade em prever essa observação. Logo, quanto mais difícil for uma observação de ser prevista, maior o número de observações artificias que serão criadas na sua vizinhança (Haibo He *et al*., 2008). O algoritmo consiste em:

Algoritmo ADASYN

1. Começa por calcular o grau de dados desbalanceados segundo a classe minoritária;
2. Para cada observação da classe minoritária, o ADASYN encontra os **k** vizinhos mais próximos de ambas as classes, baseados na distância Euclidiana (por *default*);
3. De seguida, determina o número ideal de observações artificiais que devem ser geradas na vizinhança de cada observação da classe minoritária, tendo em conta o rácio entre a quantidade de vizinhos mais próximos pertencentes à classe maioritária, e o total de vizinhos mais próximos;
4. Por fim, gera dados artificiais, criando observações no espaço dimensional que corresponde ao vetor que une cada registo da classe minoritária, aos **k** vizinhos mais próximos da mesma classe.

Figura 3 – Algoritmo ADASYN
Fonte: (Haibo He *et al*., 2008)

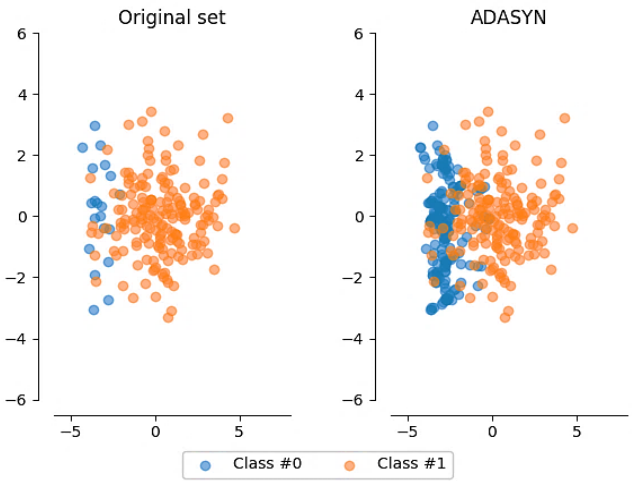


Figura 4 – Exemplo de aplicação – ADASYN
Fonte: (Aridas & Lemaitre, 2018)
Acedido a: 20-03-2020

2.7 TOMEK

TOMEK ou *Tomek – Links* é um algoritmo de *undersampling*. O objetivo deste algoritmo é eliminar observações que têm maior probabilidade de ser incorretamente classificadas (Devi *et al*., 2017). Este algoritmo pode ser considerado um algoritmo de limpeza de dados, e pode ser aplicado tanto em *data sets* originais, como em *data sets* que já contêm observações artificiais geradas por algoritmos de *oversampling*, como por exemplo o SMOTE e o ADASYN.

O conceito principal deste algoritmo é: dadas duas observações: $\boldsymbol{x_i}$ e $\boldsymbol{x_j}$ pertencentes a duas classes diferentes, e, sendo $\boldsymbol{d(x_i, x_j)}$ a distância entre $\boldsymbol{x_i}$ e $\boldsymbol{x_j}$, um "*Tomek – Link"* (apresentado no segundo gráfico da figura 5), ocorre quando não existe nenhuma amostra $\boldsymbol{x_k}$ tal que $\boldsymbol{d(x_i, x_k) < d(x_i, x_j)}$ ou $\boldsymbol{d(x_j, x_k) < d(x_i, x_j)}$ (Batista *et al*., 2004). Ou seja, se existir um "*Tomek – Link"* entre duas observações, isso indica-nos que ou uma das duas observações é ruído, ou ambas são observações fronteira (observações que se encontram no limite de separação das classes em estudo).

Assim, o algoritmo TOMEK é utilizado para remover essas observações que causam ruído, que geralmente pertencem à classe maioritária (Thai-Nghe & Schmidt-Thieme, 2010). Após essas observações serem removidas do *data set*, os algoritmos de ML tendem a melhorar a sua capacidade preditiva, uma vez que a separação dos dados passa a ser menos complexa (Fig.5).

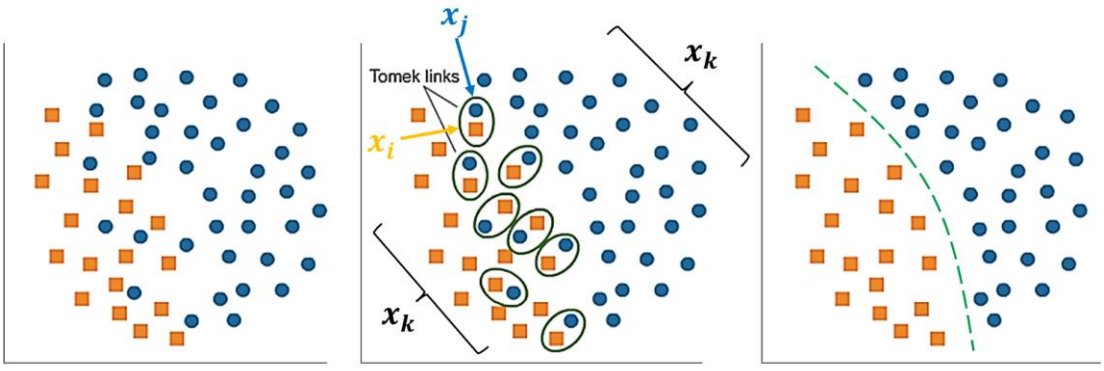


Figura 5 – Exemplo de *Tomek – Links*
Fonte: (Alencar, 2017)
Acedido a: 25-09-2020

2.8 *Support Vector Machines*

Luís Torgo (2016) descreve as *Support Vector Machines* como um algoritmo de ML que pode ser aplicado a problemas de regressão e de classificação, sendo um dos algoritmos com melhor e mais robusto *background* teórico. Os primeiros modelos de SVM foram desenvolvidos nos anos 60 por Vladimir Vapnik, e têm-se tornado uma das ferramentas mais eficientes na área do ML (Kuhn & Johnson, 2013).

Este algoritmo tem o seguinte comportamento: as SVM determinam $\boldsymbol{x}$ *input vectors,* num elevado espaço dimensional $\boldsymbol{Z}$. Neste espaço dimensional, são construídos *hyperplanes* que serão as várias tentativas candidatas a serem a fronteira ideal, que irá servir como separação de cada uma das classes existentes no *data set* (Vapnik, 2000) (Fig.6). Vapnik definiu uma métrica chamada ***margin***, que possibilita melhorar a eficácia dos modelos SVM através do seu *tuning*. A margem é a distância entre a fronteira de classificação de cada *hyperplane,* e a observação mais extrema de cada uma das classes do conjunto de dados. Existe ainda um outro parâmetro personalizável chamado ***cost*** que indica o custo de violação da margem (Torgo, 2016; Vapnik, 2000).

Assim, o objetivo consiste em maximizar a margem, o que se traduz em alcançar uma distância mínima entre o *hyperplane* e os diferentes *clusters* das classes dos dados. Um possível *hyperplane* de classes linearmente separáveis pode dar-se por:

$$\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b} = \boldsymbol{0}$$

onde $\boldsymbol{w}$ é o vetor dos pesos (*weights*), " · " representa o produto escalar, e $\boldsymbol{b}$ é o vetor de enviesamento. De seguida, é necessário criar uma função de classificação, dada por:

$$\boldsymbol{D}(\boldsymbol{x}) = \boldsymbol{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})$$

Os valores positivos do *output* da função $\boldsymbol{D}(\boldsymbol{x})$ são atribuídos à classe +1, e logicamente os valores negativos são atribuídos à classe –1, sendo $\boldsymbol{sign}$ a nomenclatura dada à função que extraí o sinal do resultado obtido através de $\boldsymbol{D}(\boldsymbol{x})$ (Liboni, 2017).

Uma das vantagens da utilização deste algoritmo denomina-se por ***Kernel Trick***. Este, permite ao algoritmo a extensão da sua natureza linear, para naturezas de classificação não lineares, originando uma melhor adaptação das SVM aos dados em estudo. Não é expectável que todos os *data sets* tenham um comportamento em que seja possível classificar as classes com apenas um *hyperplane* de origem linear. Logo, surge a

necessidade de recorrer a outras transformações. As transformações mais comuns são a transformação linear; a polinomial ou a transformação com a função de base radial (RBF).

$$\boldsymbol{Linear} = \boldsymbol{x'u}$$
$$\boldsymbol{Polinomial} = (\boldsymbol{\Phi(x'u)} + \mathbf{1})^{\boldsymbol{grau}}$$
$$\boldsymbol{RBF} = \mathbf{exp}\,(-\boldsymbol{\sigma}\,||\boldsymbol{x} - \boldsymbol{u}||^{\mathbf{2}})$$

Devem, portanto, ser testadas várias configurações dos parâmetros das SVM, para que seja possível encontrar a melhor adaptação aos dados em estudo (Kuhn & Johnson, 2013).

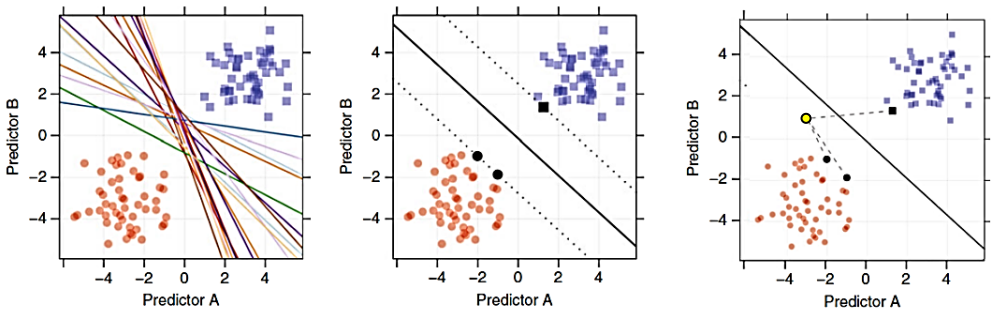


Figura 6 – Exemplo de aplicação – SVM
Fonte: (Kuhn & Johnson, 2013)

No primeiro gráfico (Fig.6), podemos observar um enorme conjunto de *hyperplanes* candidatos a vetor fronteira ideal. No segundo gráfico (Fig.6), podemos observar o *hyperplane* ideal que servirá como fronteira para classificação das duas classes dos dados, ou seja, acima do *hyperplane* iremos ter as observações que vão ser classificadas como "roxo", e abaixo do *hyperplane* as observações vão ser classificadas como "laranja". No terceiro gráfico (Fig.6), podemos ver a margem a tracejado cinza, que irá ajudar a classificar a nova observação (ponto amarelo), que será classificada tendo em conta a menor distância da margem, aos respetivos pontos extremos de cada classe do *data set*. Assim, a nova observação a amarelo será classificada como pertencente à classe "laranja".

O algoritmo SVM oferece uma limitação no que toca ao seu *output*, uma vez que apenas permite obter resultados binários (ou –1, ou +1), e neste trabalho, é necessário calcular a probabilidade de cada observação. John Platt (2000) desenvolveu o método de Platt, que permite ultrapassar esta limitação do SVM. Este método consiste em aplicar uma regressão logística ao *output* dos resultados, possibilitando assim alcançar a probabilidade associada a cada observação, dada por: $P(y = 1 \mid x) = \frac{1}{1+exp(Af(x)+B)}$, onde os parâmetros *A* e *B* são obtidos através do método da máxima verossimilhança.

2.9 *Random Forest*

O *Random Forest* é um algoritmo de ML composto por árvores de decisão agregadas, e é um dos melhores métodos de *ensemble*[2] tanto para classificação como para regressão (Choudhury & Gupta, 2018; Patel *et al*., 2018). É um algoritmo poderoso dado que permite combater o *overfitting*, não oferece limitações em cenários com elevadas dimensões de dados, e é ainda capaz de lidar com *missing values*. Porém, oferece pouco controlo relativamente aos seus parâmetros de configuração (Kuhn & Johnson, 2013).

