LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO

MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

APLICAÇÃO DE MACHINE LEARNING NO COMBATE AO BRANQUEAMENTO DE CAPITAIS E AO FINANCIAMENTO DO TERRORISMO

ALEXANDRE MIGUEL GONÇALVES GOMES

OUTUBRO - 2019

LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO EM

MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

RELATÓRIO DE ESTÁGIO

APLICAÇÃO DE MACHINE LEARNING NO COMBATE AO BRANQUEAMENTO DE CAPITAIS E AO FINANCIAMENTO DO TERRORISMO

ALEXANDRE MIGUEL GONÇALVES GOMES

ORIENTAÇÃO:

PROF. DR. JOSÉ PEDRO GAIVÃO
DR.JOÃO PAULO CARVALHO

OUTUBRO - 2019

Agradecimentos

O presente trabalho não seria possível sem a colaboração da Quidgest, e por isso agradeço ao Dr. João Paulo e ao departamento onde estive inserido: Banca, Seguros e Serviços Financeiros.

Agradeço também ao Prof. Doutor José Romana Gaivão não só pelo contributo prestado no decorrer do estágio e pelos seus conselhos, mas também pelo apoio concedido no desenvolvimento deste relatório.

Aos alunos e professores da NOVA IMS, UFF e ISEG que, de alguma forma, contribuíram durante o meu percurso académico na minha aprendizagem.

À minha família agradeço o apoio prestado principalmente aos meus pais. Sou muito grato pelo vosso apoio, carinho, educação e dedicação, isto fez de vocês os meus principais pilares durante toda a minha formação, proporcionando-me oportunidades únicas. Agradeço também ao meu padrinho, Júlio, ao meu irmão, Tiago, ao meu primo, Daniel, por terem contribuído de forma excecional durante todo este percurso.

Aos meus amigos, em particular a vocês: David Mendes, João Custódio e Pablo Gadea pelo vosso companheirismo.

Por fim e não menos importante, agradeço à minha querida xodó jurídica (Giovanna Lacerda), pelo amparo, incentivo e ajuda incondicional prestada na elaboração deste trabalho.

Resumo

Este trabalho resulta de um estágio desenvolvido na Empresa Quidgest, S.A. O trabalho final de mestrado versa sobre uma aplicação de *Machine Learning* na resolução da problemática de combate ao branqueamento de capitais e ao financiamento do terrorismo. Tal problema é conhecido como um caso de dados desbalanceados. Por conseguinte, a questão é abordada no decorrer do trabalho, apresentando várias formas de resolução. São ainda tratados os conceitos *Machine Learning*, *Data Mining* e *Knowledge-Discovery in Databases*. No âmbito do *Machine Learning*, o presente trabalho apenas se debruça sobre algoritmos supervisionados. Mais especificamente, os classificadores *Random Forest*, *Adaboost* e *Boosting* C5.0. Tais métodos foram aplicados sobre um repositório de dados que se encontravam alojados no sistema de gestão de base de dados Microsoft SQL Server.

A investigação seguiu a metodologia CRISP-DM e teve a sua implementação no *software* R.

Palavras-Chave: Aprendizagem Supervisionada, Machine Learning, Random Forest, Adaboost, Boosting, C5.0, Data Mining, Classe Desbalanceada, Árvores de Decisão

Abstract

This work results from an internship developed at Quidgest, S.A. This Master Final Work deals with an application of the Machine Learning in order to solve the problem of money laundering and the financing of terrorism. This problem is known as a case of unbalanced data. Therefore, the issue is addressed in the course of the work, presenting various forms of resolution. The concepts of Machine Learning, Data Mining and Knowledge-Discovery in Databases are also discussed. In Machine Learning, this paper only focuses on supervised algorithms. More specifically, the classifiers: Random Forest, Adaboost, and Boosting C5.0. These methods were applied to a data repository that was hosted in Microsoft SQL Server database management system.

The research followed the CRISP-DM methodology and was implemented in the R software.

Keywords: Supervised Learning, Machine Learning, Random Forest, Adaboost, Boosting, C5.0, Data Mining, Unbalanced Class, Decision Trees

Sumário

Índice de figuras

Glossário de Abreviaturas e Siglas

ACC - Accuracy

AML - Anti-Money Laundering

AUC - Area Under the Curve

BCFT - Branqueamento de Capitais e ao Financiamento do Terrorismo

CSV - Comma-Separated Values

CRISP-DM - Cross Industry Standard Process for Data Mining

DD - Data-driven

DM - Data Mining

FSS - Feature Subset Selection

GA - Genetic Algorithm

GAFI - Grupo de Acão Financeira

IA - Inteligência Artificial

IOT - Internet das Coisas

KAIS - Knowledge and Information Systems

KDD - Knowledge discovery in database

ML - Machine Learning

MST - Minimum Spanning Tree

NN - Neural Network

PR - Precision-Recall

RF - Random Forest

ROC - Receiver Operating Characteristics

SGBD - Sistema de Gestão de Base de Dados

SVM - Support Vector Machines

SMOTE - Synthetic Minority Over-sampling Technique

TFM - Trabalho Final de Mestrado

T-SQL - Transact-Scructured Query Language

Capítulo 1. Introdução

1.1 Enquadramento

Compreende-se que as técnicas de *Machine Learning* (ML) e de inteligência artificial (IA) oferecem benefícios significativos aos tomadores de decisão, em termos de novas abordagens de modelação e previsão, a partir de dados (Aziz *et al.*, 2019) . A IA está, cada vez mais, presente no mundo empresarial. A título de exemplo, tem-se uma das maiores empresas de *streaming*, como o Netflix, a qual utiliza inteligência artificial no seu sistema de recomendações (Levy, 2010). A aplicação de técnicas de ML tem ganho uma maior popularidade na última década, uma vez que, cada vez mais, as organizações produzem dados com uma maior velocidade, variedade e volume. Estes três "Vs" caracterizam o que é popularmente conhecido como *Big Data*.

O avanço tecnológico tem impulsionado a popularidade do ML, devido às significantes melhorias de poder de processamento computacional e armazenamento de grandes quantidades dados (Coenen, 2011). Nos dias de hoje, as oportunidades para a obtenção de dados, fora dos sistemas operacionais, aumentaram substancialmente (Marquesone, 2016) . No contexto da internet das coisas (IOT), imensidão de objetos podem ser gerados por sensores, *RFID*, *smartwatches*, *IP Camera* e entre outros (Marquesone, 2016) .

Os primeiros trabalhos em ML surgiram em meados de 1990s, projetos desenvolvidos com base em modelos que "aprendem" com os dados (Marr, 2016) , comummente , conhecidos em inglês como modelos *data-driven* (DD). Ainda na década de 90, cientistas da área criaram programas capazes de lidar com a imensidão de volume de dados, com o propósito de analisá-los, extrair conclusões e obter conhecimento (Marr, 2016).

As empresas progressivamente dependem menos do instinto do líder, ao invés disso, utiliza-se frequentemente análises de dados (Brynjolfsson *et al.*, 2011). Segundo o Diretor do centro de investigação de *Digital Bussines* do MIT, as companhias que adotam decisões com suporte em modelos *data-driven* conseguem aumentos na sua produtividade estimada entre cinco e seis por cento (Brynjolfsson *et al.*, 2011).

1.2 Empresa Quidgest - Consultores de Gestão, SA

Quidgest é uma empresa especializada em Engenharia de Software, fundada em 1988 e de origem portuguesa. A empresa conta com mais de 100 colaboradores entre 12 equipas especializadas, cada equipa tem uma área de ação distinta. Em resultado disso, a empresa tem uma grande variedade de clientes, como clientes da área da saúde, banca, instituições de ensino, seguradoras, instituições públicas, entre outras. No momento atual, presta serviços no mercado nacional e internacional, tendo como principais países de atuação Portugal, Alemanha, Moçambique, Macau, Jamaica, Cabo Verde, Brasil e Argentina.

O Genio é um software desenvolvido pela Quidgest. Esta plataforma tem o objetivo de gerar software de forma automática, rápida e ágil. Assenta em mais do que uma linguagem de programação, como C#, Java, C++ e também nos sistemas de gestão de base de dados (SGBD). O sistema é tido como um dos mais adotados no mercado empresarial, como SQL Server, Oracle e DB2 (Quidgest, 2019) . O *software* é considerado *user-friendly*, uma vez que foi projetado não apenas para engenheiros de *software*, mas também para gestores de negócio. A plataforma permite que qualquer indivíduo possa desenvolver um sistema de informação, mesmo não tendo qualquer formação em programação. Isso tem sido evidenciado pelo uso dos clientes da empresa no desenvolvimento de sistemas de gestão de recursos humanos, gestão documental, gestão financeira, páginas de internet, gestão patrimonial, etc (Quidgest, 2019).

1.3 Objetivo do estágio

O estágio realizado e descrito no contexto deste Trabalho Final de Mestrado (TFM), teve início de uma parceria realizada entre a instituição de ensino Instituto Superior de Economia e Gestão, no âmbito do mestrado em Métodos Quantitativos para a Decisão Económica e Empresarial, e a empresa Quidgest – Consultores de Gestão, SA.

Este estágio foi realizado por um período de 3 meses, entre Novembro de 2018 e Janeiro de 2019. Foi desenvolvido na equipa responsável pela prestação de serviços na área da Banca, Seguros e Soluções Financeiras. O departamento conta com uma solução conhecida como *QUID AML(Anti-Money Laudering)*. O seu desenvolvimento foi feito

com base em diretrizes do Banco de Portugal e recomendações do Grupo de Ação Financeira (GAFI), tendo a finalidade de apoiar as instituições financeiras no combate ao Branqueamento de Capitais e ao Financiamento do Terrorismo (BCFT). A atual solução permite que as instituições financeiras, em tempo real e de forma automática, possam identificar transações, entidades e contas suspeitas. Em 2017, Portugal foi o melhor classificado no âmbito do combate ao branqueamento de capitais a par da Espanha e Itália (Lígia Simões, 2017). É possível evidenciar, dessa forma, os esforços realizados em Portugal, no que concerne ao BCFT, tendo em vista que o departamento se mantém atento e empenhado em melhorar a sua solução, para atender as reais necessidades dos seus clientes.

A solução Quid AML conta com um repositório de alertas gerados nos últimos quatro anos de possíveis fraudes. Depois de analisadas, 99,4% dos alertas foram arquivados como falsos positivos. Entendem-se como falsos positivos alertas de pouca ou nenhuma relevância no combate ao BCFT. Surge assim a necessidade de introduzir ML na sua atual solução. Isto é, criar um sistema que classifique cada suspeita de fraude de forma automática com um determinado nível de prioridade. Os níveis de prioridade estão fixados numa escala intervalar com limite inferior 0 e limite superior 100, serão considerados não prioritários valores compreendidos entre 0 e menores que 50, e como prioritários valores iguais ou superiores 50. A solução será desenvolvida no *software* R e posteriormente integrada no SGBD SQL Server.

1.4 Estrutura do TFM

O seguinte TFM encontra-se organizada da seguinte forma: no capítulo 2 são apresentados os conceitos *Machine Learning e Data Mining*. Ainda neste capítulo é realizada a revisão bibliográfica dos algoritmos adotados neste Trabalho Final de Mestrado. No capítulo 3 segue-se a apresentação das principais tecnologias e metodologias aplicadas em trabalhos desta natureza. O capítulo 4 é constituído pela aplicação prática de cada uma das etapas que compõe a metodologia CRISP-DM, sendo possível acompanhar a implementação no *software* R e opções metodológicas. No *software* R são também descritas as principais bibliotecas, assim como, as adotadas na realização deste projeto. Por fim, no capítulo 5, são discutidas as conclusões e propostas para futuros trabalhos.

Capítulo 2. Revisão da literatura

2.1 Aplicações de *Machine Learning* no combate à fraude.

É possível evidenciar que os algoritmos de ML fornecem ferramentas, as quais apoiam a deteção de fraude e têm sido aplicadas, com sucesso, no combate à lavagem de dinheiro (Bolton e Hand, 2013). Os primeiros trabalhos publicados de aplicação de ML no combate à fraude surgiram na década de 90. Em 1994, foi publicado um artigo de aplicação de redes neuronais artificias na deteção de fraude dos cartões de crédito (Altman *et al.*, 1994). Em 1995, foi desenvolvida uma aplicação de árvores de decisão na deteção de lavagem de dinheiro (Senator *et al.*, 1995). Desde 2005 a 2017, conforme a pesquisa Salehi *et al.* (2017), pode-se observar, na Figura 1, que surgiram vários artigos em que são aplicadas técnicas de ML no combate e deteção à lavagem de dinheiro. Sendo possível constatar que foram aplicados inúmeros algoritmos de aprendizagem supervisionada, desde *Support Vector Machines* (SVM), Árvores de Decisão, *Naive Bayes* e Redes Neuronais *(Neural network* - NN*)*, como as várias variantes de cada método.

Simultaneamente, com maior frequência, tem-se popularizado o uso de algoritmos de aprendizagem não supervisionada no presente contexto de combate à fraude. Segundo Salehi *et al.* (2017), os métodos frequentemente adotados, entre os algoritmos não supervisionados, possuem enfâse notória: *Clope Clustering*, *Minimum Spanning Tree* (MST) e *K-means*. Esses métodos não se enquadram no âmbito deste trabalho, consequentemente, não serão abordados com detalhe neste segundo capítulo.