O processo de funcionamento do algoritmo RF é: perante um conjunto de dados de treino, o algoritmo gera amostras *bootstrap*[3] (selecionando aleatoriamente ***n*** observações com reposição), e o modelo "aprende" com as correspondentes e aleatórias árvores de decisão de resposta binária (Genuer *et al*., 2017). Assim, irá obter-se o melhor modelo possível (*strong learner*) que é gerado através de *weak learners* (representados pelas raízes a vermelho na árvore) (Fig.7), seguindo assim o paradigma *Bagging*.

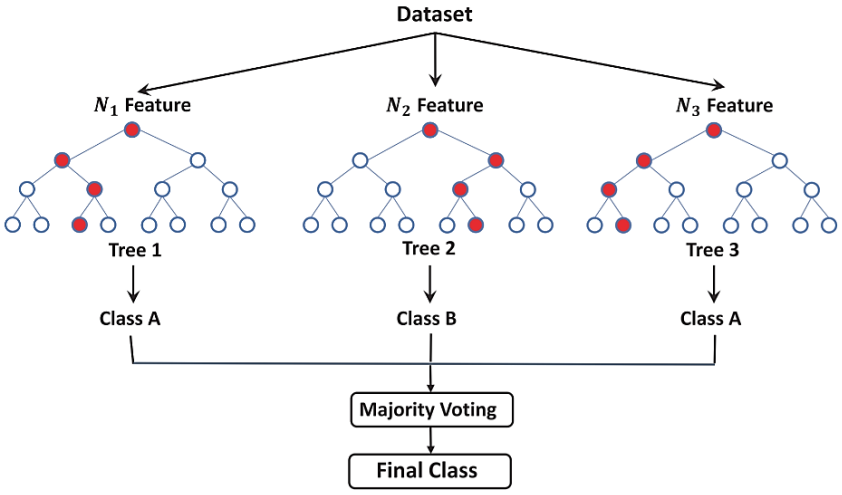


Figura 7 – Diagrama RF
Fonte: (Tahmasebi *et al*., 2020)

[2] ***Ensemble*** é um paradigma de aprendizagem em que em vez de a aprendizagem ser feita apenas com um único modelo poderoso, vários modelos mais fracos são estimados com a finalidade de serem agrupados, proporcionando um modelo ainda mais poderoso, que o modelo singular (Burkov, 2019).

[3] Amostras ***bootstrap*** são amostras aleatórias com reposição. Isto significa que depois de um registo ser classificado como pertencente a uma das classes do conjunto de dados, ele ainda pode voltar a ser selecionado para uma reavaliação. Muitas das observações voltam a ser reavaliadas e outras nem sequer chegam a ser selecionadas para avaliação, sendo normalmente estas as observações "*out-of-bag*". Assim, os modelos são construídos com a finalidade de usar a informação das várias observações com múltiplas avaliações *bootstrap*, para tentar prever as observações "*out-of-bag*" (Kuhn & Johnson, 2013).

No âmbito de ML, existem dois conceitos de *ensemble learning*: *Bagging* e *Boosting* (que será explicado no subcapítulo 2.11). O conceito de *Bagging* é baseado na criação de várias cópias do conjunto de dados de treino, em que cada cópia é marginalmente diferente das restantes. Cada uma dessas cópias da amostra de treino, é colocada dentro de um "*bag*" (saco), representando assim um conjunto de dados único, que será utilizado para treinar um modelo individual.

Assim, um *weak leaner* é aplicado a cada cópia do conjunto de dados para se obterem múltiplos *weak models*, para que no final os melhores modelos sejam incorporados num só. Isto é, enquanto por exemplo numa regressão linear simples, existe apenas um modelo que explica ou prevê um determinado fenómeno, no paradigma *Bagging* são estimados vários modelos do mesmo algoritmo, que explicam diferentes partes do fenómeno, e no final, incorporam-se todos esses modelos para permitir uma explicação mais abrangente e conjunta (*ensemble*). O RF é portanto um tipo de algoritmo que se insere no paradigma *Bagging* (Burkov, 2019).

Algoritmo *Random Forest* para Regressão e Classificação

1. De $\boldsymbol{b} = 1$ até $\boldsymbol{B}$:

a) Selecionar uma amostra *bootstrap* $\mathbf{Z}^*$, de dimensão $\boldsymbol{N}$, entre os dados de treino
b) Produzir uma árvore *Random Forest* $\mathbf{T_b}$ para os dados *bootstrap*, repetindo recrusivamente os passos seguintes, em cada nó terminal da árvore, até que o mínimo de nós pré definidos ($\boldsymbol{n_{min}}$), sejam atinguidos:

   i. Selecionar aleatóriamente $\boldsymbol{m}$ variáveis, das $\boldsymbol{p}$ variáveis
   ii. Escolher o melhor ponto de separação entre as $\boldsymbol{m}$ variáveis
   iii. Separar o nó (pai), em dois nós (filhos)

2. Recolher o *output* das árvores *ensemble* $\{\mathbf{T_b}\}_{\mathbf{1}}^{\mathbf{B}}$.

Para realizar uma previsão de um novo registo:

Regressão: $\hat{\mathbf{f}}_{\mathbf{rf}}^{\mathbf{B}}(\boldsymbol{x}) = \frac{1}{\mathrm{B}}\sum_{\mathrm{b}=1}^{\mathrm{B}} \mathbf{T_b}(\boldsymbol{x})$

Classificação: Assumindo que $\hat{\mathbf{C}}_{\mathbf{b}}(\boldsymbol{x})$ é a classe de previsão da $\boldsymbol{b}^{\text{a}}$ árvore de RF

Então $\hat{\mathbf{C}}_{\mathbf{rf}}^{\mathbf{B}}(\boldsymbol{x})$ = voto maioritário dos vários $\{\ \hat{\mathbf{C}}_{\mathbf{b}}(\boldsymbol{x})\ \}_{\mathbf{1}}^{\mathbf{B}}$

Figura 8 – Algoritmo RF
Fonte: (Hastie *et al*., 2009)

2.10 *Neural Networks*

Neural Networks, ou Redes Neuronais em português, são algoritmos de ML que se baseiam em *layers* de decisão para treinar e detetar e prever o comportamento de um determinado *data set*. O comportamento do cérebro humano deu origem ao nome deste algoritmo, e apesar das NN não serem uma representação virtual dos neurónios do cérebro, há de facto semelhanças na forma como ambos processam a informação para determinar o resultado de uma determinada ocorrência (Lantz, 2013; Mitchell, 1997).

Yoav Goldberg (2017) descreve o *Deep Learning* como um ramo do ML, que advém de um *rebranding* moderno das NN. Existem atualmente dezenas de diferentes tipos de NN, e cada uma delas contêm funções matemáticas diferentes. São exemplos de NN: *Multi-Layer Perceptron*; *Feed-Forward Neural Networks*; *Recurrent Neural Networks*; *Markov-Chain*; *Auto Encoder* e *Extreme Learning Machine* (Leijnen & Veen, 2020).

Assim, o conceito de *Deep Learning*, resulta do uso em simultâneo de vários tipos de NN, proporcionando a agregação de variadas funções matemáticas, originando então uma aprendizagem mais profunda e robusta.

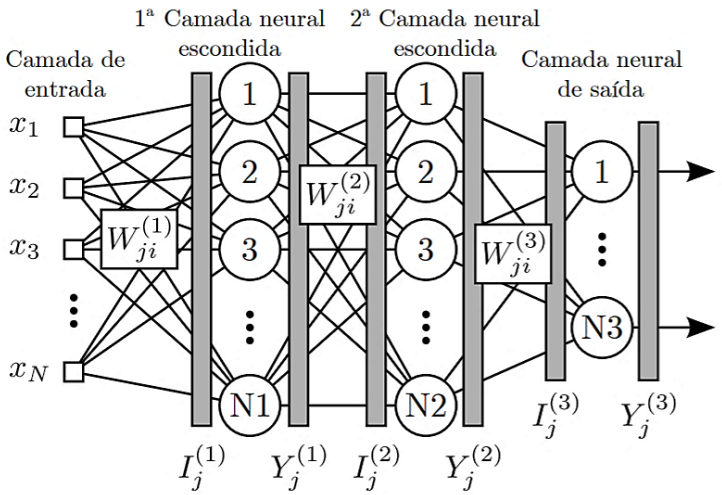


Figura 9 – Esquema genérico de uma *Neural Network*
Fonte: (Suetake, 2012)

Na figura anterior (Fig.9), está representada a arquitetura de uma NN de classificação genérica, que possui o seguinte comportamento:

Cada variável inicial está associada a uma *input layer* $(\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \ldots, \boldsymbol{x_N})$, que representa um estímulo aos vários neurónios. Seguidamente, cada *weight* $(\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \ldots, \boldsymbol{w_N})$ representa o peso que cada neurónio terá na seguinte camada de neurónios. Dentro de cada neurónio, em cada *hidden layer*, será realizada a combinação linear – $\sum$ – de todos os *inputs* que esse neurónio recebe, sendo $\boldsymbol{u}$, o resultado de cada combinação linear, que terá ainda associada uma variável de ativação do neurónio, representada por $\boldsymbol{\theta}$, que indica se esse neurónio irá prosseguir para a próxima *layer*. Cada neurónio tem na sua essência, uma função de ativação $\boldsymbol{g}(\boldsymbol{u})$, que é responsável por limitar a saída de cada neurónio perante certos intervalos de valores pré-definidos. Finalmente, o *output* gerado pelas várias camadas de neurónios, dado por $\boldsymbol{y}$, corresponde à previsão baseada na aprendizagem ao longo da rede neuronal (Liboni, 2017; Mitchell, 1997).

As funções matemáticas dentro de cada neurónio podem tomar várias naturezas, e nem sempre as combinações lineares são funções matemáticas suficientemente poderosas. Assim, é necessário recorrer a transformações para funções não lineares, nomeadamente a transformação para a função logística, computacionalmente conhecida como *sigmoid*, ou para outras transformações como a transformação Gaussiana ou a transformação da Tangente Hiperbólica (Kuhn & Johnson, 2013; Liboni, 2017).

Abordando agora o número de *hidden layers* e de neurónios, não existe uma regra pré-definida para se determinar o número ideal de cada. A combinação ótima depende do volume da dimensão de dados; do volume do ruído; e da complexidade do problema a resolver (Lantz, 2015), atribuindo ao *data scientist* a responsabilidade de realizar o *tuning* apropriado, testando várias combinações, para tentar obter a melhor performance preditiva do algoritmo, nunca olvidando a etapa da normalização das variáveis, de modo a otimizar a performance computacional e a performance estatística.

2.11 *Adaboost*

O *Adaboost* (ADB) foi introduzido por Freund e Schapire em 1995 (Freund *et al*., 1999), sendo o primeiro algoritmo baseado no paradigma *Boosting*. Tem aplicações em diversas áreas e é um dos algoritmos mais adotados em trabalhos de ML (Schapire, 2013).

O paradigma *Boosting* consiste em definir um *weigth* (peso) para cada observação, baseado na importância da mesma. Quanto maior o *weigth*, maior será a influência que

essa observação da amostra irá ter na árvore de decisão. No princípio, todas as observações têm o mesmo *weigth*, então à medida que as iterações do algoritmo avançam, novas árvores de decisão vão sendo construídas, e os respetivos pesos vão sendo atualizados, dando maior importância às observações incorretamente previstas, e menor importância às observações corretamente previstas (Pang & Gong, 2009).