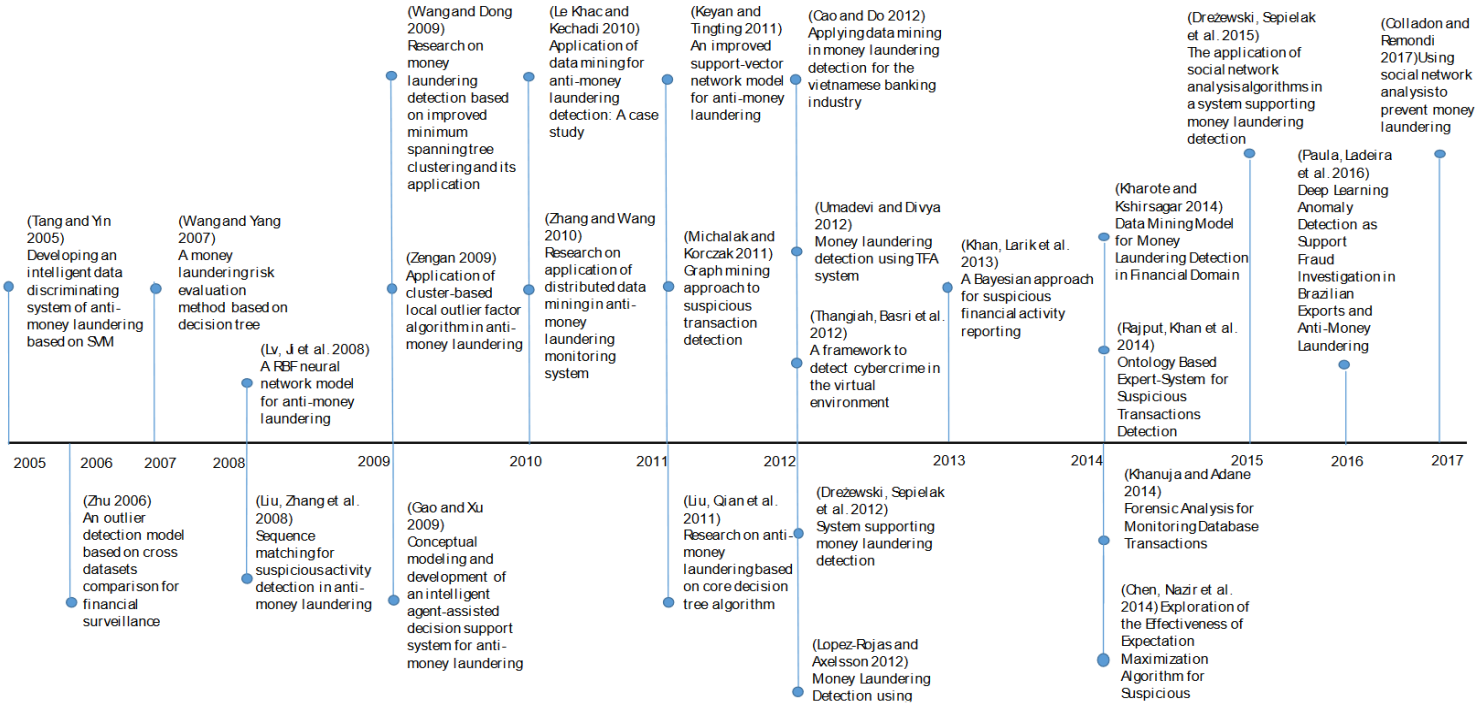


Figura 1 – Principais artigos publicado entre 2005 e 2017 no combate ao BCFT com adoção de técnicas de ML

Fonte: (Salehi *et al.*, 2017)

2.2 *Data Mining* e *Machine Learning*

Data Mining (DM) e *Machine Learning* são conceitos vastamente abordados em trabalhos de *Data Science*. Isto porque ambos se encontram interligados quando se trata de obter conhecimento e padrões. No entanto, é possível diferenciá-los, uma vez que ML é apenas uma das várias áreas de conhecimento que integram o DM. Este é também composto pela estatística, análise espacial de dados, computação de alto desempenho, computação natural, etc (de Castro e Ferrari, 2017).

DM ou o processo KDD (*Knowledge discovery in database*), apresentado por Fayyad *et al.* (1996), muitas vezes, são utilizados como sinónimos. No entanto, pode-se entender o DM como a etapa de extração de conhecimento com a aplicação das seguintes técnicas: *clustering*, regras de associação, análise descritiva, redução de dimensionamento, previsão e classificação.

Conforme Fayyad *et al.* (1996), KDD é um processo não trivial que identifica padrões válidos, novos, úteis e compreensíveis. O processo é dividido em cinco etapas:

seleção de dados, pré-processamento de dados, transformação, mineração de dados e interpretação/validação. Como se pode observar na *Figura 2*.

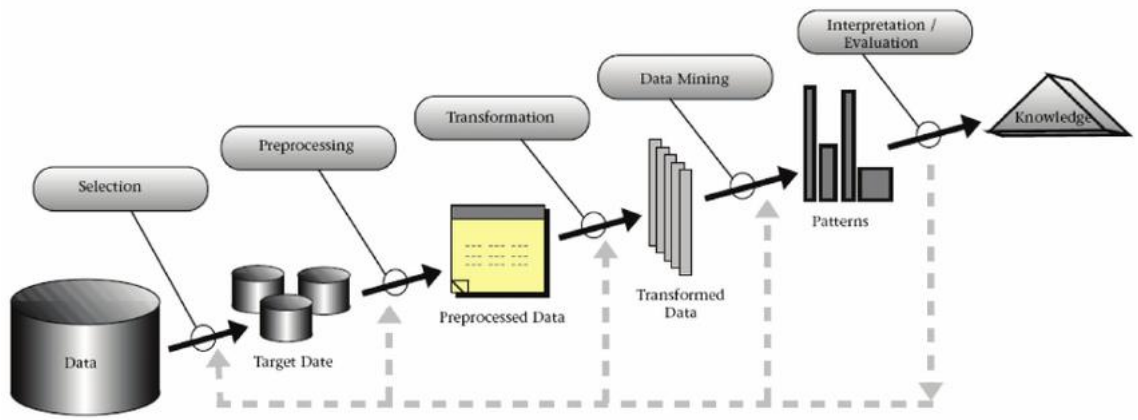


Figura 2 - Processo KDD

Fonte: (Fayyad *et al.*, 1996)

Após descrito o processo DM, entende-se por *Machine Learning* a criação de um modelo que seja capaz de aprender com os dados e responder ao problema proposto na investigação. De acordo com Gama *et al* (2011) , pode-se classificar as técnicas de ML como preditivas ou descritivas. As técnicas preditivas têm como objetivo a previsão ou classificação de um valor contínuo ou discreto, sendo utilizado um conjunto de dados *input*. Entende-se como dados *input* o conjunto de valores das variáveis consideradas relevantes para o modelo de previsão. Diferencia-se desse conjunto de variáveis os dados da variável *output*, visto que esta consiste no valor que se pretende estimar.

As técnicas descritivas, diferentemente das preditivas, caracterizam-se pela procura de padrões e principais singularidades do repositório. Sendo assim possível extrair características informativas como as relações existentes entres os dados. No caso das técnicas descritivas não existe uma variável *output*, esta não é conhecida, apenas se utiliza o conjunto *input*. Como principais algoritmos desta técnica, podem-se mencionar as regras de associação, *clustering* e deteção de anomalias.

2.3 Modelação estatística

Segundo Breiman (2011), existem duas culturas na modelação estatística: *data modeling culture* e *algorithmic modeling culture*. Na primeira o principal objetivo reside não só na inferência estatística, mas também na interpretação dos parâmetros. Enquanto que na segunda o principal foco é a predição, não existindo previamente um modelo correto e sendo esta a cultura adotada em ML.

2.4 Árvores de decisão

Compreende-se por árvore de decisão um modelo não paramétrico muito utilizado em aprendizagem supervisionada. São úteis para resolver problemas de classificação e também de regressão. Para problemas de variável dependente quantitativa são conhecidas como árvores de regressão. No entanto, em casos de variável de interesse qualitativa, essas árvores são chamadas de árvores de classificação. Este modelo supervisionado apresenta uma estrutura de árvore, é composta por uma raiz, folhas e ramos. Na literatura, é várias vezes referenciado como um modelo *"Divide-and-conquer",* visto que o algoritmo particiona as instâncias em sub-problemas, tornando-o num problema mais simples (Witten *et al.*, 2017). Um dos fatores que mais influencia a grande popularidade deste modelo é a sua fácil interpretação, sendo possível extrair regras da solução gerada, como pode ser observado na Figura 3.

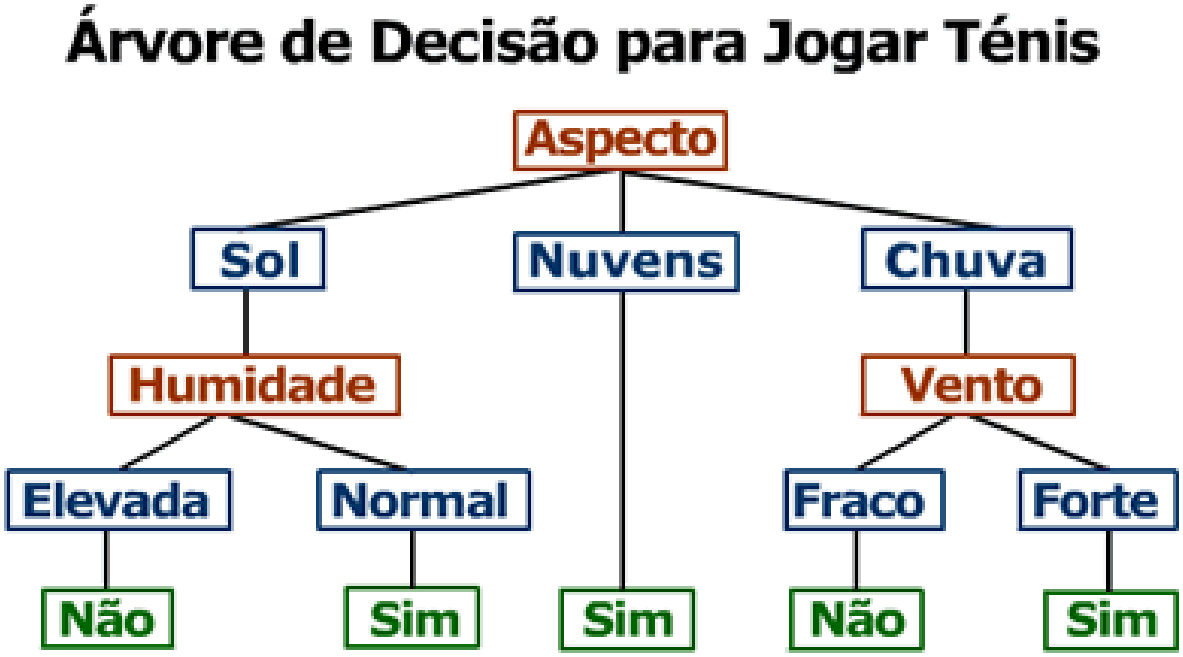


Figura 3 - Árvore de decisão para classificação
Fonte: (Freitas, sem data)

A literatura refere diferentes técnicas de ramificação, que sugerem critérios diferentes para a seleção dos atributos, como por exemplo CART, ID3, C4.5 e C5.0, foram apresentados, respetivamente, em Breiman *et al.* (1984), Quinlan (1986) , Quinlan (1993) e Rulequest (2019 b).

2.4.1 Algoritmo C4.5

O método de indução de árvores C4.5 foi publicado em 1993 por Quinlan (1993). Em 2008, este surgiu no top 10 de algoritmos mais utilizados (Wu *et al.*, 2008), artigo publicado no fórum internacional da comunidade *Knowledge and Information Systems* (KAIS). O critério utilizado em C4.5 para ramificar é a "razão de ganho", a qual tem por base o cálculo, a priori, da Entropia. Este é um conceito que surge em problemas de árvore de decisão com a técnica ID3, do mesmo autor de C4.5. Por conseguinte, esse algoritmo é conhecido como uma melhoria da técnica ID3. Com o seu surgimento o método veio colmatar algumas limitações da técnica ID3, passando a ser capaz de suportar atributos contínuos, *missing values e outliers.* Tal algoritmo também conta com a técnica de *pruning*, o que permite evitar problemas de *overfitting* (Witten *et al.*, 2017).

2.4.2 Algoritmo C5.0

Apresentado como uma melhoria ao já existente C4.5, surgiu como versão comercial em 1997. Possui como principal vantagem a possibilidade de escalabilidade, traduzindo-se em grandes ganhos computacionais, dada a possibilidade de trabalhar em múltiplos *cores* e CPUs (Wu *et al.*, 2008). Conforme o estudo publicado em Rulequest (2017 a) , o método C5.0, também conhecido por See5, apresenta uma melhor performance que o C4.5. Pode-se evidenciar tal desempenho porque os resultados, na sua globalidade, permitiram concluir que C5.0 é muito mais rápido, utiliza menos memória RAM e, mais importante ainda, gera uma árvore de menor dimensão.

O algoritmo surge com novas funcionalidades, de destacar o *Cost-Sensitive Learning e Boosting*, inspirado no já existente *Adaboost*. A primeira funcionalidade tem-se mostrado adequada em problemas de variável dependente desbalanceada (Thai-Nghe *et al.*, 2010; Krawczyk, 2016).

2.5 *Ensemble Learning*

Compreende-se por *ensemble* a classificação de dados que pode acontecer por meio de combinação múltipla entre modelos. Considera-se que este é homogéneo, quando são utilizados os mesmos modelos para inferir o output. Caso contrário, quando utilizadas técnicas diferentes, considera-se o *ensemble* heterogéneo. Um *ensemble* normalmente apresenta resultados de *accuracy* melhores que qualquer um dos classificadores bases (Han *et al.*, 2012).

A combinação de modelos tem sido, cada vez mais, adotada na área financeira, com maior incidência em estudos de risco de crédito. Com base em publicações nos anos mais recentes, como mencionado em Zareapoor e Shamsolmoali (2015), é possível constatar que o uso de *ensemble* não só apresentou melhores valores nas métricas adotadas na avaliação dos modelos utilizados, mas também se verificou como um classificador robusto em situações de classes desbalanceadas. Tais factos foram evidenciados nas seguintes pesquisas Maclin e Opitz (2011) e posteriormente em Patel *et al.* (2019).

As várias técnicas para combinar classificadores tiveram início na década de 90, com o surgimento dos métodos *Boosting* e *Bagging*, apresentados respetivamente em Freund e Schapire (1996) e Breiman (1996). A principal motivação destes modelos consistia em transformar *weak learners* em *strength learners*. O conceito *weak learner*, proposto em Schapire (1990) , consiste na execução de modelos sem qualquer complexidade. O que permite obter classificadores ligeiramente melhores que a pura aleatoriedade e, posteriormente, de forma sequencial ou paralela, melhorá-lo a fim de obter um *strength learner*.

Na Figura 4, é apresentada uma ilustração exemplificativa de um método ensemble paralelo.

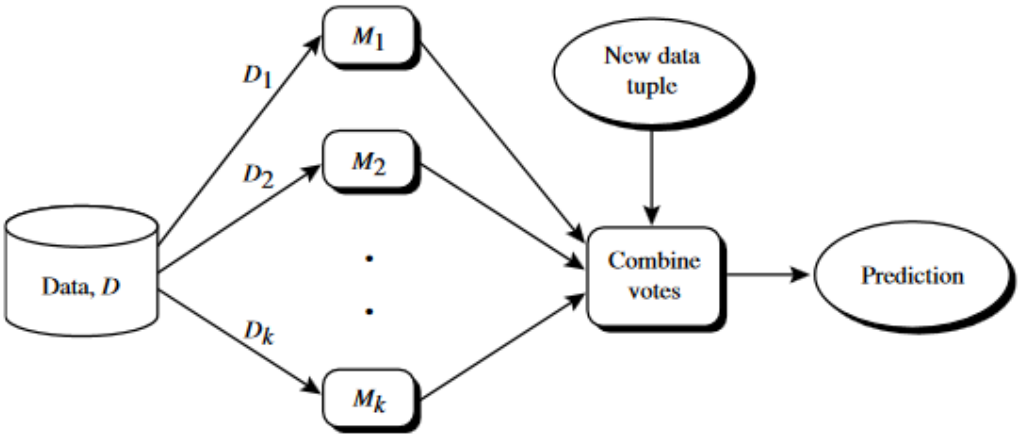


Figura 4 - Ensemble paralelo
Fonte: (Boonkwang *et al.*, 2018)

2.5.1 *Adaboost*

O algoritmo *AdaBoost*, ou também conhecido como *Adaptative boosting*, considerado como um dos mais populares, é um método da classe dos algoritmos de *boosting*, apresentado por Freund e Schapire (1997). Inicialmente, foi concebido para a resolução de problemas de classificação, conhecido como *Discrete Adaboost*. Mais tarde, surgiram adaptações do algoritmo para resolução de problemas de regressão. O algoritmo é executado de forma sequencial, o que permite adicionar, no decorrer do treino, conhecimento com os erros cometidos durante a execução do mesmo. Para isso o algoritmo atribui pesos às instâncias utilizadas ao longo do treino do modelo. As instâncias que forem incorretamente classificadas serão atribuídas pesos mais altos que as restantes. Essas, por sua vez, terão uma probabilidade maior de serem analisadas pelo classificador posterior, o que permite uma maior especialização nas instâncias incorretamente classificadas. Na aplicação do algoritmo *AdaBoost*, será gerado um vetor $J = [J_1, \ldots, J_n]$ de classificadores. No primeiro classificador, J_1, todas as instâncias terão pesos iguais. Após gerado o primeiro modelo, será comparado o *output* previsto com o *output* real de cada instância. Em seguida, com base em alguma medida de atribuição de pesos, sucederá uma diminuição do peso de cada instância corretamente prevista e o inverso nos restantes casos. Por fim, com base nos **n** classificadores gerados, é alcançado um classificador final. O algoritmo apresenta, como principais características, a falta de interpretabilidade e um alto custo computacional. Na Figura 5, surge uma pequena ilustração do algoritmo *Adaboost* e na Figura 6 o respetivo pseudocódigo.

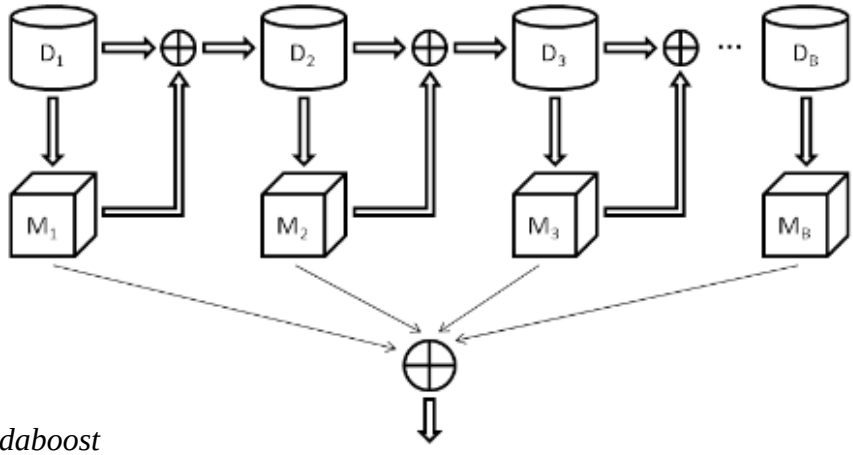


Figura 5 – *Adaboost*

Fonte:(Gironés Roig *et al.*, 2017)

Declaração de variáveis

T # Número de classificadores gerados

N # Número de instâncias

$\boldsymbol{D_{(t,i)}}$ # peso da instância i no classificador T.

Passo 1

$t = 1$

$D_{(1,i)} = \frac{1}{N}$ # Definir os pesos de todas as instâncias no classificador t=1 como $\frac{1}{N}$

Repetir

Treinar o modelo h(t)

Calcular $Erro_t$, $Erro_t = \sum_{i:h_t(x_i)\neq y_i} D_{(t,i)}$ # Cálculo do erro referente ao classificador t

Calcular $a_t = \frac{1}{2}\ln(\frac{1-Erro_t}{Erro_t})$ # Ponderação do classificador obtido

Para i =1 até N **faz**

$D_{(t+1,i)} = \begin{cases} D_{(t,i)}e^{-a_t}, \ se\ h_t(x_i) = y_i \\ D_{(t,i)}e^{a_t}, \ c.c. \end{cases}$ # Atualização dos pesos de cada instância

Fim Para

$t = t + 1$

$D_{(t,i)} = \frac{D_{(t,i)}}{\sum_{i=1}^{N} D_{(t,i)}}$,# Normalização dos pesos calculados

t até T

Hipótese final

$H(X) = sign(\sum_{t=1}^{T} a_t h_t(x)$ #Combinação Linear dos modelos obtidos.

Figura 6 - Pseudocódigo *Adaboost*

Fonte: (Jansson, 2016)

2.5.2 *Random Forest*

A combinação de árvores de decisão, ou florestas aleatórias (Random Forest -RF), é também um método muito popular entre os algoritmos *ensemble learning* na resolução de problemas de classificação e regressão. Foi apresentado por Breiman (2001). Conforme demonstrado na *Figura 8*, o algoritmo visa criar várias árvores de decisão em paralelo. Para cada árvore é escolhida, de forma aleatória e sem reposição, uma amostra do conjunto original de variáveis. O conjunto de treino é recalculado para cada classificador, sendo necessário efetuar a sua seleção de forma aleatória e com reposição, o que permite criar uma maior diversidade entre os classificadores. Em problemas de classificação, a previsão da classe é obtida pela classe majoritária entre as várias árvores geradas. Diferentemente, em problemas de regressão, o resultado será a média dos valores de saída dos classificadores treinados.

Contrariamente às árvores de decisão, a aplicação de *random forest* torna a interpretação uma tarefa não trivial, uma vez que o output obtido é resultado de uma combinação de árvores. O algoritmo, muitas vezes, é comparado ao desempenho obtido com aplicação do *AdaBoost*, o qual apresenta alta robustez na presença de erros e *outliers* (Han *et al.*, 2012). Por conseguinte, é um algoritmo adequado para os conjuntos de dados que apresentam muitas variáveis. Em contrapartida dos seus benefícios, é considerado como principal obstáculo o alto custo computacional, que dependerá, em larga medida, de outros fatores, tais como o número de árvores geradas, número de variáveis, bem como do tamanho do conjunto de treino (Baranauskas *et al.*, 2018). Estando representado na Figura 7 o pseudocódigo do *random forest* e o seu fluxograma na Figura 8.

Declaração de variáveis.
Dataset #conjunto de todas as instâncias utilizadas para treino
T #número de árvores geradas
$\boldsymbol{h_t}$# Classificador t com t=1,…,T
N #definir o tamanho da amostra a aplicar em cada árvore
Treino #conjunto de dados utilizado no treino de cada classificador
D #vetor formado por todas as variáveis
d #número de variáveis utilizadas em cada classificador, em que d < D, usualmente, log2D+1.
dd #vetor com dimensão **d** que armazena as variáveis utilizadas em cada iteração.
Passo 1
t = 1
Repetir
Escolher aleatoriamente e sem reposição **d** variáveis do vetor **D** e armazenar no vetor **dd**

Escolher aleatoriamente e com reposição **N** instâncias do **Dataset** e substituir os dados existentes em **Treino**

Treinar o classificador $\boldsymbol{h_t}$ com o conjunto de **Treino** incluindo apenas as variáveis do vetor **dd.**

$t = t + 1$

t **até** T

Figura 7 - Pseudocódigo *Random Forest*
Fonte: (Flach, 2012)

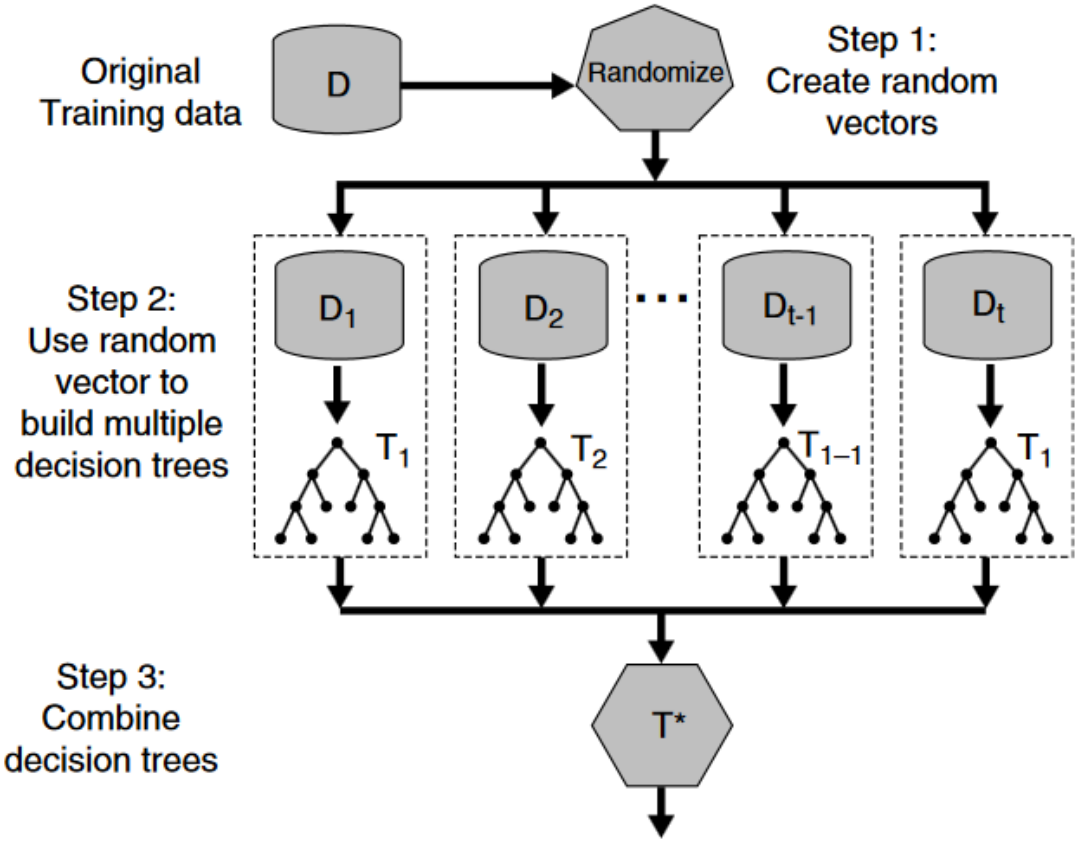


Figura 8 - Random Forest

Fonte: (Tan *et al.*, 2013)

2.5.3 Boosting C5.0

O método de *boosting* introduzido no algoritmo C5.0 é inspirado no AdaBoost. Esses dois algoritmos possuem semelhanças como inicialmente todas as instâncias possuírem igual peso de 1/N. Contudo, é possível diferenciar um do outro, a principal distinção é a forma de atualizar os pesos das instâncias corretamente e incorretamente classificadas. No C5.0, as instâncias em que as suas classes previstas não coincidem com

as classes reais terão uma maior penalização. A consequência desse evento resultará em um aumento superior no seu peso atribuído, quando comparado com o decréscimo do peso da instância corretamente prevista, como se pode observar na *Figura 9*. Como é possível observar no pseudocódigo na Figura 10.

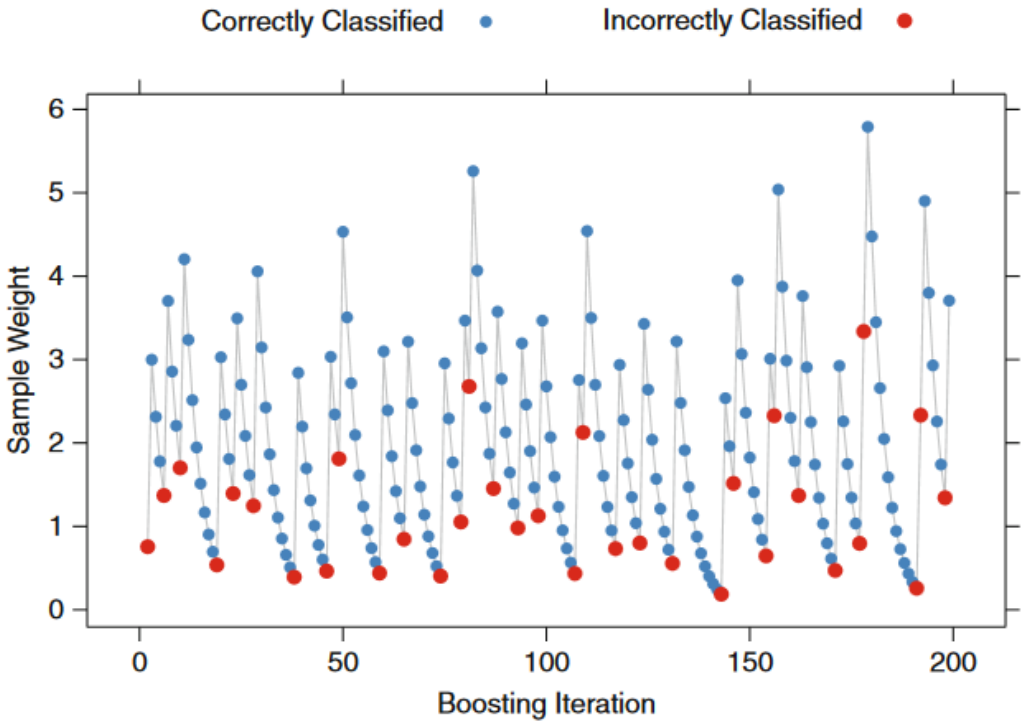


Figura 9- Ilustração de atualização dos pesos durante a estimação do modelo Boosting C5.0

Fonte: (Kuhn e Johnson, 2013)

Este método também permite terminar a geração de árvores antes do número de iterações definido. Para isso, será necessário cumprir uma de duas propriedades: 1) O somatório do peso das instâncias incorretamente classificadas não seja superior a 0.10; 2) O modelo não consiga classificar corretamente mais de 50% das instâncias.