Algoritmo *Adaboost* para problemas com duas classes

1. Marcar uma classe dos dados binários com o valor $+1$, e a outra classe com o valor -1
2. Definir a mesma importância de peso para cada amostra inicial: $\boldsymbol{weight} = (\mathbf{1}/\boldsymbol{n})$
3. Para cada $\mathbf{k} = \mathbf{1}$ até **K**, fazer:
4. Ajustar um $weak\ classifier$ usando as amostras $weighted$, e calcular o erro do modelo $\mathbf{k}^{\underline{o}}$: $(\mathbf{err_k})$
5. Calcular o $\mathbf{k}^{\underline{o}}$ valor de estágio com a fórmula: $\mathbf{ln}((\mathbf{1} - \mathbf{err_k})/\mathbf{err_k})$.
6. Atualizar o peso das observações da amostra, dando mais importância às amostras previstas incorretamente, e menos importância às amostras previstas corretamente
7. Fim
8. Calcular o classificador de previsão ***boosted*** para cada amostra, multiplicando o $\mathbf{k}^{\underline{o}}$ valor de estágio pelo valor de previsão do $\mathbf{k}^{\underline{o}}$ modelo, adicionando essas quantidades ao longo de **k**. Se a soma for positiva, então a observação é classificada com o valor $+1$ (positiva), caso contrário a observação será classificada com -1 (negativa).

Figura 10 – Algoritmo *Adaboost*
Fonte: (Kuhn & Johnson, 2013)

A ideia conceptual de funcionamento do ADB pode ser descrita da seguinte forma: o algoritmo produz *arrays* de classificadores fracos, e a cada iteração do algoritmo são escolhidos os classificadores mais fracos baseados nos pesos atribuídos na iteração anterior, com a finalidade de transformar esses classificadores fracos em classificadores fortes (Kuhn & Johnson, 2013).

À medida que o algoritmo avança, as amostras assinaladas pelo algoritmo como amostras incorretamente classificadas na iteração $\boldsymbol{k}$, irão receber mais importância na iteração $\boldsymbol{k}+1$. E por outro lado, as amostras assinaladas como corretamente classificadas na iteração $\boldsymbol{k}$, irão receber menos importância na iteração $\boldsymbol{k}+1$.

Assim, esta metodologia ao atribuir maior peso às observações que são mais difíceis de classificar corretamente, permite que o algoritmo se foque nessas observações até que as consiga prever corretamente, resultando assim numa maior assertividade das classificações (Fig.11).

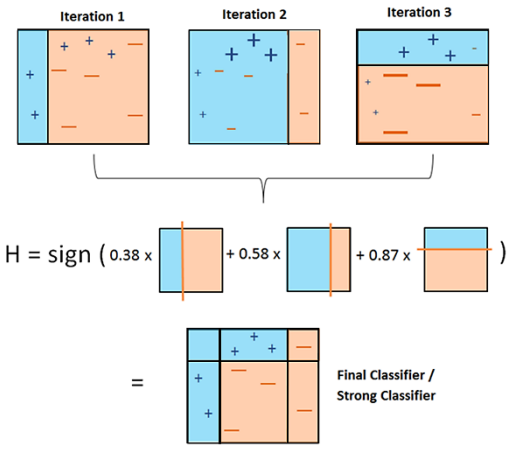


Figura 11 – *Adaboost* para problemas de classificação de duas classes
Fonte: (Dangeti, 2017)

2.12 C5.0

Árvores de decisão são o tipo de algoritmos de ML que são aplicados com maior frequência. A variedade de opções e de técnicas para este tipo de algoritmos é enorme, e cada vez existem mais derivações e alternativas entre eles. De entre os algoritmos mais conhecidos e assertivos, o algoritmo C5.0 encontra-se no topo das listas.

O cientista da computação J. Ross Quinlan desenvolveu o algoritmo C5.0 para melhorar o seu algoritmo anterior, denominado C4.5 (Lantz, 2015). De forma resumida, o C5.0 é mais rápido a executar que o C4.5; o uso da CPU e da memória RAM do computador que executa o algoritmo C5.0 mostram-se mais eficientes aquando comparado com o C4.5; e o C5.0 gera árvores de decisão mais pequenas que o C4.5.

Uma das vantagens em utilizar o algoritmo C5.0, deve-se a que este algoritmo permite obter menores taxas de erro para classes minoritárias e para observações mais difíceis de prever. Então, o C5.0 é também altamente robusto em identificar e ignorar as variáveis com menos contribuição, para a correta previsão das *labels* das observações (Pandya & Pandya, 2015).

É um algoritmo robusto independentemente do tamanho do *data set*, tem um *output* de fácil interpretação e consegue lidar com *missing values*. No entanto, por vezes tende a provocar *overfitting*, levando a que pequenas alterações na amostra de treino possam levar a bruscas alterações na lógica das árvores.

O algoritmo C5.0 está incluído no paradigma *Boosting* (Fig.12), e a implementação deste paradigma é uma das melhorias mais relevantes relativamente ao C4.5.

Algoritmo C5.0 *Boosting*

Notação:

$\mathbf{N}$ = dimensão da amostra de treino
$\mathbf{N_-}$ = número de amostras classificadas incorretamente
$\boldsymbol{w_k}$ = peso da amostra na iteração *boosting* **k**º
$\mathbf{S_+}$ = soma dos pesos de amostras corretamente classificadas
$\mathbf{S_-}$ = soma dos pesos de amostras incorretamente classificadas

Repetir:

De $\mathbf{1}$ até $\boldsymbol{N}$:

1. O algoritmo começa por determinar um ***midpoint*** entre a soma dos pesos das amostras incorretamente classificadas com metade da soma total dos pesos:

$$\boldsymbol{midpoint} = \frac{\mathbf{1}}{\mathbf{2}}\left[\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{S_-} + \boldsymbol{S_+}) - \boldsymbol{S_-}\right] = \frac{\mathbf{1}}{\mathbf{4}}(\boldsymbol{S_+} - \boldsymbol{S_-})$$

2. Depois, os pesos das amostras corretamente classificadas são ajustadas com a equação:

$$\boldsymbol{w_k} = \boldsymbol{w_{k-1}} \times \frac{\boldsymbol{S_+} - \boldsymbol{midpoint}}{\boldsymbol{S_+}}$$

3. E os pesos das amostras incorretamente classificadas, são atualizadas usando:

$$\boldsymbol{w_k} = \boldsymbol{w_{k-1}} \times \frac{\boldsymbol{midpoint}}{\boldsymbol{N_-}}$$

Nota: Este processo de atualização permite um largo salto positivo na gestão dos pesos das amostras incorretamente previstas. Quando uma amostra é corretamente prevista, a natureza multiplicativa da equação faz com que os pesos caiam mais devagar, até que essa mesma observação continue a ser corretamente prevista, melhorando assim a capacidade de previsão.

Figura 12 – Algoritmo C5.0
Fonte: (Kuhn & Johnson, 2013)

Por fim, é também importante salientar que o algoritmo C5.0 utiliza uma ferramenta teórica dos dados, denominada Entropia (Lantz, 2015), que é uma medida de incerteza que provoca o caos na organização dos dados. A forma ideal de construir modelos preditivos robustos e assertivos, é maximizando a variância da informação, em prol de minimizar o enviesamento dos resultados (Webber, 1977).

2.13 Métricas de avaliação

Para futura avaliação e comparação dos modelos estimados, são necessárias métricas de avaliação, para que seja possível selecionar o melhor modelo com base em várias medidas de desempenho. Neste projeto as métricas utilizadas serão: *Accuracy* (ACC); *F1 Value*; *Balanced Accuracy* (BACC); *Area Under the Curve* (AUC); *Classification Error*; *Precision* e *Recall* (Brodersen *et al*., 2010; Zheng, 2015). Seguem as respetivas fórmulas:

$$Precision = \frac{\#\ Verdadeiros\ Positivos}{\#\ Verdadeiros\ Positivos\ +\ \#\ Falsos\ Positivos} \qquad Recall = \frac{\#\ Verdadeiros\ Positivos}{\#\ Verdadeiros\ Positivos +\ \#\ Falsos\ Negativos}$$

$$Accuracy = \frac{\#\ de\ Previsões\ Corretas}{\#\ Observações\ Totais} \qquad F1\ Value = 2 \times \frac{Precision \times Recall}{Precision\ +\ Recall}$$

$$Balanced\ Accuracy = \frac{1}{2} \times \left(\frac{\#\ Verdadeiros\ Positivos}{\#\ Total\ dos\ Positivos} + \frac{\#\ Verdadeiros\ Negativos}{\#\ Total\ dos\ Negativos} \right)$$

$$Classification\ Error = \frac{\#\ de\ Previsões\ Incorretas}{\#\ Observações\ Totais}$$

A ACC mede o rácio entre o número de observações corretamente previstas e o número total de observações. A *Classification Error* mede a percentagem de observações incorretamente previstas, perante o número total das observações.

A *Precision* indica-nos a proporção de observações previstas como verdadeiros positivos, relativamente a todas as previsões classificadas como positivos (Boyd *et al*., 2013; Goutte & Gaussier, 2005). Esta métrica é relevante em cenários onde existe uma elevada importância relativamente ao peso dos falsos positivos. Por exemplo, quando um algoritmo de ML classifica um *email* como lixo eletrónico, mas na realidade esse *email* não é lixo eletrónico (falso positivo). Assim, quando a *Precision* de um modelo é pobre, corre-se o risco de o modelo preditivo classificar incorretamente as observações da classe menos importante de detetar, causando neste exemplo a perda de *emails* que não eram lixo eletrónico, devido às erradas previsões dos falsos positivos.

A métrica *Recall* indica-nos qual a proporção dos previstos como verdadeiros positivos, relativamente aos positivos reais (Boyd *et al*. 2013; Goutte & Gaussier, 2005). Esta métrica é relevante em cenários onde existe uma elevada importância relativamente

ao peso dos falsos negativos. Por exemplo, em cenários de deteção de fraudes, se um movimento fraudulento for classificado como não fraudulento (falso negativo), tal classificação irá resultar em consequências prejudiciais para a entidade lesada.

O *F1 Value* é a métrica que corresponde à média harmónica entre a *Precision* e a *Recall*. Esta métrica é eficaz perante a necessidade de encontrar um equilíbrio entre a *Precision* e a *Recall*, especialmente quando existe um cenário de dados desbalanceados (Goutte & Gaussier, 2005; Saito & Rehmsmeier, 2015; Zheng, 2015).

A BACC é uma métrica bastante relevante, e é aplicada em classificadores binários. Por vezes, as classes estão desbalanceadas, havendo muitas observações da classe negativa, e poucas da classe positiva. A BACC, permite obter a precisão média entre as classes, evitando cenários em que a ACC é bastante elevada e enviesada, uma vez que o modelo prevê corretamente a maioria das observações da classe negativa, e erra na maioria das escassas observações da classe positiva (Brodersen *et al*., 2010).

Finalmente, a métrica AUC revela a sensibilidade do modelo de previsão, sendo que quanto maior for a área abaixo da curva, melhor será o modelo preditivo (Zheng, 2015). Neste trabalho, a AUC é utilizada na análise no gráfico das Curvas de ROC (*Receiver Operating Characteristic*), e no gráfico das curvas *Precision – Recall*.

2.14 Síntese

Este capítulo engloba todas as ferramentas e algoritmos utilizados neste trabalho, que contribuíram para a estimação do melhor modelo, que será implementado no *software* de previsão de *leads*. O CV *K – Fold* foi utilizado inicialmente como uma ferramenta auxiliar para seleção das melhores variáveis, e a correlação entre variáveis foi calculada para garantir a ausência de colinearidade entre as mesmas. Os algoritmos SMOTE, ADASYN e TOMEK foram utilizados como ferramentas de aumento da dimensão de dados, e limpeza dos mesmos. Foi também possível identificar, através da revisão da literatura, os algoritmos mais adequados a utilizar neste contexto, nomeadamente: SVM; RF; NN; ADB e C5.0, que serão treinados e estimados recorrendo à repartição CV *K – Fold* com $K = 10$. E finalmente, foram selecionadas as métricas mais adequadas para realizar a avaliação e comparação entre os modelos, nomeadamente: ACC; *F1 Value*; BACC; AUC; *Classification Error*; *Precision* e *Recall*.