Declaração de variáveis
T #número de classificadores gerados
$\boldsymbol{h_t}$# Classificador t com t=1,..,T
N #número de instâncias
N_ #número de instâncias incorretamente classificadas
$\boldsymbol{D_{(i,t)}}$ #peso da instância **i** no classificador **t**.
S+ #Somatório dos pesos das instâncias corretamente classificadas
S_ #Somatório dos pesos das instâncias incorretamente classificadas
Passo 1

$t = 1$

$D_{(1,i)} = \frac{1}{N}$ #Definir os pesos de todas as instâncias no classificador $t = 1$ como $\frac{1}{N}$

Repetir

Treinar a árvore de decisão h_t

Para i =1 até N **faça**

Calcular *midpoint* , *midpoint* =$\frac{1}{2}[\frac{1}{2}(S_+ + S_-) - S_-]$

Se $(h_t(x_i) = y_i)$ **então**

$D_{(i,t)} = D_{(i,t-1)} \frac{S_+ - midpoint}{S_+}$

Senão

$D_{(i,t)} = D_{(i,t-1)} \frac{S_+ - midpoint}{N_-}$

Fim se

Se (S_ ≤ 0.10**) então #** se o somatório das instâncias incorretamente classificadas for inferior ou igual a 0.10, o algoritmo deve parar a sua execução

$t = T + 1$

Fim se

Se ($\overline{D_{(i,t)}}$,mal classificados > 0.5) **então** # se a média dos pesos das instâncias mal classificadas for maior que 0.5, o algoritmo deve parar a sua execução.

$t = T + 1$

Fim para

$t = t + 1$

t até T

Figura 10 - Pseudocódigo Boosting C5.0
Fonte: (Jansson, 2016)

Capítulo 3. Principais metodologias e tecnologias

3.1 Principais metodologias adotadas em projetos de ML

Existem diferentes metodologias que podem ser adotadas no desenvolvimento de projetos de *machine learning*. A exemplo disso, CRISP-DM, KDD e SEMMA são as mais elegidas pela comunidade KDNUGGETS, conforme publicado pela mesma Piatetsky (2019). Esta é uma comunidade que reúne vários cientistas de dados com inúmeros desafios no âmbito de *machine learning*. CRISP-DM é a abordagem mais utilizada, seguida de "própria metodologia" e SEMMA. *Cross Industry Standard Process for Data Mining* (CRISP-DM), que surgiu em 1996 pelas empresas DaimlerChryrler,

NCR e SPSS, foi concebida para dar resposta a qualquer problema independentemente da indústria. Essa metodologia assenta sobre seis estágios, trata-se de uma metodologia cíclica, sendo assim possível recuar e avançar entre os diferentes estágios, como se pode observar na *Figura 11*.

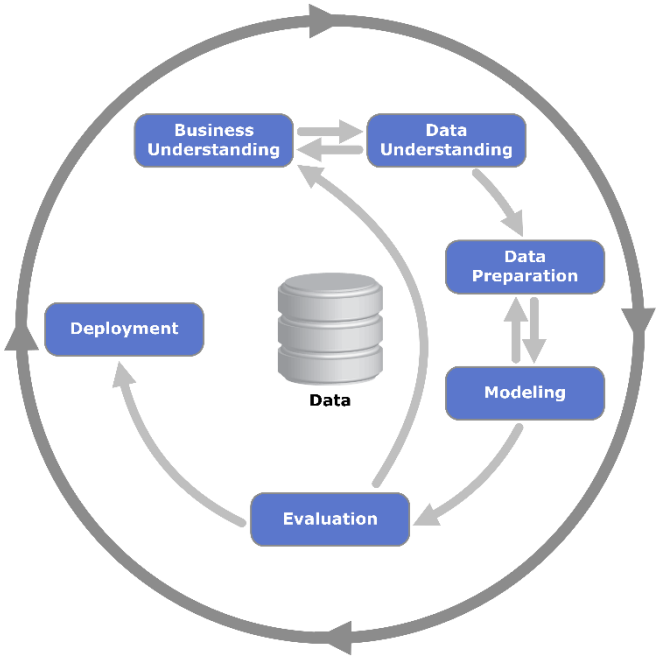


Figura 11 - CRISP-DM
Fonte: (Vorhies, 2016)

Os seis estágios são conhecidos como compreensão do negócio, compreensão dos dados, preparação dos dados, modelação, avaliação e execução.

O projeto desenvolvido neste TFM adota a metodologia CRISP-DM. Analisar-se-á cada estágio da metodologia adotada. A primeira etapa, compreensão do negócio, passou pela compreensão das necessidades do departamento, ou seja, os objetivos, os resultados esperados, bem como as principais ferramentas adotadas no projeto, por exemplo SQL Server 2014 e o s*oftware* estatístico R. No que diz respeito à segunda etapa, compreensão dos dados, foi realizada uma exploração exaustiva do repositório existente, de forma a conhecer quais dados se encontravam disponíveis e a sua granularidade para a elaboração deste projeto. Em seguida, no terceiro estágio, é realizada a preparação dos dados. Essa fase é composta pelas seguintes tarefas: identificação de valores ausentes, valores inconsistentes, transformação *one-hot encoding* e a criação de novas variáveis. Ainda na terceira fase, tem-se um grande subconjunto de variáveis redundantes e variáveis não relevantes para o estudo, dadas as características do repositório, o qual conta com centenas de tabelas e milhares variáveis, fenómeno conhecido por *curse of*

dimensionality. Desta forma surge a necessidade da aplicação de técnicas de seleção de um subconjunto de variáveis (*Feature Subset Selection* – FSS). A tarefa FSS permite obter modelos com melhor *performance*, interpretabilidade e uma execução em tempo útil (Baranauskas *et al.*, 2018) . Terminado o terceiro estágio, cumpre entender a etapa seguinte: modelação. Nesta etapa são aplicados os modelos escolhidos sobre os dados selecionados, *mineable data*, como por exemplo árvores de decisão, *random forest*, máquina de suporte de vetores, redes neuronais, regressão logística, *ensembles*, entre outros. Depois de estimados os modelos, segue-se a quinta etapa, o momento de avaliar os classificadores obtidos e comparar com as metas definidas *a priori*. Com base nos resultados alcançados, seguir-se-á ao sexto estágio. Esse dependerá da convergência dos resultados com os interesses da investigação para que se proceda à execução do modelo obtido. Situações diversas são os casos em que não exista essa convergência de resultados, uma vez que haverá a possibilidade de um *rollback* entre o primeiro e quarto estágio.

3.2 Principais tecnologias adotadas em projetos de *Machine Learning*

Conforme dados publicados na página de internet KDNUGGETS Piatetsky (2019), entre 2017 e 2019, o TOP 5 de ferramentas mais utilizadas em projetos de *machine learning* pela sua comunidade conta com os seguintes *software*: Python,

RapidMiner, R Language, Excel e Anaconda, como se pode observar na *Figura 12*.

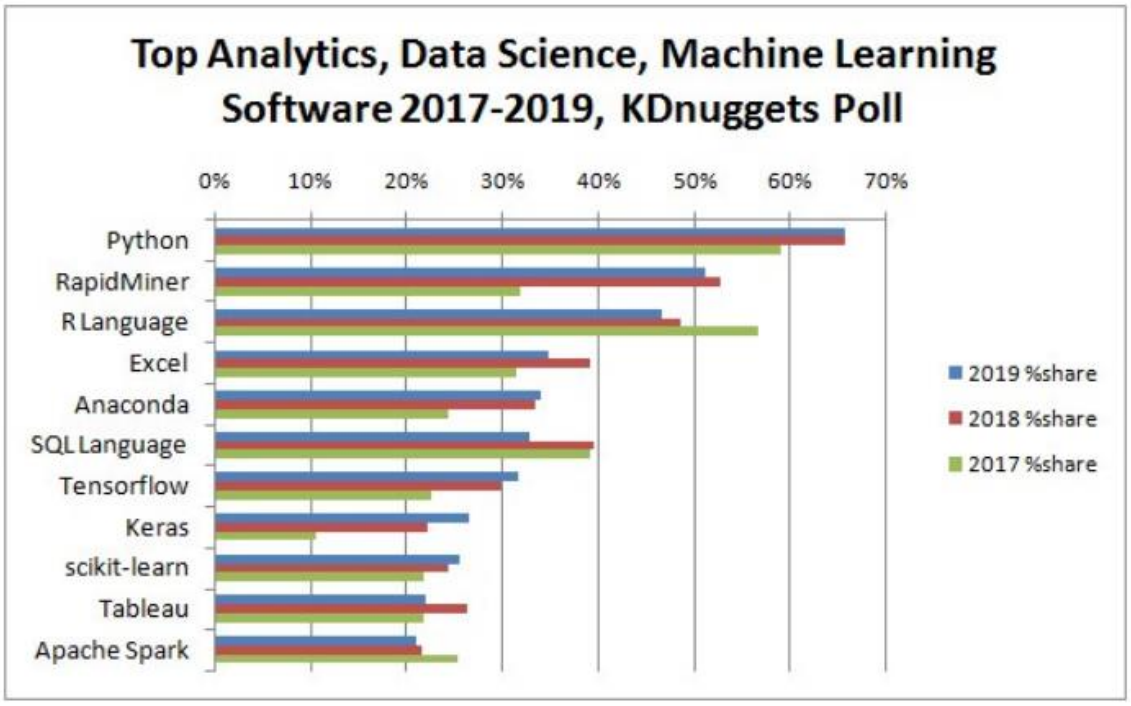


Figura 12 -Ferramentas mais adotadas em ML
Fonte: (Piatetsky, 2019)

A lista conta com *software* gratuito e pago. Para além dessa informação, entende-se também que a ferramenta mais utilizada, segundo a Figura 12, é o Python, uma vez que se encontra há 3 anos na liderança e é uma tecnologia gratuita. Em seguida, na segunda posição, surge a ferramenta RapidMiner (*software* pago) que nos anos de 2018 e 2019 conseguiu ultrapassar a linguagem R, sendo esse o antigo detentor do segundo lugar em 2017. O terceiro lugar é ocupado pela linguagem R que se encontra em declínio desde 2017, podendo-se observar que houve uma queda maior entre 2017 e 2018, tendo-se mantido a tendência de recuo em 2019 para 46,6% contra os 56,6% de 2017.

No presente trabalho, foi adotada a linguagem R. Esta é uma ferramenta utilizada em computação estatística, dispondo de um alargado reportório de *packages* que contém várias funções, algoritmos e ferramentas para visualização gráfica. A tecnologia é de *software open-source* e apresenta como principais características as suas capacidades de adaptabilidade, robustez, velocidade e confiabilidade (Kaya *et al.*, 2019) .

3.3 Métricas e avaliação de desempenho

Após estimados os modelos de classificação, é necessária a avaliação individual de cada um dos modelos estimados, a fim de que seja possível comparar e escolher o "vencedor". No entanto, esta não é uma tarefa trivial, uma vez que existem métodos diferentes de selecionar o melhor modelo.

Em problemas de classificação binária, é comum a projeção da matriz de confusão, (*confusion matrix*), *Figura 13*.

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	P'	N'	$P + N$

Figura 13 - Matriz de Confusão

Fonte: (Witten *et al.*, 2011)

A partir desta, é possível calcular inúmeras **métricas** como *accuracy*, *recall*, *precision*, *F-factor*, *sensitivity*, *specificity*, etc. Na *Figura 14*, pode-se observar o cálculo de cada uma delas. Antes de tais métricas serem analisadas, é preciso ter em consideração que os problemas de variável dependente dicotómica apresentam duas classes, conhecidas como classe positiva (+) e classe negativa (-). Em problemas de classificação de classe desbalanceada considera-se a classe majoritária como a classe negativa.

Measure	Formula
ACC	(TP + TN) / (TP + TN + FN + FP)
ERR	(FP + FN) / (TP + TN + FN + FP)
SN, TPR, REC	TP / (TP + FN)
SP	TN / (TN + FP)
FPR	FP / (TN + FP)
PREC, PPV	TP / (TP + FP)
MCC	(TP * TN—FP * FN) / ((TP + FP)(TP + FN)(TN + FP)(TN + FN))$^{1/2}$
$F_{0.5}$	1.5 * PREC * REC / (0.25 * PREC + REC)
F_1	2 * PREC * REC / (PREC + REC)
F_2	5 * PREC * REC / (4 * PREC + REC)

ACC: accuracy; ERR: error rate; SN: sensitivity; TPR: true positive rate; REC: recall; SP: specificity; FPR: false positive rate; PREC: precision; PPV: positive predictive value; MCC: Matthews correlation coefficient; F: F score; TP: true positives; TN: true negatives; FP: false positives; FN: false negatives

Figura 14 - Principais métricas de avaliação
Fonte: (Saito e Rehmsmeier, 2015)

Accuracy, ou também conhecida como ACC, é uma das métricas mais destacadas em testes de performance, sendo bastante útil em casos de dados balanceados. Tal métrica não é indicada caso a variável de interesse seja assimétrica. Como exemplo ilustrativo, considere um problema de classificação em que a variável de interesse é dicotómica tendo uma distribuição assimétrica com 99% dos dados como bombas explosivas e apenas 1% verdadeiras bombas. Caso o algoritmo classifique todas as bombas como falsas bombas, este terá uma *accuracy* de 99%. O que representa um acerto de 100% na previsão de falsas bombas e 0% no acerto de verdadeiras bombas, uma vez que não detetou qualquer verdadeira bomba. Com isso, por esta razão, a métrica não é ajustada a problemas com esta característica. São também bastante comuns as métricas *recall/sensitivity* e *precision*. Através da primeira podemos apurar entre a classe positiva qual o rácio de positivos corretamente estimados, enquanto que a métrica *precision* dá-nos a seguinte informação por um rácio, entre os que foram estimados como positivos quantos realmente eram positivos.

Análise gráfica de ROC (*Receiver Operating Characteristics*), apresentado em Spackman (1989), tem sido, cada vez mais, adotado em problemas de *machine learning* (Fawcett, 2006). Esta tem ganho popularidade em problemas destas características de classe assimétrica como também a análise gráfica *precision-recall*, surgindo como uma forma alternativa de **avaliação de desempenho** às tradicionais métricas apresentadas anteriormente.

A análise gráfica de ROC é calculada a partir da taxa de verdadeiros positivos e taxa de falsos positivos, o que resulta na curva de ROC. Com essa curva é possível calcular a área abaixo dessa curva, a qual é conhecida como métrica AUC (*Area Under the Curve*). Essa área pode variar entre 0 e 1 tendo como valor desejado o mais próximo e inclusive 1. Na *Figura 15*, pode-se observar que quanto mais próxima estiver a curva do canto superior esquerdo, maior é a sua AUC. Portanto, resulta num modelo com melhor performance, enquanto que qualquer modelo que resulte numa curva ROC abaixo da bissetriz do primeiro quadrante, apresentará um resultado inferior a um classificador aleatório.

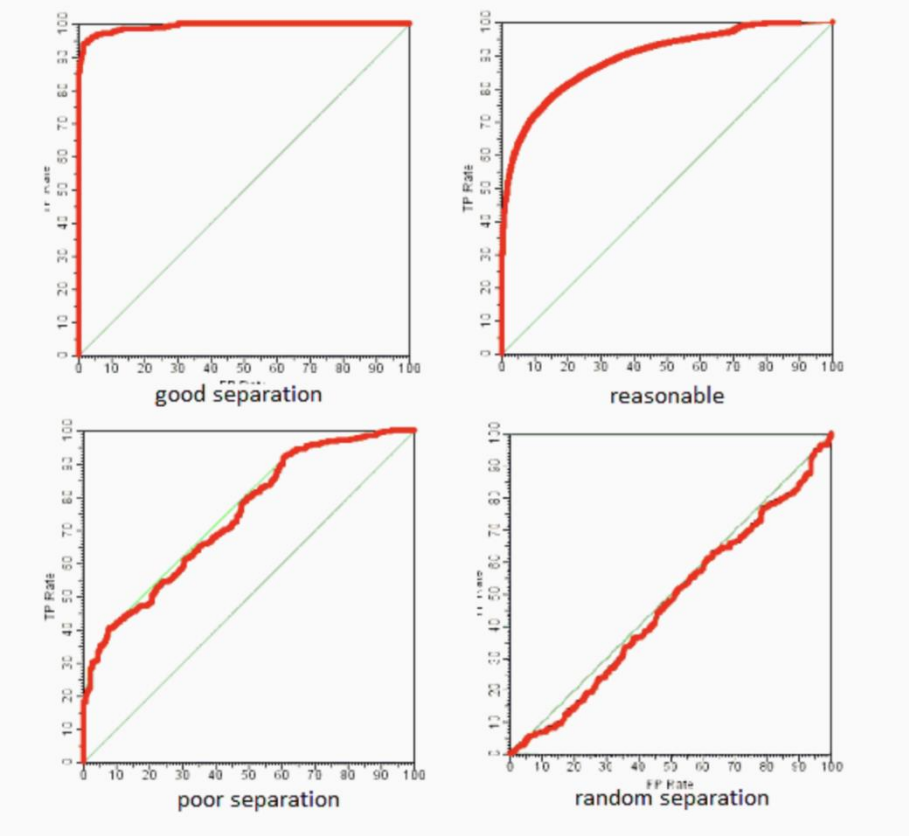


Figura 15 - Curvas de ROC
Fonte: (Flach, 2004)

Com as métricas apresentadas anteriormente, *recall* e *precision*, é também possível projetar o gráfico *precision-recall*, o qual também é recorrentemente utilizado para avaliação de performance. A partir deste, é possível eleger o classificador mais adequado. Sendo o melhor resultado esperado o de *precision* e *recall* igual a 1. Da mesma forma que se calcula a área do gráfico ROC, utiliza-se o mesmo procedimento para o gráfico de *precision-recall*, ou seja, é calculada a área abaixo da curva com a finalidade de comparar as áreas produzidas por cada classificador. Com base nos resultados apresentados em Saito e Rehmsmeier (2015), o gráfico de *precision-recall* (PR) é mais

informativo que o gráfico de ROC, em contexto de dados desbalanceados, sendo este o mais adequado para comparação de testes de *performance* entre classificadores. Segundo o estudo, os gráficos de PR são sensíveis a dados balanceados e não balanceados, o mesmo não acontece nas curvas de ROC, tal foi demonstrado por via de simulações executadas na seguinte investigação Saito e Rehmsmeier (2015). Sendo possível concluir que um modelo que apresente um AUC elevado no gráfico de ROC pode não apresentar um bom resultado AUC no gráfico PR (*Precision-Recall*).

Capítulo 4. *Machine Learning* na Prática

4.1 Aplicação da metodologia CRISP-DM

4.1.1 Compreensão do negócio

O repositório de dados utilizado, no presente trabalho, encontrava-se armazenado num SGBD Microsoft SQL Server 2014. Esse SGBD apresenta uma linguagem própria de manipulação de dados conhecida como T-SQL (*Transact-Scructured Query Language*). O uso da mesma permitiu explorar e, a partir disso, conhecer os tipos de dados existentes e também disponíveis para a elaboração do trabalho. Tal facto permitiu a possibilidade não só de identificar alguns erros de integridade, inconsistência, *missing values*, mas também valores não esperados. Contudo, esses erros foram removidos através da substituição de valores inconsistentes por valores esperados ou conhecidos. Ou em casos extremos, pela falta de informação e impossibilidade de resolução, procedeu-se à remoção das variáveis.

4.1.2 Compreensão dos dados e Preparação dos dados

Após feita uma análise exaustiva e correção de possíveis erros, foram extraídos os dados desse SGBD para um ficheiro do tipo CSV (*Comma-Separated Values*). Este documento gerado é constituído por 148431 registos e 87 variáveis, entre esses registos 147466 são pertencentes à classe negativa e apenas 965 à positiva, o que evidencia um grande desequilíbrio entre classes. A constituição do ficheiro teve como finalidade a exportação para a ferramenta R. Com o objetivo de dar continuidade à tarefa de *Data Understanding e Data Preparation*, como referido no capítulo 3, aplicando a metodologia

CRISP-DM. Estas são a segunda e terceira tarefa, tendo sido iniciadas no Microsoft SQL Server 2014 (MSSQL 2014) e tiveram a sua atividade estendida até ao *software* R. No *software* R, foram utilizadas inúmeras bibliotecas para o efeito, tendo como principais e mais utilizadas, *package* "caret", "Rattle", "e1071", "fastDummies". Foi possível através da ferramenta Rattle, a qual apresenta uma interface gráfica o que permite de forma rápida e *user-friendly*, obter estatísticas resumidas dos dados, como a média, mediana, moda, mínimo, máximo. Para além disso, a ferramenta possibilita, de forma intuitiva, a projeção de gráficos tais como: caixa de bigodes, histogramas, matriz de correlações. De notar também a possibilidade de criação e transformação de variáveis, como a tarefa de normalizar; conversão *one-hot encoding* e a conversão do tipo de variáveis. Na *Figura 16*, pode-se observar a interface gráfica do *Rattle*, as várias *tabs* que o constituem, dividindo o conjunto de etapas executadas durante a realização do projeto.

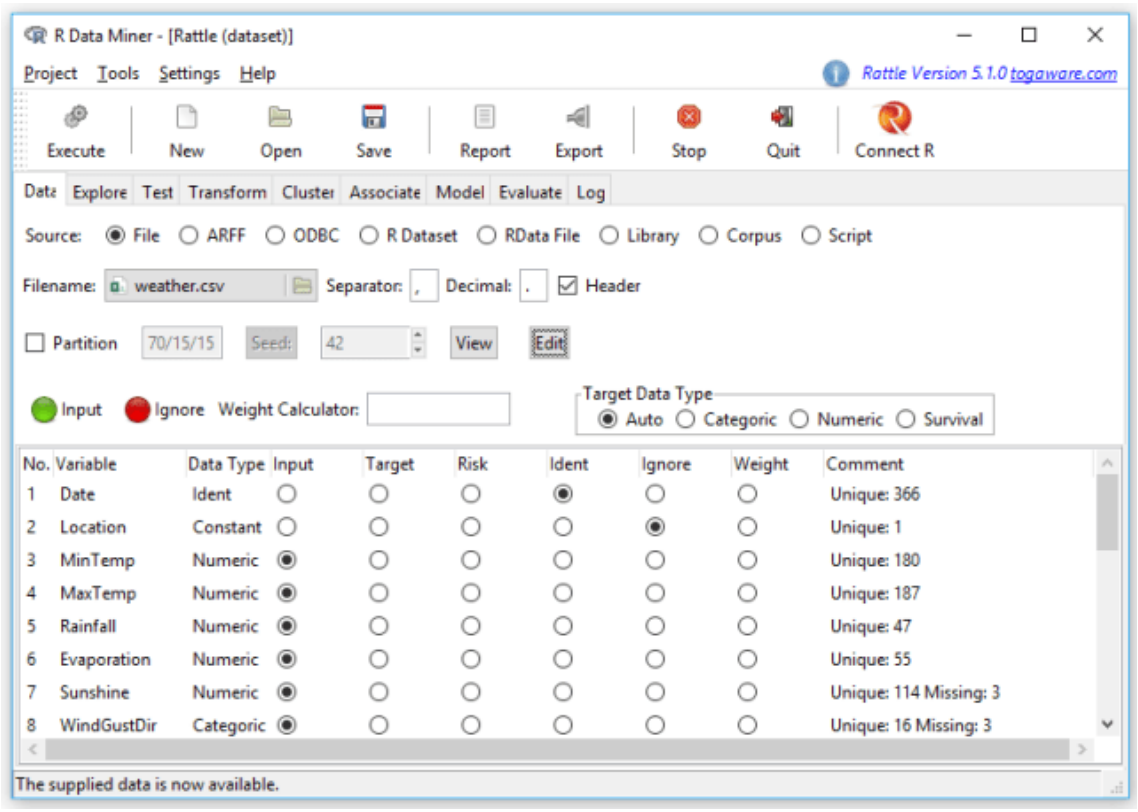


Figura 16 - Interface Rattle
Fonte: (Muenchen, 2018)

Concluída a segunda e terceira tarefa, na interface *Rattle*, deu-se início a uma das etapas mais críticas e morosas de um projeto com estas características, conforme enunciado no capítulo 3, o fenómeno de *curse of dimensionality*. A literatura conta com várias possíveis soluções para este tipo de acontecimento, sendo esta uma das fases mais

importantes: a seleção de variáveis relevantes, para tal tarefa contamos com distintas abordagens, como *embedded*, filtro e *wrapper*. Analisar-se-á cada uma.

A primeira, *embedded*, está presente no próprio classificador, tal é comum em árvores de decisão em que o próprio modelo realiza a seleção das variáveis conforme abordado anteriormente no capítulo 2. Contudo, nem todos os classificadores possuem a seleção automática e, mesmo em árvores de decisão, é importante que exista alguma pré-seleção, visto que isso poderá resultar em grandes ganhos computacionais, traduzindo-se numa execução muito mais rápida do próprio algoritmo.

Para além desta, existe a abordagem filtro. Esta segunda é realizada antes da seleção do classificador, sendo totalmente independente do mesmo, com a finalidade de remover variáveis irrelevantes e redundantes. Têm-se como principais algoritmos nesta abordagem: o teste *chi-square*, distância euclidiana, *information gain* e Relief-F (Kira e Rendell, 1992) . A aplicação de filtro apresenta como principais vantagens: rapidez, escalabilidade e independência dos classificadores (Kumari e Swarnkar, 2011)**.**

Por último, a abordagem *wrapper*. Esta diferencia-se da abordagem filtro, uma vez que dependerá do classificador. O algoritmo *wrapper* testa vários subconjuntos de variáveis num determinado classificador, com a finalidade de obter o subconjunto que consiga melhor desempenho entre os vários testados. Tal tarefa é executada até que algum critério de paragem definido a priori seja satisfeito, existindo também formas distintas de formar os subconjuntos de variáveis. Como principais exemplos temos a estratégia *Backward, Forward, Hill-climbing*, algoritmos genéticos (*GA – Genetic Algorithm*). O esquema do algoritmo pode ser observado na *Figura 17*.

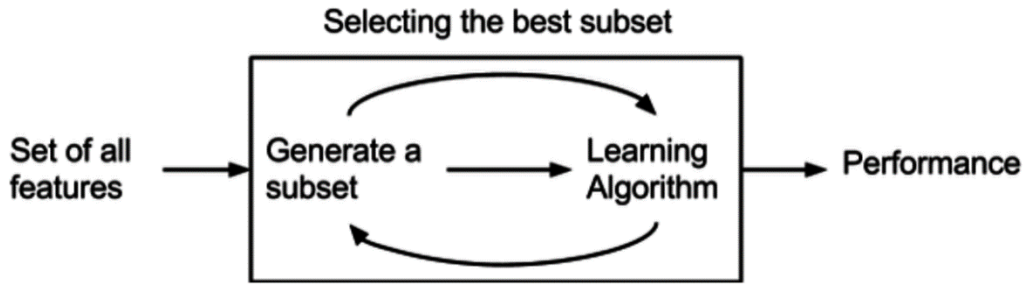


Figura 17 - Algoritmo Wrapper
Fonte: (Mayo, 2018)

Um dos principais problemas que pode ser posto à abordagem é o alto custo computacional, podendo ter uma duração excessiva como demonstrado no estudo publicado em Lee *et al.* (1999). A aplicação de *wrapper* pode demorar 3079 vezes mais

que o uso de filtro. Tal se sucedeu nesta publicação Lee *et al.* (1999), em que foram realizadas várias simulações com abordagens diferentes em diversos *datasets*. A aplicação de *wrapper*, no *dataset* utilizado nesta pesquisa, não obteve resultados num tempo inferior a 1 semana, por essa razão esta foi abandonada, sendo apenas utilizado a abordagem filtro e *embedded*. E também como demonstrado em Lee *et al.*(1999), a abordagem filtro além de ser bastante mais rápida, conseguiu obter resultados bastante próximos da aplicação *wrapper*.

O algoritmo *Relief-F* apresentado em Kononenko (1994) surge como mais um algoritmo entre os vários existentes na abordagem filtro. Tem sido amplamente adotado pela comunidade científica, aparecendo em inúmeros trabalhos publicados. O algoritmo foi concebido para problemas de *feature selection*, mas atualmente é também utilizado para outros fins, como heurísticas para indução de regras de associação, discretização, critério de indução de arvores de decisão (Demšar, 2010). O algoritmo caracteriza-se por ter em conta a interação entre atributos, sendo assim adequado e possível a captação das suas dependências. Para além disso, este também se caracteriza por recorrer a um classificador conhecido como *lazy algorithm*, ou também como vizinho mais próximo (*Nearest Neighbor)*, que é frequentemente utilizado como classificador em problemas de aprendizagem supervisionado e não supervisionado.