3. METODOLOGIA – TRABALHO EMPÍRICO

Tanto em projetos de ML como de *Data Mining*, é necessário seguir e respeitar metodologias *standard*, com passos sequenciais, de forma a garantir o cumprimento de etapas necessárias no processo de construção de um *software*. CRISP-DM; KDD; SEMMA; ASUM-DM; TDSP e POST-DM são as metodologias destacadas no que toca a processos de construção de sistemas computacionais. A metodologia CRISP-DM (*Cross-Industry Standard Process for Data Mining*) (Fig.13) é uma das mais adotadas em todo o mundo, sendo usualmente a mais competitiva entre as metodologias existentes (Costa & Aparicio, 2020; Freire & Omar, 2020).

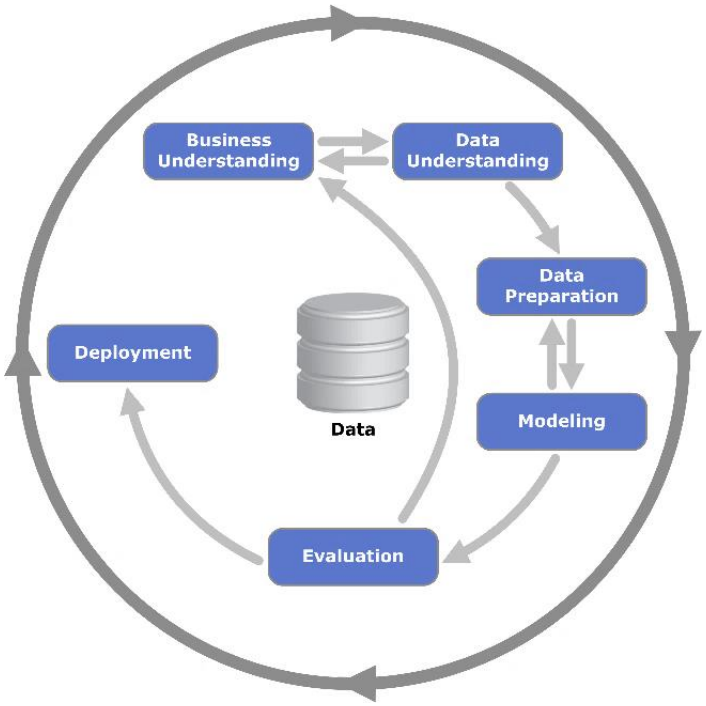


Figura 13 – Diagrama da metodologia CRISP-DM
Fonte: (Chapman *et al*., 2000)

As fases da metodologia CRISP-DM são as seguintes: *Business Understanding*; *Data Understanding*; *Data Preparation*; *Modeling*; *Evaluation* e *Deployment* (Wirth, 2000) (Fig.13). Neste projeto, o CRISP-DM foi a metodologia utilizada, e o seguimento das suas etapas enquadra-se perfeitamente na lógica de construção de um sistema computacional, capaz de melhorar a capacidade de decisão, de acordo com uma prévia compreensão do negócio e dos respetivos dados de uma empresa.

Começando com o ***Business Understanding***, foi realizada uma inicial compreensão do negócio e dos processos da empresa, e foi feito um levantamento do conhecimento teórico e prático dos funcionários da empresa, com base na experiência profissional dos mesmos. Esta fase é fundamental para que seja possível um correto alinhamento entre a realidade do negócio, e o objetivo que se pretende alcançar com a ferramenta digital.

Seguidamente, as fases de ***Data Understanding*** e ***Data Preparation*** são fundamentais para garantir que os dados recolhidos sejam pertinentes, e que consigam transmitir a realidade do negócio. Foi realizada uma recolha de dados, tendo por base os seguintes documentos: relatórios preliminares, atas finais e relatórios finais. O suporte de armazenamento de dados selecionado foi uma estrutura de *Web Application* com recurso a base de dados em *Microsoft SQL Server*. Nesta fase, os dados foram tratados, foi feita a sua normalização, a *feature selection*, a análise de correlações, bem como a aplicação de técnicas de aumento e limpeza de dados: SMOTE, ADASYN e TOMEK.

Modeling é a fase de maior exigência computacional e analítica. É a fase onde se realiza o treino, *tuning* e teste dos algoritmos de ML, que posteriormente permitem obter os modelos preditivos. A ferramenta selecionada para a realização desta etapa foi o *software* R com recurso ao *software* R *Studio*.

Posteriormente, vem a fase de ***Evaluation***, que tal como o nome indica, é a fase de avaliação e comparação dos resultados com base em métricas previamente definidas. Essas métricas são calculadas durante o processos de treino e de teste dos algoritmos, para que posteriormente seja possível efetuar a seleção do melhor modelo preditivo, recorrendo a métricas comparáveis, com maior assertividade e rigor.

E finalmente, a fase do ***Deployment***, ou seja, a implementação do modelo numa ferramenta tecnológica de previsão, em que seja possível inserir novos registos e prever a respetiva probabilidade de vitória, com base no melhor modelo preditivo encontrado. Dado que a empresa possuiu um *software* próprio, o ficheiro R do melhor modelo obtido foi adaptado, para possibilitar a sua utilização dentro da ferramenta criada com o *software* GENIO.

4. ETAPAS DE CONTRUÇÃO DO *SOFTWARE*

4.1 Recolha de dados e criação da BD

Foi realizado um levantamento dos documentos disponíveis, relativamente aos concursos públicos que a Quidgest ganhou e perdeu, nomeadamente: relatórios preliminares, atas finais e relatórios finais. Estes documentos serviram para criar uma base de pesquisa, e, através deles, foi possível obter informações sobre quais são alguns dos concorrentes da Quidgest, sobre as condições das propostas da Quidgest e dos seus concorrentes, sobre quais são alguns dos clientes, e ainda sobre quais os resultados das propostas dos concursos públicos analisados, tendo em conta os critérios de adjudicação.

Para aumentar a informação disponível relativamente a cada proposta, foi realizada uma recolha de dados na plataforma: https://amadeus.bvdinfo.com, que é uma base de dados com informações financeiras comparáveis de empresas públicas e privadas, em toda a Europa. Este levantamento consistiu em recolher dados financeiros sobre a Quidgest, bem como sobre os seus respetivos concorrentes e clientes. Foi também usada a plataforma: www.base.gov.pt, onde foram recolhidas informações sobre os clientes, como por exemplo: gasto anual em *software*; prazo médio de entrega dos projetos que solicitam; ou o preço médio anual dos concursos em que o cliente é adjudicante.

Foi necessário desenhar o modelo relacional da BD (Fig.14), para a criação da mesma. Para otimizar tempo e recursos, foram frequentadas formações de conhecimento do nível 1 e 2, em GENIO, para desenvolver a BD diretamente no próprio *software* GENIO.

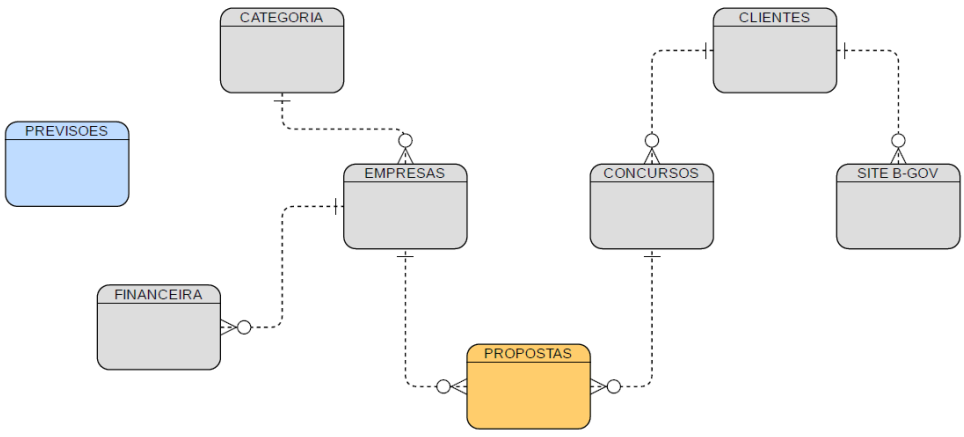


Figura 14 – Modelo relacional da base de dados do projeto
Fonte: Autor

Foi então desenvolvida em GENIO, uma plataforma que será utilizada no dia-a-dia pelos elementos do departamento de CN, que servirá como um auxiliar fundamental ao negócio, e ao mesmo tempo como uma ferramenta de gestão e armazenamento de dados.

Todos os dados recolhidos até este ponto foram importados para a plataforma, o que tornou possível o carregamento da informação para a BD "**MLQ**", (BD criada através do GENIO, onde foi armazenada toda a informação para o trabalho). Após toda a informação estar disponível nas tabelas da BD "**MLQ**", foram efetuadas *queries* SQL (*Structured Query Language*), para importar para o *software* R, toda a informação necessária.

A ferramenta utilizada para análise estatística, tratamento de dados, aplicação de algoritmos de ML, e outras operações auxiliares, foi o *software* R com recurso ao *software* R *Studio*. Através do R *Studio*, com o *Package* **RODB**, foi estabelecida uma ligação de ODBC (*Open Database Connectivity*) para permitir a conexão entre a BD "**MLQ**" e a consola do R *Studio*.

Após os dados estarem disponíveis no R *Studio*, iniciou-se o tratamento e limpeza dos mesmos, que é uma fase rigorosa e fundamental antes da aplicação de algoritmos de ML. Sem um tratamento de dados adequado, os resultados obtidos podem originar falsas interpretações, e tal consequência é perigosa e prejudicial em qualquer âmbito, seja a nível empresarial, médico ou tecnológico.

4.2 R – *Packages*

Para a realização deste projeto, foram utilizados os seguintes *Packages* do R: **adabag; base; C50; caret; cluster; corrplot; cowplot; dbscan; DMwR; e1071; factoextra; fpc; JOUSBoost; Metrics; mlbench; neuralnet; pROC; PRROC; randomForest; RODBC; UBL; unbalanced**.

Para utilizar *Packages* no *software* R, é necessário instalá-los e fazer o *call* das funcionalidades que cada *Package* oferece. Exemplificando:

```
> install.packages("DMwR")
> library(DMwR)
```

4.3 Pré processamento dos dados

Variáveis em estudo

Na tabela que se segue (Tab.1), estão listadas as variáveis que foram recolhidas para a realização deste trabalho. De seguida, serão abordadas as técnicas e ferramentas utilizadas no processo de seleção das variáveis a incluir na estimação dos modelos.

TABELA 1 – DESCRIÇÃO DAS VARIÁVEIS RECOLHIDAS

Nome da Variável	Descrição da Variável
GANHOU	Variável binária que toma o valor 1 caso o concurso tenha sido ganho
EM	Variável binária que toma o valor 1 caso o cliente seja uma empresa municipal
EPE	Variável binária que toma o valor 1 caso o cliente seja uma entidade pública empresarial
ES	Variável binária que toma o valor 1 caso o cliente seja uma entidade do ensino superior
AD	Variável binária que toma o valor 1 caso o cliente seja uma entidade de administração direta
AI	Variável binária que toma o valor 1 caso o cliente seja uma entidade de administração indireta
SNS	Variável binária que toma o valor 1 caso o cliente seja do serviço nacional de saúde
AUT	Variável binária que toma o valor 1 caso o cliente seja uma autarquia
OUTROS	Variável binária que toma o valor 1 caso o cliente não tenha uma entidade associada
GASTSOFT	Gasto em *software* até à data de extração, por parte do cliente, em euros
GASTO_ANUAL	Gasto total anual em todas as áreas por parte do cliente, até à data de extração, em euros
AVG_PRAZO_EN_ANUAL	Prazo de entrega médio dos projetos para o cliente, em dias
PRECO	Preço da proposta, em euros
PRAZO	Prazo de entrega do projeto, em dias
P_PRECO	Percentagem do preço da proposta nos critérios de adjudicação
P_TECNIC	Percentagem da robustez técnica nos critérios de adjudicação
P_MANTEC	Percentagem da qualidade e horas de manutenção técnica nos critérios de adjudicação
P_PRAZO	Percentagem do prazo da proposta nos critérios de adjudicação
P_OUTROS	Percentagem de outros critérios da proposta nos critérios de adjudicação
NREMPREG	Número de empregados da empresa que realiza a proposta
IDADE_FORN	Idade da empresa que realiza a proposta, em anos
VENDAS	Total de vendas anual da empresa que realiza a proposta, em euros

Fonte: Autor

Tratamento de *missing values* e de zeros desprovidos de sentido

Algumas variáveis, por indisponibilidade de dados em certas observações, tomam valor zero, e tal valor não pode ser considerado porque iria influenciar os resultados. Exemplificando: quando não existe informação sobre qual o prazo de entrega de uma proposta a um concurso, "zero" significa de facto zero dias para a entrega, e claramente esta informação é desprovida de sentido. Uma possível solução passa por atribuir a essa variável o valor "NA", mas alguns algoritmos descartam todo o registo se apenas uma variável dessa observação estiver como "NA", causando assim perda informação.