O algoritmo, no *software* R, encontra-se implementado na biblioteca "CORElearn". Para a sua utilização apenas é necessário carregar esta biblioteca e executar a função *relief* sobre o *dataset*.

Após aplicada a função *relief* e a partir desta biblioteca foi possível observar o resultado na Figura 18. O seguinte histograma encontra-se ordenado por ordem decrescente, conforme o peso de importância de cada uma das variáveis. No presente trabalho, foram extraídas 20 variáveis dependentes de um total de 86, o que possibilitou uma redução do espaço de variáveis dependentes a aplicar na etapa de modelação.

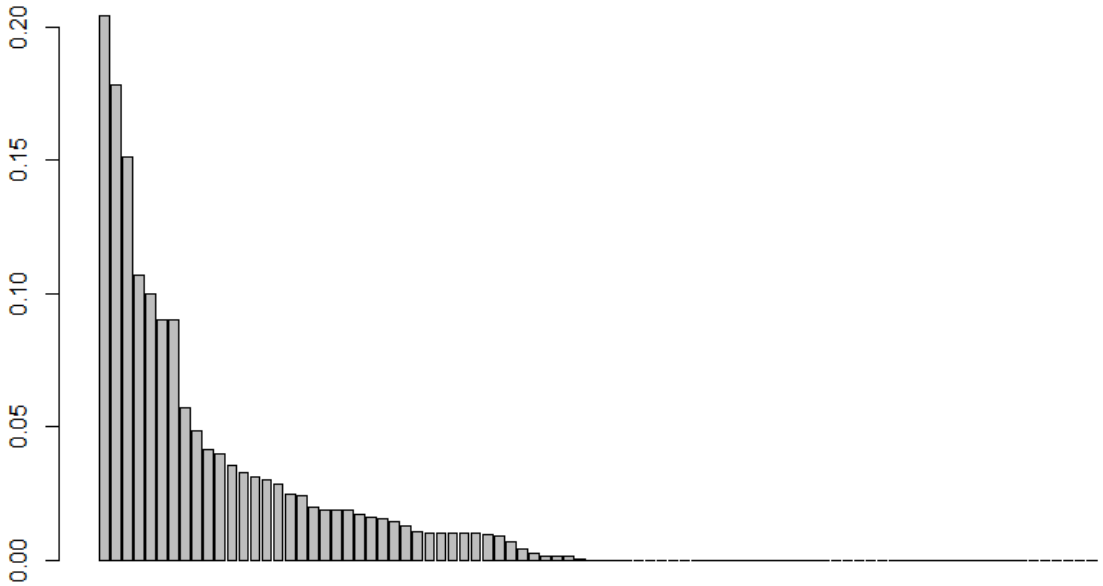


Figura 18- Histograma Relief
Fonte: Autor

4.1.3 Modelação

Nesta etapa, foram aplicados os classificadores propostos sobre o conjunto de dados disponíveis e tratados até à presente fase. Entretanto, ainda antes dessa mesma implementação, o conjunto de dados deve ser separado em dois: conjunto de treino e conjunto de teste. Dessa forma, será possível testar efetivamente o desempenho do modelo, evitando assim a fraca generalização. Tal problema surge quando o classificador tem um alto desempenho sobre os dados utilizados para treinar, porém possui uma fraca capacidade de desempenho sobre novos dados. Com o intuito de colmatar a fraca generalização, são sugeridos na literatura diversos métodos de amostragem a aplicar sobre o conjunto de dados, a título de exemplo as seguintes técnicas: *K-folds*, *Holdout* e *Leave-one-hot*.

A técnica *K-folds* tem-se mostrado uma das mais adotadas em diversos trabalhos apresentados na área, uma vez que aumenta a precisão do classificador (Kim, 2009).

A mesma é conhecida por *K-folds* por ser uma adaptação da técnica conhecida como a mais simples, *Holdout*. Esta simplesmente divide o conjunto de dados em duas partições: partição de teste e de treino, o que equivale a 1-*Fold*, *Figura 19*.

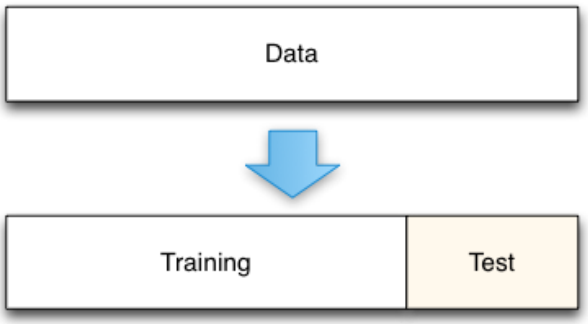


Figura 19 - Exemplo *Holdout*
Fonte: (Hopkins, 2019)

Normalmente, a partição de teste varia entre 20% e 35% do conjunto de dados, já os restantes dados são utilizados no conjunto de treino. No entanto, a técnica K-folds consiste na divisão do conjunto de dados em K partições (*folds)* de tamanhos aproximadamente iguais. A técnica aplica K-1 partições no conjunto de treino, enquanto que os restantes dados serão utilizados para teste. Este procedimento repete-se K vezes, em que cada uma das K partições é utilizada uma vez para teste, ao mesmo tempo que os restantes dados são utilizados para treino. Tal técnica utilizará todos os dados no momento de treinar o classificador. Tal não acontece na técnica *Holdout*, visto que não inclui as observações utilizadas para teste. Na *Figura 20*, pode-se observar um exemplo da técnica K-*fold*, com K = 4.

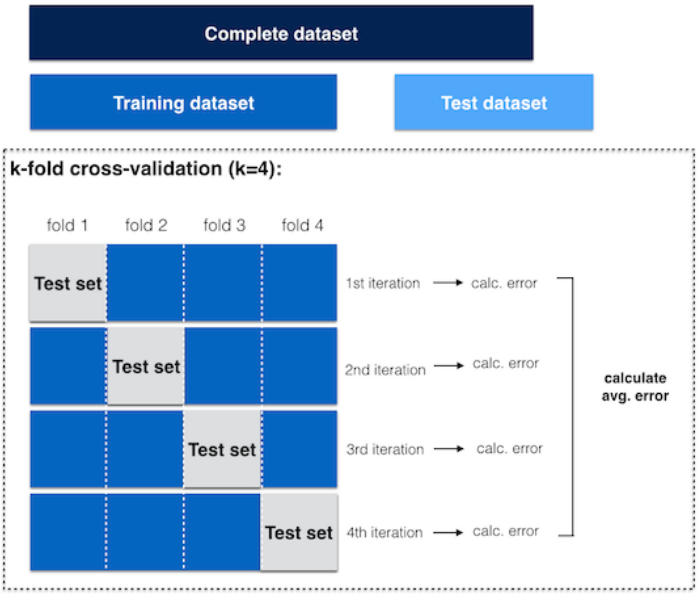


Figura 20- Ilustração K-fold com K=4
Fonte: (Raschka, 2014)

Sendo o valor de K um parâmetro necessário a definir, que usualmente se compreende entre 1 a 10, é bastante comum a adoção de K=2, 5 ou 10. Não é recomendável um valor para K elevado, particularmente em casos de conjuntos de dados caracterizados por uma dimensão pequena. Tal teria como consequência um conjunto de testes com pouca ou nenhuma representatividade, o que não é recomendável em problemas de classe desbalanceada. Observe-se o seguinte exemplo: um conjunto desbalanceado com 1400 observações, em que 1385 instâncias pertencem à classe negativa e apenas 15 à classe positiva. Caso sejam realizadas 20 partições, ter-se-ia, na melhor das hipóteses, 15 partições em que cada uma delas contaria com 1 observação positiva e as restantes 5 divisões sem qualquer dado de classe positiva, o que demonstra a problemática de atribuir a K um valor elevado. Algumas pesquisas, por via de simulações, têm demonstrado que K=10, geralmente, é um ótimo candidato (Rodriguez *et al.*, 2010; Moreira *et al.*, 2018).

Contudo, após feita a divisão do conjunto de dados não ocorreu de imediato a modelação. Foi necessário, com o intuito de melhorar a precisão dos classificadores, a aplicação de técnicas de amostragem, uma vez que a classe positiva conta com poucas observações. Para resolver a baixa representatividade da classe positiva, têm sido propostas técnicas de *undersampling* e de *oversampling*. Na primeira técnica, *undersampling*, é realizada uma redução do número de observações da classe majoritária, enquanto que na segunda técnica é realizada uma criação artificial de observações na classe minoritária. Existindo várias formas de realizar *oversampling* e *undersampling*, neste TFM foi adotada a combinação de *Smote* (*Synthetic Minority Over-sampling Technique)* e *Tomek*. Com base na investigação Batista *et al.* (2004), a junção desses dois algoritmos demonstrou apresentar melhores resultados, quando comparado com as restantes técnicas aplicadas nessa simulação sobre dados desbalanceados. Ambas as técnicas, *Smote* e *Tomek*, podem ser consultadas com mais detalhes nas seguintes publicações Chawla *et al.* (2002) e Tomek (1976), respetivamente. A aplicação da técnica *Smote* teve como finalidade a geração de observações artificiais da classe minoritária, enquanto que a técnica de *Tomek* foi utilizada com o objetivo de remover observações semelhantes com classes dependentes distintas. A título de exemplo, na *Figura 21*,

surgem novas observações representadas de amarelo no gráfico que ocupa o lado direito, sendo o resultado da aplicação da técnica de *oversampling* SMOTE.

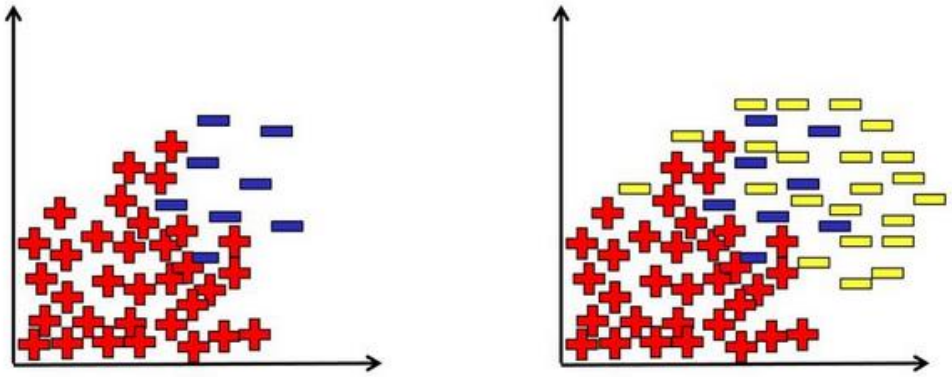


Figura 21- Smote
Fonte: (Fernández, 2018)

É possível observar que surgem classes distintas muito próximas, as quais são consideradas ruídos, o que poderá implicar perda de desempenho na modelação. Como forma de colmatar esse facto, é sugerida a aplicação da técnica *Tomek*. Esta técnica tem como finalidade a remoção dessas observações, conforme pode ser observado na *Figura 22*.

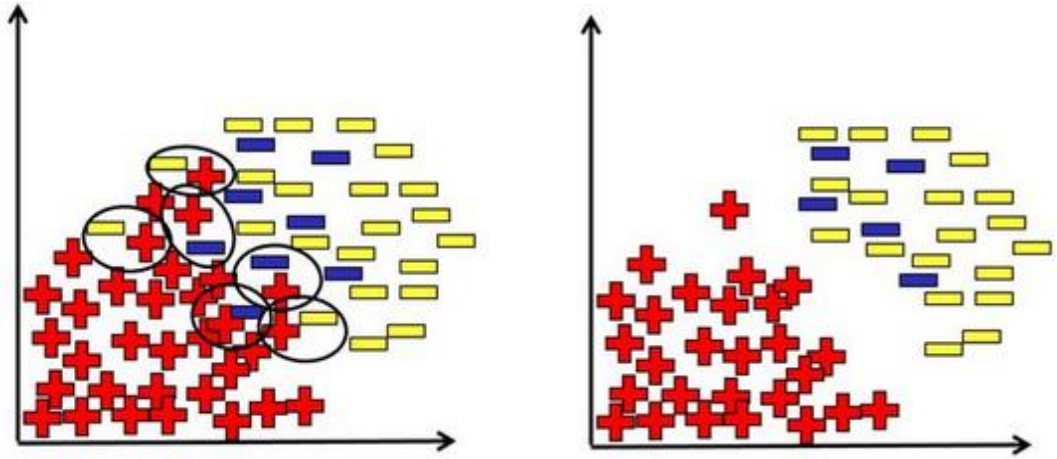


Figura 22- Smote + Tomek
Fonte: (Fernández, 2018)

No *software* R, a biblioteca "UnBalanced" conta com várias técnicas úteis para a resolução do problema apresentado, tendo ela sido utilizada para a aplicação da combinação SMOTE e TOMEK.

Após realizadas todas as tarefas que englobam as atividades de preparação e lapidação dos dados disponíveis, segue-se a fase de modelação. Conforme a metodologia CRISP-DM, dar-se-á o início ao quarto estágio. Nesta etapa, procedeu-se ao treino dos seguintes classificadores: *Adaboost*, *Random Forest* (RF) e *Boosting* C5.0.

A biblioteca do R conta com vários packages que integram tais classificadores. Segundo dados publicados na comunidade KDnuggets Peddibhotla (2015), as principais bibliotecas mais descarregadas para esse fim são as seguintes: "e1071", "rpart", "nnet", "randomForest" e "caret". Esse resultado pode ser observado na Figura 23.

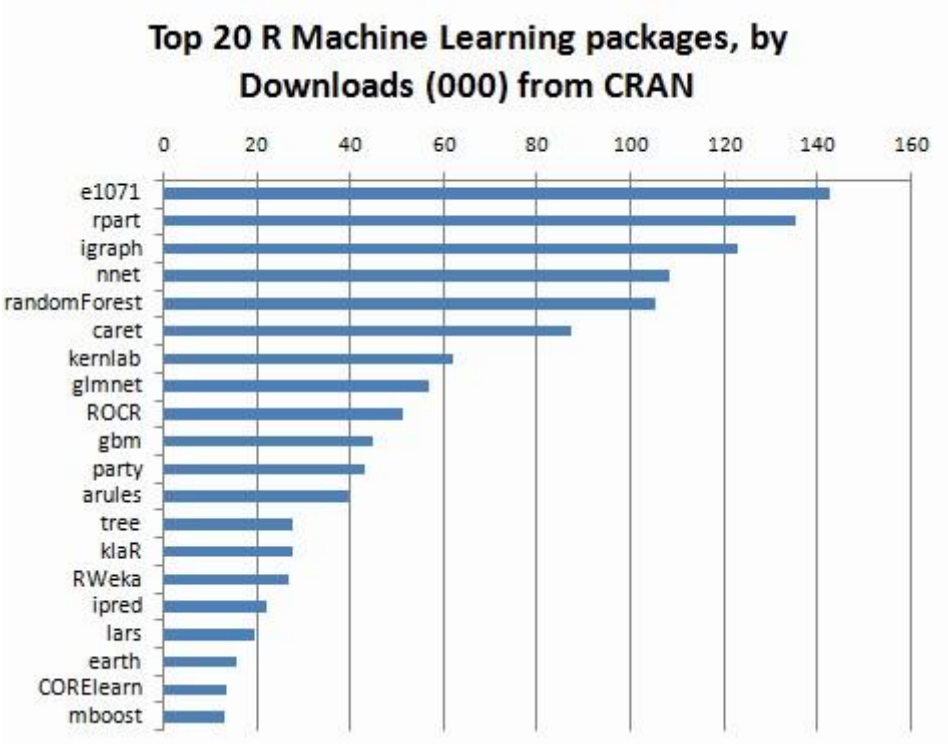


Figura 23 - Bibliotecas mais descarregas no *software* R
Fonte: (Peddibhotla, 2015)

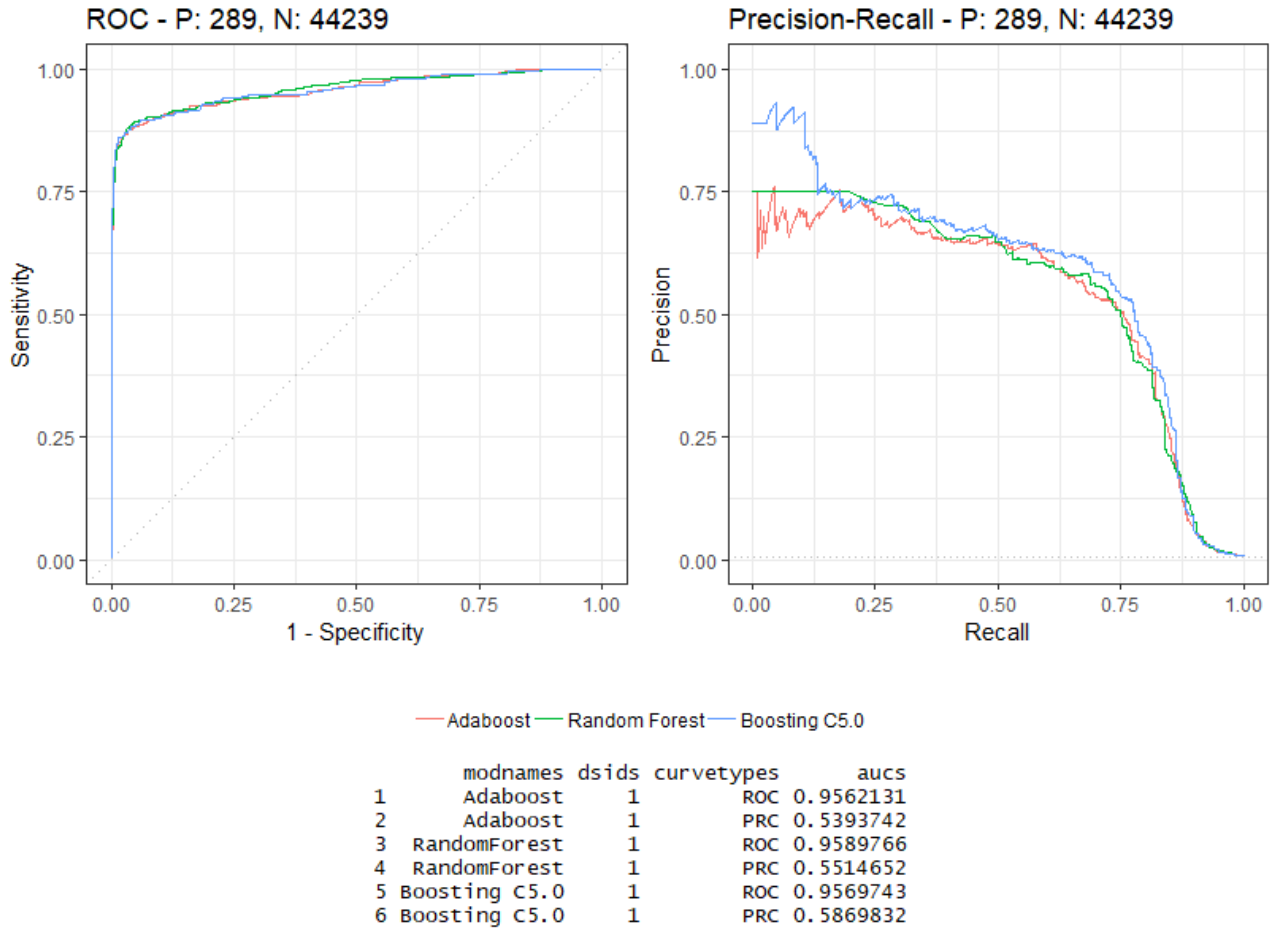
Depois de realizada uma análise exaustiva de cada uma das principais bibliotecas, foram adotadas as seguintes: "randomForest", "caret" e "fastAdaboost". Esta última, mesmo não constando na lista mencionada anteriormente, foi elegida por apresentar melhores resultados ao nível do tempo de execução. Consoante os próprios nomes sugerem, nas bibliotecas "randomForest" e "fastAdaboost" apenas se estimam os classificadores *Random Forest* e *Adaboost*, respetivamente. O que não sucede com a *package "*caret". Esta agrega diversos classificadores incluindo os já mencionados, o seu manual de utilização encontra-se publicado em Kuhn (2018). No entanto, os dois classificadores não foram executados a partir desta biblioteca, uma vez que a aplicação destes algoritmos pela

biblioteca "caret" originou tempos de execução muito longos. Isto é, a *package* "caret" foi somente aplicada na estimação do modelo *Boosting* C5.0.

Na estimação dos classificadores supramencionados, foi aplicada o método *K-folds* com K = 10, com a seguinte divisão dos dados: 70% para treino e 30% para teste.

4.1.4 Avaliação

Terminado o quarto estágio, fase em que foram treinados diversos classificadores, segue-se o momento de comparar e avaliar cada classificador gerado. Como tratado no capítulo 3.3 Métricas e avaliação de desempenho, é possível comparar o desempenho entre diversos classificadores pela representação gráfica das Curvas de ROC e pelo gráfico *Precision-Recall*. Nesse mesmo capítulo foi referenciado que, dada natureza do problema tratado neste trabalho, a escolha do melhor modelo acontecerá pela observação do gráfico *Precision-Recall*. A biblioteca utilizada para este fim foi a "precrec", o resultado obtido a partir da mesma pode ser observado na Figura 24. No lado esquerdo, apresenta-se o gráfico com as curvas de ROC de cada um dos 3 modelos estimados. Surgindo ainda do lado direito o gráfico *Precision-Recall* com a representação de cada uma das 3 curvas. Como se pode constatar no gráfico do lado esquerdo, as 3 curvas encontram-se sempre muito próximas, não sendo claro visualmente qual a curva com maior AUC. No entanto, é possível obter essa resposta observando a tabela que apresenta a área calculada de cada uma das curvas. Analisando o gráfico *Precision-Recall* é mais evidente e dessa forma possível constatar visualmente que o classificador *Boosting* C5.0 apresenta uma área maior abaixo da sua curva, comparativamente com os demais classificadores, possuindo uma área de aproximadamente 0,58. Ainda com base nos valores obtidos pela tabela da Figura 24, é possível notar que o modelo que apresenta maior área nas curvas de ROC não coincide com o modelo com maior área em *Precision-Recall*. Não sendo esta uma surpresa como mencionado na secção 3.3 Métricas e avaliação de desempenho .



	modnames	dsids	curvetypes	aucs
1	Adaboost	1	ROC	0.9562131
2	Adaboost	1	PRC	0.5393742
3	RandomForest	1	ROC	0.9589766
4	RandomForest	1	PRC	0.5514652
5	Boosting C5.0	1	ROC	0.9569743
6	Boosting C5.0	1	PRC	0.5869832

Figura 24- Curvas de ROC e gráfico *Precision-Recall*
Fonte: Autor

Continuando a análise ao modelo vencedor, observe-se na Figura 25 as principais métricas calculadas a partir da matriz de confusão:

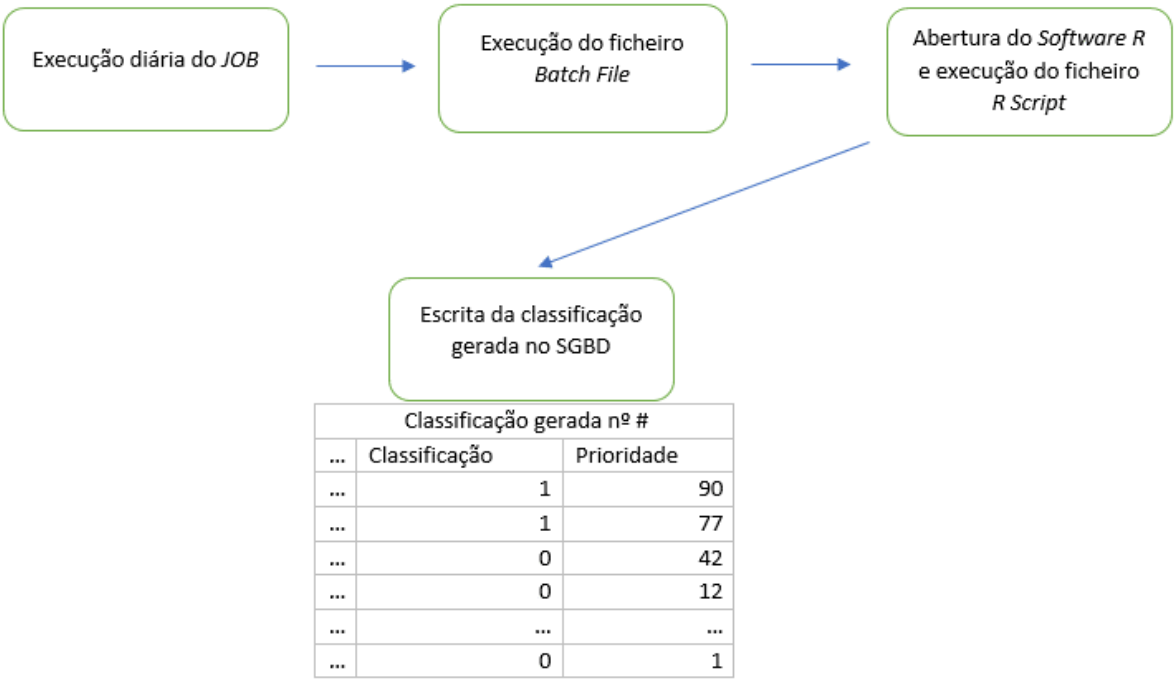
	BOOSTING C5.0	**RANDOM FOREST**	**ADABOOST**
ACCURACY	0.9903	0.9892	0.9908
SENSITIVITY	0.8304	0.8201	0.8131
SPECIFICITY	0.9914	0.9903	0.9919
BALANCED ACCURACY	0.9109	0.9052	0.9025

Figura 25- Métricas obtidas nos classificadores *Boosting* C5.0, *Random Forest* e *Adaboost*
Fonte: Autor

A partir da métrica *balanced accuracy*, é possível concluir que tal classificador, no conjunto de testes, obteve um acerto médio de aproximadamente 91% entre as duas classes. Os resultados conduziram a um acerto de 99,14% das observações pertencentes à classe negativa e um acerto de 83,04% das observações que pertencem à classe positiva. Ou seja, das observações que estavam classificadas como prioritárias e que constituem o conjunto de dados para teste, o classificador *Boosting* C5.0 conseguiu identificar corretamente 83,04% dessas observações.

4.1.5 Execução

Iniciando a última etapa, a de execução, após realizada a escolha do modelo vencedor, procedeu-se à automatização do processo de classificação sobre novos dados. Com base nessa necessidade foi criado um *R script* com todo o conjunto de comandos necessários para obter as novas observações geradas no SGBD. Por conseguinte, possibilitando classificações sobre esses dados e, após isso, a escrita no SGBD de cada classificação em cada nova instância. Concluído esse ficheiro *R script*, dada a necessidade de garantir que esse é executado diariamente, procedeu-se à geração de um arquivo do tipo *Batch File*. Esse executa o comando de abertura automática do R e leitura do *R script*. Por fim, foi gerado um *Job* no SGBD com periocidade diária que terá como finalidade a execução do *Batch File*, conforme demonstrado na Figura 26.



Classificação gerada nº #		
...	Classificação	Prioridade
...	1	90
...	1	77
...	0	42
...	0	12
...	...	...
...	0	1

Figura 26 - Esquema do *Job* diário implementado no SGBD
Fonte: Autor

Capítulo 5. Conclusões e propostas para futuros trabalhos

O emprego das técnicas de *Machine Learning* tem-se demonstrado útil no suporte à tomada de decisão. Tal foi possível pela exploração de dados que se encontravam armazenados num SGBD. Porém, esses dados apenas representavam volume nos discos rígidos.

Com a presente investigação, foi possível extrair valor desse volume de dados ao gerar um modelo que possibilita a classificação de novas instâncias. Tal classificador é executado diariamente e de forma automática. Consequentemente, permite qualificar cada novo evento como prioritário ou não prioritário, assim como quantificar a sua prioridade, possibilitando a ordenação de cada nova observação conforme o seu nível de importância. O que resulta em ganhos de tempo para os utilizadores, que antes teriam como principal tarefa a exploração e devida classificação para cada novo evento de forma manual e sem qualquer critério de fila. Ou seja, anteriormente o profissional teria de analisar cada evento sem que conhecesse previamente a prioridade. Agora, é possível entender quais os eventos que terão de ser averiguados primeiro face aos restantes, uma vez que ocorre uma classificação.