Uma boa alternativa passa por realizar a imputação desses valores com base na média, moda ou mediana, de todos os valores diferente de zero da respetiva variável. Se a distribuição dos valores das variáveis seguir um comportamento aproximado à distribuição normal, pode-se usar o valor da média, e em casos em que os dados apresentam algumas distorções, a solução mais indicada é usar a mediana (Hastie *et al*., 2009). Assim, o primeiro passo consistiu em atribuir a todos os zeros desprovidos de sentido, e a todos os *missing values*, o valor da mediana da variável correspondente.

Normalização dos dados

De seguida, foi realizada a normalização dos dados, que consiste na transformação dos dados reais, com medidas e amplitudes diferentes, num novo conjunto de dados com um intervalo padronizado. Este processo torna possível a comparação de variáveis que têm medidas diferentes (como por exemplo: o preço de um concurso que pode oscilar entre 30 mil a 500 mil euros, e o prazo de entrega que pode oscilar entre 50 a 500 dias), uma vez que todas as variáveis passam a estar compreendidas num intrevalo padronizado, usualmente entre [-1; 1], onde a unidade de medida deixa de ser considerada.

Os dados após serem normalizados passam a ter média zero ($\boldsymbol{\mu = 0}$), e desvio padrão igual a um ($\boldsymbol{\sigma = 1}$) (Burkov, 2019). Assim, o conjunto de dados foi normalizado usando a função `scale` do R, que se baseia na fórmula da distribuição normal padronizada:

$$\widehat{\boldsymbol{x}}^{(j)} = \frac{\boldsymbol{x}^{(j)} - \boldsymbol{\mu}^{(j)}}{\boldsymbol{\sigma}^{(j)}}$$

Nesta etapa, é de extrema importância guardar os resultados da média e do desvio padrão de todas as variáveis. Mais tarde, quando for necessário normalizar os novos dados a entrar no *software*, será com base nestes valores que a normalização se irá realizar.

***Cross Validation K – Fold* e correlação entre as variáveis**

Recorrendo aos *Packages* do *software* R: `mlbench` e `caret`, o CV *K – Fold* foi utilizado como um auxiliar, para determinar as variáveis mais importantes e o respetivo número ideal de variáveis a reter. É importante salientar que o algoritmo utilizado como modelo de avaliação em cada iteração do CV *K – Fold* foi o algoritmo RF (Choudhury & Gupta, 2018), usando uma repartição de $K = 10$ *Folds*.

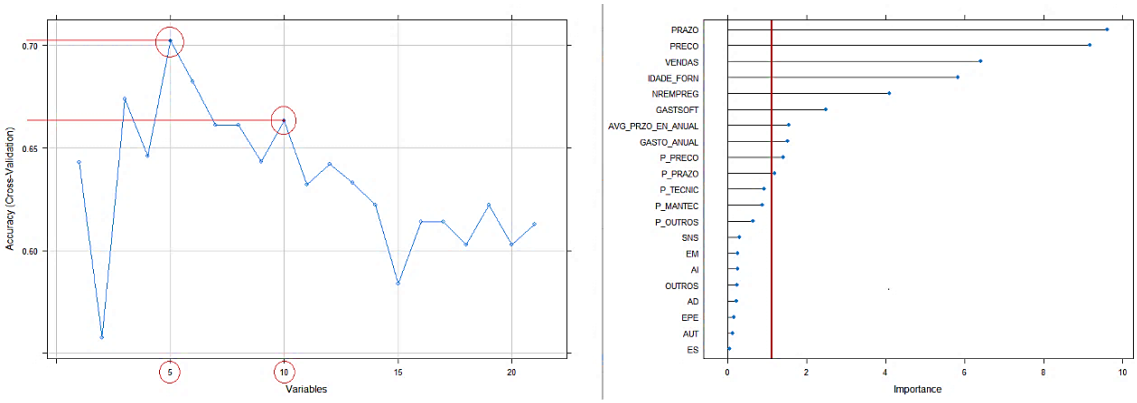


Figura 15 – Número ideal de variáveis e importância das mesmas
Fonte: Autor

O primeiro gráfico da figura 15 sugere que o número ideal de variáveis a reter deve ser 5. Porém o segundo ponto mais alto de ACC coincide com a seleção de 10 variáveis, e dado que a perda de ACC pela seleção de 10, em vez de 5, é inferior a 5 pontos percentuais (de ACC[5] = 0,702 para ACC[10] = 0,663), optou-se por se selecionar mais variáveis para se obterem modelos mais robustos. No segundo gráfico, da figura 15, está listada a importância que as variáveis têm no *data set*, e como tal, foram selecionadas as 10 variáveis com maior importância: **`PRAZO`**; **`PRECO`**; **`VENDAS`**; **`IDADE_FORN`**; **`NREMPREG`**; **`GASTSOFT`**; **`AVG_PRAZO_EN_ANUAL`**; **`GASTO_ANUAL`**; **`P_PRECO`**; **`P_PRAZO`**.

A correlação entre variáveis varia entre [-1; 1] sendo analisada através do *Pearson Correlation Coefficient* (Lantz, 2015). Colinearidade é o termo técnico para uma situação em que um conjunto de variáveis preditivas têm uma correlação substancialmente elevada entre si (Kuhn & Johnson, 2013), e variáveis preditivas altamente correlacionadas originam *overfitting* nos modelos. Isto é, essas variáveis com elevadas correlações tendem a explicar o mesmo fenómeno, e isso gera uma situação em que os algoritmos treinam demasiado bem o fenómeno em causa, gerando repercussões negativas na estimação dos modelos, e nas futuras previsões de novas observações.

De seguida, usando o *Package* **`corrplot`**, foi realizada a análise de correlações entre as 10 melhores variáveis selecionadas, não havendo casos alarmantes de colinearidade e, portanto, não foi removida qualquer uma das 10 variável selecionadas através do CV *K* – *Fold* com RF, com *K* = 10 *Folds* (Fig.16).

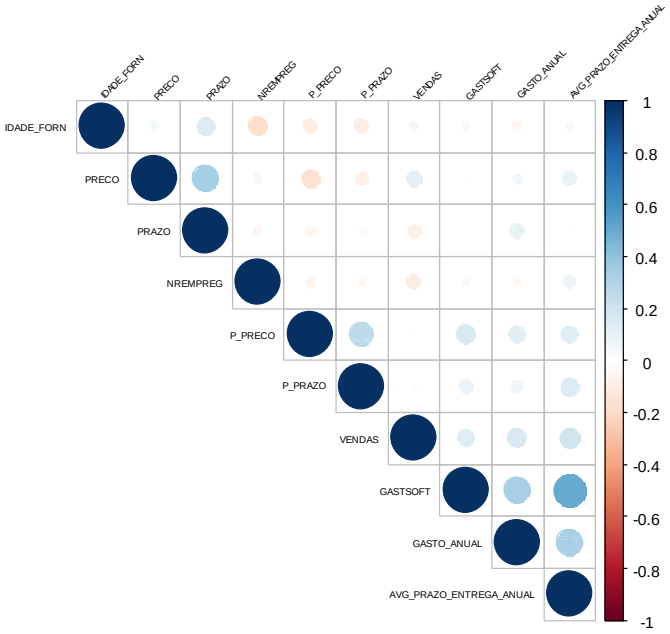


Figura 16 – Correlação entre as 10 melhores variáveis selecionadas
Fonte: Autor

4.4 SMOTE – ADASYN – TOMEK

O conjunto de dados tinha como dimensão total $n = 101$ observações. Este conjunto foi dividido em 70% para treino, e 30% para teste. Aos 70% para treino, foram aplicadas as técnicas de aumento de dados SMOTE e ADASYN, e posteriormente, em cada um dos cenários, foi aplicado o algoritmo TOMEK para limpeza de dados. Estas transformações originaram o seguinte cenário de dimensões das amostras de treino (Tab. 2):

TABELA 2 – DIMENSÃO DAS AMOSTRAS DE TREINO FINAIS

	Dados Artificiais	**Dados Artificiais com TOMEK**
SMOTE	$\boldsymbol{n = 323}$	$\boldsymbol{n = 317}$
ADASYN	$\boldsymbol{n = 83}$	$\boldsymbol{n = 75}$

Fonte: Autor

Na figura 17, é demonstrada graficamente a dispersão dos dados originais e a dispersão dos dados artificiais gerados com SMOTE (sem a aplicação de TOMEK). Observa-se que o *data set* contém todas as observações dos dados originais, e contém

também as observações geradas artificialmente. É percetível que os dados gerados artificialmente seguem o padrão de comportamento dos dados originais, o que indica uma apropriada criação do conjunto de dados artificiais.

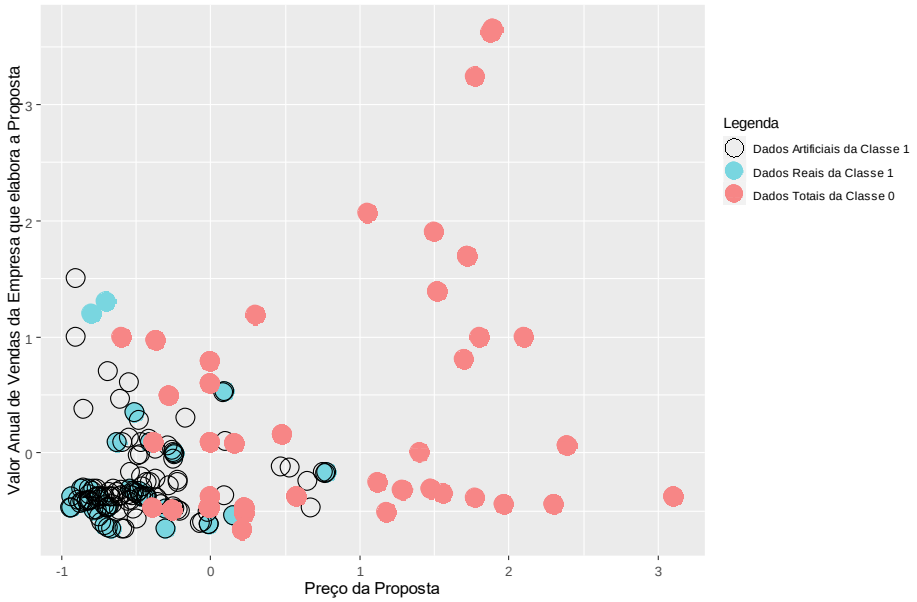


Figura 17 – Dados artificiais gerados com SMOTE
Fonte: Autor

4.5 Aplicação dos algoritmos de ML

Após os dados terem sido recolhidos, armazenados e tratados (normalização de dados; seleção das melhores variáveis; análise de correlações; SMOTE; ADASYN e TOMEK), o *data set* está então apto a ser utilizado no processo de treino e teste dos algoritmos. O conjunto de dados a ser usado como conjunto de treino tem 317 observações, incluindo as observações artificiais geradas com SMOTE – TOMEK. E como conjunto de teste final, têm-se 31 observações originais, o que corresponde aos 30% da divisão inicial.

A estimação de cada modelo foi acompanhada pelo CV *K – Fold* com *K* = 10, de forma a evitar problemas de *overfitting*. Cada algoritmo tem parâmetros diferentes, logo, foi realizado o *tuning* apropriado durante o treino de cada algoritmo por forma a aumentar a sua capacidade preditiva. Obtiveram-se então vários modelos por cada algoritmo, e o melhor modelo de cada um foi guardado num ficheiro `rds`. Este passo é de extrema importância, uma vez que permite gravar em ficheiros individuais, as configurações estatísticas e matriciais dos respetivos modelos.

Todos os modelos seguem uma estrutura lógica de aplicação, acompanhada pelo cálculo das métricas de avaliação. Como o melhor modelo obtido foi através do algoritmo SVM, segue a estrutura do código em R que permitiu obter o modelo (Fig.18):

```r
##---------------------------------------
# SVM - Using Smote + Tomek Clean Data
##---------------------------------------

# teste  = dataset com dados de teste  (n=31)
# smoted = dataset com dados de treino (n=317)

# Aplicação do Modelo SVM com CV 10-Folds
folds = createFolds(smoted$GANHOU, k = 10)

cv_kfolds = lapply(folds,function(x) {

    train_fold = smoted[-x, ]
    test_fold = smoted[x, ]

    svm_model <- svm(GANHOU~., data = train_fold,
              scale = FALSE, kernel = 'polynomial',
              cost = 1, shrinking = TRUE,
              probability = TRUE)
```

Ativação do Método de Platt através de probability = TRUE

```r
  # Previsão
  predSVM = predict(svm_model,  test_fold)

  # Obter dados para calcular métricas
  performSVM = table(predSVM, test_fold$GANHOU)
```

SVM (K = 10 Folds):

```r
  # Nota: TP estão em [2,2] e os TN em [1,1]
  TP = performSVM[2,2]; FP = performSVM[1,2]
  FN = performSVM[2,1]; TN = performSVM[1,1]

  # Curva De ROC
  aucSVM = roc(test_fold, predSVM,
               levels = c(0, 1), direction = "<")

  # Calcular as Métricas
  ACC = accuracy(test_fold$GANHOU,predSVM)
  CE = ce(test_fold$GANHOU,predSVM)
  P = (TP/(TP+FP))
  R = (TP/(TP+FN))
  F1 = (2*P*R)/(P+R)
  AUC = (aucSVM$auc)
  BACC = ((TP/(TP+FP))+(TN/(FN+TN)))/2

  metrics = list(ACC, CE, P, R, F1, AUC, BACC)

  return(metrics)
})
```

Média das Métricas:

```r
# Por fim, foram calculadas as médias de cada
# métrica presentes em cada um dos 10 Folds

# Exemplo para média da ACC:

acc_k1 = cv_kfolds[["Fold01"]]$ACC
acc_k2 = cv_kfolds[["Fold02"]]$ACC
# ... igual até acc_k10

all_acc = c(acc_k1, acc_k2, acc_k3, acc_k4,
            acc_k5, acc_k6, acc_k7, acc_k8,
            acc_k9, acc_k10)

ACC_SVM = mean(all_acc)
```

Figura 18 – Código R – Aplicação do algoritmo SVM em R
Fonte: Autor

Após todos os algoritmos terem sido treinados, o melhor modelo de cada algoritmo foi gravado num ficheiro, usando o *Package* **`base`** do R. Assim, após a avaliação e seleção do melhor entre os cinco modelos, o modelo vencedor já estará pronto a ser importado, para ser então utilizado na previsão de novos dados.

Finalmente, os cinco melhores modelos foram testados, usando o conjunto de teste inicial com apenas dados reais (n=31). Estes resultados estão presentes tabela 3 em anexo, e foi com base nestes resultados que se procedeu à seleção do modelo vencedor.

4.6 Comparação de modelos e discussão de resultados

Para realizar a avaliação das métricas de performance dos modelos, tanto o gráfico das curvas de ROC como o gráfico de *Precision - Recall* são boas ferramentas para dados binários. Contudo, o gráfico de *Precision - Recall* tem vindo a ser progressivamente mais utilizado na avaliação de performance de algoritmos de ML (Goutte & Gaussier, 2005; Saito & Rehmsmeier, 2015). Isto deve-se à sua maior versatilidade de funcionamento tanto com dados balanceados, como com dados desbalanceados, e devido à combinação das duas métricas em questão, que permitem alcançar uma melhor e mais assertiva comparação e interpretação da performance dos modelos, recorrendo aos resultados alcançados, que são organizados numa matriz de confusão (Fig.19).

	Observação classificada como Positiva	**Observação classificada como Negativa**
Classe Positiva	Verdadeiro Positivo (**TP**)	Falso Negativo (**FN**)
Classe Negativa	Falso Positivo (**FP**)	Verdadeiro Negativo (**TN**)

Figura 19 – Matriz de confusão
Fonte: (Sokolova & Lapalme, 2009)

Através da análise dos gráficos das curvas de ROC e de *Precision - Recall* (Fig.20), e seguindo o princípio de que quanto maior a área abaixo das curvas, melhor a performance dos modelos (Kuhn & Johnson, 2013), é notória a qualidade do modelo obtido com SVM face aos restantes algoritmos.

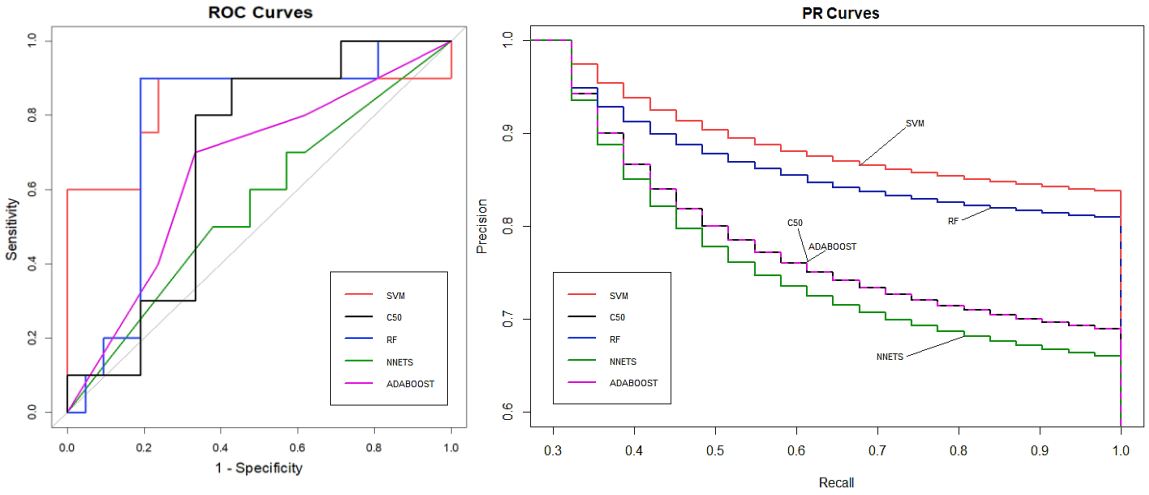


Figura 20 – Curvas de ROC e curvas *Precision – Recall*
Fonte: Autor

No gráfico das curvas de ROC (Fig.20), a AUC do SVM e do RF apresentam valores próximos, mostrando a boa performance destes dois modelos. Porém, quando analisado o gráfico das curvas de *Precision - Recall*, é o modelo SVM que domina, apresentando assim a melhor capacidade preditiva e o melhor equilíbrio entre as duas métricas em análise. Na tabela 3, em anexo, constam todos os valores das métricas calculados aquando a estimação dos melhores modelos, sendo novamente notório que o SVM foi o algoritmo que alcançou a melhor performance em praticamente todas as métricas, nomeadamente: ACC, *F1 Value*, BACC, *Classification Error* e no equilibro entre a *Precision* e a *Recall*.

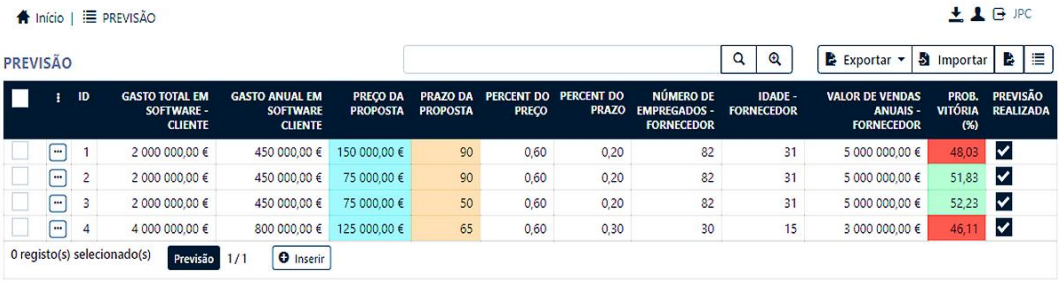
Em outros estudos sobre a mesma temática (Bohanec *et al*., 2017), foram utilizados algoritmos como: SVM; NN e RF, mas sem recorrer a técnicas de aumento de dados. Nesse estudo, o melhor modelo obtido foi o RF, e se observarmos qual é o melhor modelo obtido neste TFM no que toca apenas aos dados originais (Tab. 3 em anexo), é também o algoritmo RF que oferece o melhor modelo, reforçando a boa capacidade preditiva do RF no que toca à temática de previsão do sucesso de vendas.

Neste TFM, através da utilização de técnicas de aumento de dados, foi possível obter modelos com melhores performances, aumentando assim a assertividade das previsões. Conclui-se também que apesar do ADASYN ser uma derivação do SMOTE, os dados artificiais gerados com SMOTE mostraram ser muito mais eficazes em todos os algoritmos, à exceção das NN, que foram mais eficazes aquando a utilização dos dados artificiais gerados pelo ADASYN (Tab. 3 em anexo).

4.7 A ferramenta final

Após ter sido identificado o melhor modelo, este foi guardado dentro de uma pasta no servidor da empresa para possibilitar a sua utilização sempre que necessário. Foi também necessário configurar os *R – Services* do *Microsoft SQL Server*, para permitir a execução de blocos de código R na consola de *queries* do SQL. Foi então criado um *R – Script* (Fig.22 em anexo) que é executado diretamente no SQL, para permitir que sempre que um elemento do departamento de CN queira prever a probabilidade de sucesso de uma determinada proposta a um concurso público, esse *R – Script* irá realizar as seguintes operações para retornar a respetiva probabilidade de vitória:

1. Inicialmente é feito o *call* dos *Pakages* "`RODBC`" e "`e1071`", e é realizada a conexão à base de dados com as credenciais de acesso: *username* e *password*;
2. A variável "data" recebe o *output* de uma *query* SQL, onde serão guardados todos os novos registos das propostas a serem previstas;
3. Os novos dados são normalizados com os respetivos valores da média e do desvio padrão, que foram utilizados na produção inicial do modelo (sendo este passo fundamental para garantir a coerência das estimações);
4. É feito o *upload* do modelo (SVM), e aplica-se o modelo aos novos dados, em que o *output* será a probabilidade de vitória de cada uma das novas observações;
5. É feita uma atualização na tabela, para que o campo "previsto" tome o valor "1", indicando assim que os registos selecionados já foram previstos;
6. E finalmente, os resultados finais e resumidos serão apresentados na plataforma de *Web Application*, como consta na figura 21.


Início | PREVISÃO

JPC

PREVISÃO

Exportar | Importar

ID	GASTO TOTAL EM SOFTWARE - CLIENTE	GASTO ANUAL EM SOFTWARE CLIENTE	PREÇO DA PROPOSTA	PRAZO DA PROPOSTA	PERCENT DO PREÇO	PERCENT DO PRAZO	NÚMERO DE EMPREGADOS - FORNECEDOR	IDADE - FORNECEDOR	VALOR DE VENDAS ANUAIS - FORNECEDOR	PROB. VITÓRIA (%)	PREVISÃO REALIZADA
1	2 000 000,00 €	450 000,00 €	150 000,00 €	90	0,60	0,20	82	31	5 000 000,00 €	48,03	☑
2	2 000 000,00 €	450 000,00 €	75 000,00 €	90	0,60	0,20	82	31	5 000 000,00 €	51,83	☑
3	2 000 000,00 €	450 000,00 €	75 000,00 €	50	0,60	0,20	82	31	5 000 000,00 €	52,23	☑
4	4 000 000,00 €	800 000,00 €	125 000,00 €	65	0,60	0,30	30	15	3 000 000,00 €	46,11	☑

0 registo(s) selecionado(s) Previsão 1 / 1 Inserir

Figura 21 – Ferramenta final para previsão do sucesso de *leads*
Fonte: Autor

O utilizador insere então os dados na ferramenta (Fig.21) e quando tiver inserido todas as propostas que deseja prever, irá pressionar o botão "Previsão" que executa o *R – Script* (Fig.22 em anexo). O *software* retornará os valores da coluna "Prob. Vitória (%)", que corresponde à probabilidade de vitória de cada proposta com os respetivos valores inseridos. Esta ferramenta servirá assim como um auxílio estatístico à tomada de decisão.

5. CONCLUSÃO E TRABALHOS FUTUROS

O universo da IA tem vindo a demonstrar um crescimento célere, sendo cada vez mais frequente a utilização de algoritmos de ML, nos mais variados cenários. É um novo mundo, abundante em dados, e a IA está a tornar-se um hóspede assíduo nas nossas vidas.

Este TFM teve como objetivo principal, a elaboração de uma proposta de *software* capaz de prever o sucesso das *leads* da empresa, recorrendo a ML. Este objetivo foi atingido na sua plenitude, tanto a nível da revisão literária, como a nível do desenvolvimento do *software*, concretizando os seguintes pontos:

Identificação do contexto empresarial e dos dados a utilizar: Através dos relatórios preliminares, atas finais e relatórios finais, foram recolhidos os dados pertinentes ao estudo, que foram ainda complementados com dados extra retirados das plataformas: https://amadeus.bvdinfo.com e www.base.gov.pt.

Identificar as soluções mais adequadas de tratamento, aumento e limpeza de dados: As técnicas e algoritmos selecionados para o tratamento, aumento e limpeza de dados foram: CV *K – Fold*; testes de correlações; SMOTE; ADASYN e TOMEK.

Identificar os melhores algoritmos de ML que permitam prever com assertividade o sucesso das *leads*, avaliando e comparando a qualidade dos mesmos:

Foi realizada uma extensa pesquisa para a seleção dos algoritmos de ML mais adequados ao problema, nomeadamente: SVM; RF; NN; ADB e C5.0. Após o treino e teste dos algoritmos, foi realizada a avaliação e seleção dos modelos com base em métricas apropriadas, tendo sido o SVM a alcançar o modelo vencedor.

Desenvolver um *software* que incorpora a solução proposta: Por fim, procedeu-se ao *deployment* do melhor modelo através da integração entre GENIO, SQL e R, com recurso a *scripts* capazes de garantir o funcionamento do *software* de previsão.

Com este *software*, a empresa possui agora uma ferramenta preditiva que servirá como auxílio à tomada de decisão. Esta ferramenta veio oferecer também uma boa solução de armazenamento e gestão de dados. A cada concurso a que a empresa se candidata, novos dados são recebidos, e ao invés do armazenamento dessa informação passar por se guardar os documentos em pastas dentro do servidor da empresa, se a informação for registada neste *software*, será mais fácil consultar o histórico das

propostas, e, ao mesmo tempo, irá estar-se a contribuir para o aumento de dados na aplicação, o que futuramente irá possibilitar obter modelos mais assertivos e complexos.

A maior limitação deste trabalho foi sem dúvida a pouca quantidade de dados disponíveis. A empresa não tinha qualquer estrutura de dados já pronta, e a decisão de quais dados usar, de como os usar, e de como os armazenar, foram alguns dos maiores desafios. Foi necessário agregar informação de outras fontes para permitir uma maior complexidade nas análises, mas tais informações permitiram um aumento de mais variáveis, e não de mais observações. Assim, houve a necessidade de recorrer a técnicas de aumento de dados para balancear as classes, e para melhorar a capacidade preditiva dos modelos. No entanto, o ideal será que no futuro sejam aplicados novamente todos os modelos, e até outros novos, mas num cenário em que apenas dados reais sejam utilizados.

Para trabalhos futuros, será interessante aplicar técnicas de aprendizagem não supervisionada, nomeadamente: Análise em Componentes Principais (PCA) e Análise Fatorial (FA), e posteriormente, proceder-se à aplicação de técnicas de *clustering*, tais como: *Affinity Propagation*, *K-Means* ou DBSCAN. Apesar de não ser abordado neste trabalho, estas técnicas foram aplicadas a título de curiosidade e obtiveram-se resultados promissores, capazes de melhorar a capacidade de interpretação dos dados. Outro aspeto relevante a considerar é o Auto – ML. Este conceito possibilita treinar e testar automaticamente os algoritmos de ML, à medida que novos dados entram no *software*, permitindo assim uma aprendizagem continua por partes dos algoritmos. Isto resulta numa melhoria da qualidade das previsões, sendo necessário haver uma monitorização dos modelos preditivos, para assegurar que os mesmos evoluem da forma desejada.

O *software* R com recurso ao R *Studio* foi a principal ferramenta deste trabalho, juntamente com os *Packages* disponíveis para esta ferramenta. Existem outras ferramentas disponíveis para este tipo de trabalhos, tais como: *Phyton*, que tem *Packages* bastante conhecidos como o *TensorFlow*; *Keras* e *PyTorch*; ou o *Rapid Miner*, conhecido pela sua interface *user-friendly* de "*drag and drop*", que é útil para quem não tem conhecimentos em programação, mas deseja produzir modelos de ML. Se por outro lado houver interesse em soluções pagas, empresas dedicadas ao avanço tecnológico como a Microsoft; Amazon ou a Google, têm ferramentas exclusivamente dedicadas a ML, tais como: *Azure Machine Learning*; *Amazon SageMaker* e o *Google Cloud AI Platform*.

REFERÊNCIAS BIBLIOGRÁFICAS

Aditsania, A., Adiwijaya, K., Saonard, A. (2017). Handling imbalanced data in churn prediction using ADASYN and backpropagation algorithm – IEEE Conference Publication. doi: 10.1109/ICSITech.2017.8257170

Alencar, R. (2017). Resampling strategies for imbalanced datasets. Disponível em: https://kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets (Acedido a: 25-09-2020).

Aridas, C., & Lemaitre, G. (2018). An illustration of the Adaptive Synthetic Sampling Approach for Imbalanced Learning ADASYN method. Disponível em: http://glemaitre.github.io/imbalanced-learn/auto_examples/over-sampling/plot_adasyn.html (Acedido a: 20-03-2020).

Barstugan, M., Ozkaya, U., & Ozturk, S. (2020). Coronavirus (COVID-19) Classification using CT Images by Machine Learning Methods. [Cs, Eess, Stat]. Disponível em: http://arxiv.org/abs/2003.09424

Batista, G. E. A. P. A., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, 6(1), (pp. 20–29). doi: 10.1145/1007730.1007735

Berrar, D. (2018). Cross-Validation. In S. Ranganathan, M. Gribskov, K. Nakai, & C. Schönbach (Eds.), Encyclopedia of Bioinformatics and Computational Biology (pp. 542–545). Academic Press. doi: 10.1016/B978-0-12-809633-8.20349-X

Bohanec, M., Borštnar, M. K., & Šikonja, M. R. (2015). Integration of machine learning insights into organizational learning: A case of B2B sales forecasting. BLED 2015 Proceedings. https://aisel.aisnet.org/bled2015/33

Bohanec, M., Kljajić Borštnar, M., & Robnik-Šikonja, M. (2017). Explaining machine learning models in sales predictions. Expert Systems with Applications, 71, (pp. 416–428). doi: 10.1016/j.eswa.2016.11.010

Boyd, K., Eng, K. H., & Page, C. D. (2013). Area under the Precision-Recall Curve: Point Estimates and Confidence Intervals. In C. Salinesi, M. C. Norrie, & Ó. Pastor (Eds.), Advanced Information Systems Engineering, 7908, (pp. 451–466). Springer Berlin Heidelberg. doi: 10.1007/978-3-642-40994-3_29

Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The Balanced Accuracy and Its Posterior Distribution. 2010 20th International Conference on Pattern Recognition, (pp. 3121–3124). doi: 10.1109/ICPR.2010.764

Burkov, A. (2019). The Hundred-Page Machine Learning Book. Andriy Burkov.

Chapman ,P., Clinton, J., Kerber, R., Khabaza, Th., Reinartz, Th., Shearer, C., & Wirth, R. (2000) CRISP-DM 1.0 - Step-by-step data mining guide, SPSS, Inc.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16, (pp. 321–357). doi: 10.1613/jair.953

Choudhury, A., & Gupta, D. (2018). A Survey on Medical Diagnosis of Diabetes Using Machine Learning Techniques. Em J. Kalita V. E. Balas, S. Borah, & R. Pradhan (Eds.), Recent Developments in Machine Learning and Data Analytics (Vol. 740, pp. 67–78). Springer Publishing. doi: 10.1007/978-981-13-1280-9_6

Costa, C. J., & Aparicio, J. T. (2020). POST-DS: A Methodology to Boost Data Science. 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), (pp. 1–6). doi: 10.23919/CISTI49556.2020.9140932

Cruz-Jesus, F., Castelli, M., Oliveira, T., Mendes, R., Nunes, C., Sa-Velho, M., & Rosa-Louro, A. (2020). Using artificial intelligence methods to assess academic achievement in public high schools of a European Union country. Heliyon. 6(6). doi: 10.1016/j.heliyon.2020.e04081

Custódio, J. P. G., Costa, C. J., & Carvalho, J. P. (2020). Success Prediction of Leads – A Machine Learning Approach. 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), (pp. 1–6). doi: 10.23919/CISTI49556.2020.9141002

Dangeti, P. (2017). Statistics for Machine Learning: Techniques for exploring supervised, unsupervised, and reinforcement learning models with Python and R. Packt Publishing. Mumbai, Índia.

Devi, D., Biswas, S. kr., & Purkayastha, B. (2017). Redundancy-driven modified Tomek-link based undersampling: A solution to class imbalance. Pattern Recognition Letters, 93, (pp. 3–12). doi: 10.1016/j.patrec.2016.10.006

Freire, V. T., & Omar, N. (2020). Comparação entre métodos, metodologias e frameworks para construção de sistemas computacionais analíticos-cognitivos / Comparison between methods, methodologies, and frameworks for the construction of analytical-cognitive computational systems. Brazilian Journal of Development, 6(5), (pp. 31887–31904). doi: 10.34117/bjdv6n5-586

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society for Artificial Intelligence, 14, (pp. 771–780).

Genuer, R., Poggi, J.-M., Tuleau-Malot, C., & Villa-Vialaneix, N. (2017). Random forests for big data. Big Data Research, 9, (pp. 28–46). doi: 10.1016/j.bdr.2017.07.003

Goldberg, Y. (2017). Neural Network Methods in Natural Language Processing (G. Hirst, Ed.). Morgan & Claypool Publishers.
doi: 10.2200/S00762ED1V01Y201703HLT037

Goutte, C., & Gaussier, E. (2005). A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. Em David E. Losada, Juan M. Fernández-Luna (Eds.), Advances in Information Retrieval (pp. 345–359). Springer Publishing. doi: 10.1007/978-3-540-31865-1_25

Haibo He, Yang Bai, Garcia, E. A., & Shutao Li. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), (pp. 1322–1328). doi: 10.1109/IJCNN.2008.4633969

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, (2ª Edição). Springer, New York, NY. doi:10.1007/978-0-387-84858-7

Hendrycks, D., Mazeika, M., Kadavath, S., & Song, D. (2019). Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty. [Cs, Stat]. http://arxiv.org/abs/1906.12340

Hu, F., & Li, H. (2013). A Novel Boundary Oversampling Algorithm Based on Neighborhood Rough Set Model: NRSBoundary-SMOTE. Mathematical Problems in Engineering, 2013, (pp. 1–10). doi: 10.1155/2013/694809

Joshi, A. V. (2020). Machine Learning and Artificial Intelligence. Springer International Publishing. doi: 10.1007/978-3-030-26622-6

Kaza, N. (2018). Machine Learning for Remote Sensing. Nikhil Kaza. Disponível em: https://nkaza.github.io/post/machine-learning-for-remote-sensing/ (Acedido a: 16-08-2020).

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling (2ª Edição). Springer. doi: 10.1007/978-1-4614-6849-3

Lantz, B. (2013). Machine Learning with R: Learn How to Use R to Apply Powerful Machine Learning Methods and Gain and Insight Into Real-world Applications. Packt Publishing.

Lantz, B. (2015). Machine Learning with R - Second Edition: Expert techniques for predictive modeling to solve all your data analysis problems (2ª Edição). Packt Publishing.

Leijnen, S., & Veen, F. van. (2020). The Neural Network Zoo. Proceedings, 47(1), 9. doi: 10.3390/proceedings2020047009

Liboni, L. H. B. (2017). Diagnóstico de falhas em motores de indução trifásicos baseado em decomposição em componentes ortogonais e aprendizagem de máquinas (Tese de Doutoramento, Universidade de São Paulo, São Paulo, Brasil). doi: 10.11606/T.18.2017.tde-30062017-091155

Marr, B. (2019). Artificial Intelligence in Practice: How 50 Successful Companies Used AI and Machine Learning to Solve Problems (1ª Edição). John Wiley & Sons.

Mitchell, T. M. (1997). Machine Learning (1ª Edição). McGraw-Hill Education.

Mortensen, S., Christison, M., Li, B., Zhu, A., & Venkatesan, R. (2019). Predicting and Defining B2B Sales Success with Machine Learning. 2019 Systems and Information Engineering Design Symposium (SIEDS), (pp. 1–5). doi: 10.1109/SIEDS.2019.8735638

Pandya, R., & Pandya, J. (2015). C5.0 Algorithm to Improved Decision Tree with Feature Selection and Reduced Error Pruning. doi: 10.5120/20639-3318

Pang, S., & Gong, J. (2009). C5.0 Classification Algorithm and Application on Individual Credit Evaluation of Banks. Systems Engineering - Theory & Practice, 29 (12), (pp. 94–104). doi: 10.1016/S1874-8651(10)60092-0

Patel H., Parikh S., Patel A. & Parikh A. (2018). An Application of Ensemble Random Forest Classifier for Detecting Financial Statement Manipulation of Indian Listed Companies. Em J. Kalita V. E. Balas, S. Borah, & R. Pradhan (Eds.), Recent Developments in Machine Learning and Data Analytics (Vol. 740, pp. 349–360). Springer Publishing. doi: 10.1007/978-981-13-1280-9_33

Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. Em A. Smola, P. Bartlett, B. Schölkopf, & D. Schuurmans (Eds.), Advances in large margin classifiers. Cambridge: MIT Press.

Quidgest. (2019). Sobre a Quidgest—Breve História da Empresa. Disponível em: https://www.quidgest.pt/boasvindasPT.asp?LT=PTG (Acedido a: 7-09-2020).

Quidgest. (2020). Plataforma Genio: Software do Futuro. Disponível em: https://quidgest.com/pt-br/sobre-a-quidgest/genio/ (Acedido a: 7-09-2020).

Saito, T., & Rehmsmeier, M. (2015). The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLOS ONE, 10(3). doi: 10.1371/journal.pone.0118432

Schapire, R. (2013). Explaining AdaBoost. Princeton University, USA. Em Bernhard Schölkopf, Zhiyuan Luo, Vladimir Vovk (Eds.), Empirical Inference (pp. 37–52). Springer, Berlin, Heidelberg. doi: 10.1007/978-3-642-41136-6_5

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Information Processing & Management, 45(4), (pp. 427–437). doi: 10.1016/j.ipm.2009.03.002

Suetake, M. (2012). Sistemas inteligentes para monitoramento e diagnósticos de falhas em motores de indução trifásicos (Tese de Doutoramento, Universidade de São Paulo, São Paulo, Brasil). doi: 10.11606/T.18.2012.tde-26062012-164520

Tahmasebi, P., Kamrava, S., Bai, T., & Sahimi, M. (2020). Machine Learning in Geo- and Environmental Sciences: From Small to Large Scale. Advances in Water Resources, 142. doi: 10.1016/j.advwatres.2020.103619

Thai-Nghe, N., & Schmidt-Thieme, L. (2010). Learning Optimal Threshold on Resampling Data to Deal with Class Imbalance. https://api.semanticscholar.org/CorpusID:16886485

Torgo, L. (2016). Data Mining with R: Learning with Case Studies (2ª Edição). Chapman and Hall/CRC. New York. doi. doi.org/10.1201/9781315399102

US News. (2017). Elon Musk: Artificial Intelligence Is Society's 'Biggest Risk'. US News & World Report. Disponível em: https://www.usnews.com/news/national-news/articles/2017-07-16/elon-musk-artificial-intelligence-is-the-biggest-risk-that-we-face-as-a-civilization (Acedido a: 15-09-2020).

Vapnik, V. N. (2000). The Nature of Statistical Learning Theory (2ª Edição) 2000. Springer New York. doi: doi.org/10.1007/978-1-4757-3264-1

Webber, M. J. (1977). Pedagogy Again: What Is Entropy?. Annals of the Association of American Geographers, 67 (2), (pp. 254–266). doi: 10.1111/j.1467-8306.1977.tb01138.x

Wirth, R. (2000). CRISP-DM: Towards a standard process model for data mining. Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, (pp. 29–39). doi: 10.1.1.198.5133

Zheng, A. (2015). Evaluating machine learning models: A beginner's guide to key concepts and pitfalls. O'Reilly Media, Inc.

ANEXOS

TABELA 3 – SUMÁRIO COM AS MÉTRICAS DE AVALIAÇÃO

Algoritmo	Amostra	Accuracy	F1 Value	Balanced Acc	AUC - ROC	Class. Error	Precision	Recall
SVM	*SMOTE*	0,871*	0,750*	0,920*	0,800	0,129*	0,600	1,000
	ADASYN	0,484	0,529	0,616	0,593	0,516	0,900	0,375
	Set de Treino	0,774	0,462	0,875	0,650	0,226	0,300	1,000
RF	*SMOTE*	0,774	0,720	0,769	0,807*	0,226	0,900	0,600
	ADASYN	0,677	0,545	0,645	0,657	0,323	0,600	0,500
	Set de Treino	0,806	0,700	0,779	0,779	0,194	0,700	0,700
C5.0	*SMOTE*	0,677	0,583	0,662	0,683	0,323	0,700	0,500
	ADASYN	0,323	0,488	0,500	0,500	0,677	1,000	0,323
	Set de Treino	0,677	0,000	0,500	0,500	0,323	0,000	1,000
NN	*SMOTE*	0,548	0,462	0,554	0,562	0,452	0,600	0,375
	ADASYN	0,645	0,476	0,602	0,607	0,355	0,500	0,455
	Set de Treino	0,548	0,462	0,554	0,562	0,452	0,600	0,375
ADB	*SMOTE*	0,677	0,583	0,662	0,683	0,323	0,700	0,500
	ADASYN	0,645	0,522	0,620	0,633	0,355	0,600	0,462
	Set de Treino	0,484	0,273	0,441	0,436	0,516	0,300	0,250

Fonte: Autor

Legenda:

- ● – Melhores valores com os dados da amostra de treino SMOTE.
- ● – Melhores valores com os dados da amostra de treino ADASYN.
- ● – Melhores valores com os dados da amostra de treino do *data set* de treino original.
- * – Melhores valores em absoluto.

Nota: Com os dados da amostra SMOTE, o melhor modelo é obtido com o algoritmo SVM. Com os dados da amostra ADASYN, o melhor modelo é obtido com o algoritmo RF. E com os dados do *data set* de treino original, o melhor modelo é também obtido com o algoritmo RF.

Os cinco melhores modelos encontrados neste estudo foram:

1º. SVM com SMOTE.
2º. RF com *data set* de treino original.
3º. C5.0 com SMOTE.
4º. ADB com SMOTE.
5º. NN com ADASYN.

```
CREATE PROCEDURE previsao_mlq
AS
EXECUTE sp_execute_external_script
     @language = N'R'
    , @script = N'

library(RODBC)
library(e1071)

# Conectar à Base de Dados e fazer upload dos novos dados
myConn <- odbcDriverConnect("driver=SQL Server;server=VULCANO\\SQL_QUIDGEST_JC;
                            uid=xxxxx;pwd=xxxxx;database=MLQ")

final_data <- sqlQuery(myConn, "select * from MLQPREVI order by idprop")
final_data <- final_data[,c(-1,-13,-14)]
colnames(final_data) <- c("GANHOU","GASTSOFT","GASTO_ANUAL",
                          "AVG_PRAZO_ENTREGA_ANUAL","PRECO","PRAZO",
                          "P_PRECO","P_PRAZO","NREMPREG","IDADE_FORN","VENDAS")

n <- nrow(final_data)
i = 1

# Normalizar os dados com os valores da média e desvio padrão usados para treino
while(i <= n){

  final_data[i,2]  <- ((final_data[i,2] - 6577837.278)/ 12822251.1)
  final_data[i,3]  <- ((final_data[i,3] - 1393630.418 )/ 2343054.795)
  final_data[i,4]  <- ((final_data[i,4] - 274.5 )/ 235.873)
  final_data[i,5]  <- ((final_data[i,5] - 201447.934 )/ 217532.395)
  final_data[i,6]  <- ((final_data[i,6] - 83.5 )/ 139.07)
  final_data[i,7]  <- ((final_data[i,5] - 3.267 )/ 11.992)
  final_data[i,8]  <- ((final_data[i,8] - 0.636 )/ 3.631)
  final_data[i,9]  <- ((final_data[i,9] - 554.614 )/ 1755.888)
  final_data[i,10] <- ((final_data[i,10]- 23.071 )/ 7.269)
  final_data[i,11] <- ((final_data[i,11]- 10064872.185 )/ 15139454.257)

  i = i+1
  }

# Upload do Modelo que está na path indicada na solução
path_model <- sqlQuery(myConn, "select top 1 pathmodl from mlqglob")
full_path  <- paste0(path_model$pathmodl,"\SVM_BEST_MODEL_SMOTE.rds")
svmmodel   <- readRDS(full_path)

# Previsão
predsvm <- predict(svmmodel, final_data, probability = TRUE)
predsvm <- as.data.frame(attributes(predsvm)$probabilities[,2])
round(predsvm,3)

# Transformar a percentagem para um valor em percentagem e
# marcar todos os registos como "Previsto"
i = 1
while(i <= n){
sqlQuery(myConn,paste0("UPDATE MLQPREVI SET PBGANHAR = (",predsvm[i,1]*100
                       WHERE idprop = ",i,";"))
i <- i+1}

sqlQuery(myConn,"UPDATE MLQPREVI SET PREVISTO = 1")

'
```

Figura 22 – R *Script* em SQL para previsão de novos registos
Fonte: Autor