É importante destacar os resultados obtidos na avaliação dos classificadores. Foi essencial a projeção do gráfico *precision-recall*, uma vez que as curvas ROC não apresentaram diferenças visualmente percetíveis, não sendo suficientemente informativas, o que permitiu confirmar que este gráfico não é o mais adequado à problemática. Pode-se ainda concluir que o classificador *Boosting* C5.0 apresentou resultados superiores ao algoritmo *Random Forest* e *Adaboost* quando realizada a interpretação do gráfico *precision-recall*. De notar que os resultados entre as suas representações gráficas não convergiram e, ao analisar o gráfico das curvas de ROC, o algoritmo *Random Forest* apresentou um AUC superior ao do algoritmo *Boosting* C5.0, tal não aconteceu na representação gráfica *precision-recall.*

Para trabalhos futuros, ainda no âmbito das técnicas de *Machine Learning*, sugere-se a adoção de uma das mais renomadas tecnologias: o *RapidMiner*. A mesma tem ganho grande popularidade pela sua *interface user-friendly*, não exigindo conhecimentos de programação a quem procura realizar um trabalho de ML. Tal ferramenta possibilita a execução de outros *software* como R e Python. Outros classificadores também são sugeridos tais como *Support Vector Machine* e *Neural Network*. Com o objetivo de

procurar potenciar os desempenhos que foram apresentados. Ainda no que concerne ao combate ao Branqueamento de Capitais e ao Financiamento do Terrorismo, para abordagens futuras, seria importante aplicação de modelos descritivos tais como Regras de associação e *Clustering*.

Referências Bibliográficas

Altmbjjan, E. I. *et al.* (1994) «Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience)», *Journal of Banking and Finance*, 18(3), pp. 505–529. doi: 10.1016/0378-4266(94)90007-8.

Aziz, S. *et al.* (2019) «Machine Learning in Finance: A Topic Modeling Approach», *SSRN Electronic Journal*. doi: 10.2139/ssrn.3327277.

Baranauskas, J. A. *et al.* (2018) «A tree-based algorithm for attribute selection», *Applied Intelligence*, 48(4), pp. 821–833. doi: 10.1007/s10489-017-1008-y.

Batista, G. E. A. P. A. *et al.* (2004) «A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data», *SIGKDD Explor. Newsl.* New York, NY, USA: ACM, 6(1), pp. 20–29. doi: 10.1145/1007730.1007735.

Bolton, Richard J.; Hand, David J. Statistical Fraud Detection: A Review. Statist. Sci. 17 (2002), no. 3, 235--255. doi:10.1214/ss/1042727940. https://projecteuclid.org/euclid.ss/1042727940

Boonkwang, K. *et al.* (2018) «A Comparison of Data Mining Techniques for Suicide Attempt Characteristics Mapping and Prediction», em *2018 International Seminar on Application for Technology of Information and Communication*. IEEE, pp. 488–493. doi: 10.1109/ISEMANTIC.2018.8549835.

Breiman, L. *et al.* (1984) *Classification and regression trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software (The Wadsworth statistics/probability series). Disponível em: https://cds.cern.ch/record/2253780.

Breiman, L. (1996) «Bagging Predictors», *Machine Learning*, 24(2), pp. 123–140. doi: 10.1023/A:1018054314350.

Breiman, L. (2001) «Random Forests», *Mach. Learn.* Norwell, MA, USA: Kluwer Academic Publishers, 45(1), pp. 5–32. doi: 10.1023/A:1010933404324.

Breiman, L. (2001) «Statistical Modeling : The Two Cultures», Statistical Science, 16(3),

pp. 199–215. doi:10.1214/ss/1009213726.

Brynjolfsson, E. *et al.* (2011) «Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance?», *Ssrn*. doi: 10.2139/ssrn.1819486.

de Castro, L. N. e Ferrari, D. G. (2017) *Introdução a mineração de dados*. Editora Saraiva.

Chawla, N. V. *et al.* (2002) «SMOTE: Synthetic Minority Over-sampling Technique», *Journal of Artificial Intelligence Research*, 16, pp. 321–357. doi: 10.1613/jair.953.

Coenen, F. (2011) «Data mining: Past, present and future», *Knowledge Engineering Review*, 26(1), pp. 25–29. doi: 10.1017/S0269888910000378.

Demšar, J. (2010) «Algorithms for subsetting attribute values with Relief», *Machine Learning*, 78(3), pp. 421–428. doi: 10.1007/s10994-009-5164-0.

Fawcett, T. (2006) «An Introduction to ROC Analysis», *Pattern Recogn. Lett.* New York, NY, USA: Elsevier Science Inc., 27(8), pp. 861–874. doi: 10.1016/j.patrec.2005.10.010.

Fayyad, U. *et al.* (1996) «From Data Mining to Knowledge Discovery in Databases», *AI Magazine*, 17(3 SE-Articles), pp. 37-54. doi: 10.1609/aimag.v17i3.1230.

Fernández, G. (2018) *Balancing Techniques Gretel Fernández. - ppt download*. Disponível em: https://slideplayer.com/slide/13128031/ (Acedido: 2 de Outubro de 2019).

Flach, P. (2004) *The many faces of ROC analysisin machine learning*, ICML Tutorial.

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

Freitas, A. (sem data) *Árvores de Decisão*, *IST*. Disponível em: http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexf23d.html?id=199 (Acedido: 7 de Outubro de 2019).

Freund, Y. e Schapire, R. E. (1996) «Experiments with a New Boosting Algorithm», em

Saitta, L. (ed.) *Machine Learning, Proceedings of the Thirteenth International Conference {(ICML}'96), Bari, Italy, July 3-6, 1996*. Morgan Kaufmann, pp. 148–156.

Freund, Y. e Schapire, R. E. (1997) «A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting», *Journal of Computer and System Sciences*, 55(1), pp. 119–139. doi: 10.1006/jcss.1997.1504.

Gama, J. *et al.* (2011) *Inteligência artificial: uma abordagem de aprendizado de máquina*. Grupo Gen - LTC.

Gironés Roig, J. *et al.* (2017) *Minería de datos. Modelos y algoritmos*. Edição: 1. Editado por S. L. Editorial UOC.

Han, J. *et al.* (2012) «9 - Classification: Advanced Methods», em Han, J. et al. (eds.) *The Morgan Kaufmann Series in Data Management Systems*. Boston: Morgan Kaufmann, pp. 393–442. doi: https://doi.org/10.1016/B978-0-12-381479-1.00009-5.

Hopkins, J. (2019) *Cross-Validation — Machine-Learning-Course 1.0 documentation*. Disponível em: https://machine-learning-course.readthedocs.io/en/latest/content/overview/crossvalidation.html (Acedido: 29 de Setembro de 2019).

Jansson, J. (2016) *Decision Tree Classification od Products Using C5.0 and Prediction of Workload Using Time Series Analysis*. KTH, School of Electrical Engineering (EES).

Kaya, E. *et al.* (2019) «Spatial data analysis with R programming for environment», *Human and Ecological Risk Assessment: An International Journal*. Taylor & Francis, 25(6), pp. 1521–1530. doi: 10.1080/10807039.2018.1470896.

Kim, J.-H. (2009) «Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap», *Computational Statistics & Data Analysis*, 53(11), pp. 3735–3745. doi: 10.1016/j.csda.2009.04.009.

Kira, K. e Rendell, L. A. (1992) «A Practical Approach to Feature Selection», em *Proceedings of the Ninth International Workshop on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (ML92), pp. 249–256. Disponível em:

http://dl.acm.org/citation.cfm?id=141975.142034.

Kononenko, I. (1994) «Estimating attributes: Analysis and extensions of RELIEF BT - Machine Learning: ECML-94», em Bergadano, F. e De Raedt, L. (eds.). Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 171–182.

Krawczyk, B. (2016) «Learning from imbalanced data: open challenges and future directions», *Progress in Artificial Intelligence*, 5(4), pp. 221–232. doi: 10.1007/s13748-016-0094-0.

Kuhn, M. (2018) «caret: Classification and Regression Training». Disponível em: https://cran.r-project.org/package=caret (Acedido: 27 de Setembro de 2019).

Kuhn, M. e Johnson, K. (2013) *Applied Predictive Modeling*. Springer New York (SpringerLink : Bücher).

Kumari, B. e Swarnkar, T. (2011) «Filter versus wrapper feature subset selection in large dimensionality micro array: A review», *International Journal of Computer Science and Information Technologies*, 2, pp. 1048–1053.

Lee, H. *et al.* (1999) «Empirical Comparison of Wrapper and Filter Approaches for Feature Subset Selection» ICMC-USP.

Levy, S. (2010) *The AI Revolution is On*, *Wired*. Disponível em: https://www.wired.com/2010/12/ff-ai-essay-airevolution (Acedido: 20 de Maio de 2019).

Lígia Simões (2017) *Portugal entre os melhores no combate ao branqueamento de capitais – O Jornal Económico*, *Jornal Económico*. Disponível em: https://jornaleconomico.sapo.pt/noticias/portugal-entre-os-melhores-no-combate-ao-branqueamento-de-capitais-229382 (Acedido: 18 de Abril de 2019).

Maclin, R. e Opitz, D. W. (2011) «Popular Ensemble Methods: An Empirical Study», *CoRR*, abs/1106.0. Disponível em: http://arxiv.org/abs/1106.0257.

Marquesone, R. (2016) *Big Data: Técnicas e tecnologias para extração de valor dos*

dados. Casa do Código.

Marr, B. (2016) *A Short History of Machine Learning -- Every Manager Should Read*, *Forbes*. Disponível em: https://www.forbes.com/sites/bernardmarr/2016/02/19/a-short-history-of-machine-learning-every-manager-should-read/#259317d315e7 (Acedido: 14 de Maio de 2019).

Mayo, M. (2018) *Step Forward Feature Selection: A Practical Example in Python*, *KDnuggets*. Disponível em: https://www.kdnuggets.com/2018/06/step-forward-feature-selection-python.html (Acedido: 18 de Setembro de 2019).

Moreira, J. *et al.* (2018) *A General Introduction to Data Analytics*, Wiley. doi: 10.1002/9781119296294.

Muenchen, B. (2018) *A Comparative Review of the Rattle GUI for R | R-bloggers*, *R-bloggers*. Disponível em: https://www.r-bloggers.com/a-comparative-review-of-the-rattle-gui-for-r/ (Acedido: 18 de Setembro de 2019).

Patel, H. *et al.* (2019) «An Application of Ensemble Random Forest Classifier for Detecting Financial Statement Manipulation of Indian Listed Companies BT - Recent Developments in Machine Learning and Data Analytics», em Kalita, J. et al. (eds.). Singapore: Springer Singapore, pp. 349–360.

Peddibhotla, G. (2015) *Top 20 R Machine Learning and Data Science packages*, *KDnuggets*. Disponível em: https://www.kdnuggets.com/2015/06/top-20-r-machine-learning-packages.html (Acedido: 20 de Setembro de 2019).

Piatetsky, G. (2019) *Python leads the 11 top Data Science, Machine Learning platforms: Trends and Analysis*, *Kdnuggets*. Disponível em: https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html (Acedido: 18 de Julho de 2019).

Quidgest (2019) *Quidgest - Formação*, *Quidgest*. Disponível em: https://www.quidgest.pt/cert_Genio.asp?LT=PTG (Acedido: 21 de Maio de 2019).

Quinlan, J. R. (1986) «Induction of decision trees», *Machine Learning*, 1(1), pp. 81–106.

doi: 10.1007/BF00116251.

Quinlan, J. R. (1993) *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Raschka, S. (2014) *Predictive modeling, supervised machine learning, and pattern classification*. Disponível em: https://sebastianraschka.com/Articles/2014_intro_supervised_learning.html (Acedido: 29 de Setembro de 2019).

Rodriguez, J. D. *et al.* (2010) «Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation», *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3), pp. 569–575. doi: 10.1109/TPAMI.2009.187.

Rulequest (2017 a) *Is C5.0 Better Than C4.5?*, *Rulequest*. Disponível em: https://www.rulequest.com/see5-comparison.html (Acedido: 1 de Julho de 2019).

Rulequest (2019 b) *See5: An Informal Tutorial*, *Rulequest*. Disponível em: https://www.rulequest.com/see5-win.html (Acedido: 1 de Julho de 2019).

Saito, T. e Rehmsmeier, M. (2015) «The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets», *PLOS ONE*. Editado por G. Brock, 10(3), p. e0118432. doi: 10.1371/journal.pone.0118432.

Salehi, A. *et al.* (2017) «Data mining techniques for anti money laundering», *International Journal of Applied Engineering Research*, 12(20), pp. 10084–10094.

Schapire, R. E. (1990) «The strength of weak learnability», *Machine Learning*, 5(2), pp. 197–227. doi: 10.1007/BF00116037.

Senator, T. *et al.* (1995) «The FinCEN artificial intelligence system: identifying potential money laundering from reports of large cash transactions», *Proceedings of the 7th Conference on Innovative Applications of AI*, 16(4), pp. 21–23. Disponível em: http://www.aaai.org/Library/IAAI/1995/iaai95-015.php%5Cnpapers2://publication/uuid/3377C791-9E17-4375-99E3-450976A4CDCC.

Spackman, K. A. (1989) «SIGNAL DETECTION THEORY: VALUABLE TOOLS FOR EVALUATING INDUCTIVE LEARNING», em *Proceedings of the Sixth International Workshop on Machine Learning*. Elsevier, pp. 160–163. doi: 10.1016/B978-1-55860-036-2.50047-3.

Tan, P. N. *et al.* (2013) *Introduction to Data Mining: Pearson New International Edition*. Pearson Education Limited.

Thai-Nghe, N. *et al.* (2010) *Cost-Sensitive Learning Methods for Imbalanced Data*, *Proceedings of the International Joint Conference on Neural Networks,* pp. 1-8. doi: 10.1109/IJCNN.2010.5596486.

«Two Modifications of CNN» (1976) *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(11), pp. 769–772. doi: 10.1109/TSMC.1976.4309452.

Vorhies, W. (2016) *CRISP-DM – a Standard Methodology to Ensure a Good Outcome - Data Science Central*, *Data Science Central*. Disponível em: https://www.datasciencecentral.com/profiles/blogs/crisp-dm-a-standard-methodology-to-ensure-a-good-outcome (Acedido: 21 de Maio de 2019).

Witten, I. H. *et al.* (2011) «Chapter 5 - Credibility: Evaluating What's Been Learned», em Witten, I. H. et al. (eds.) *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. Third Edit. Boston: Morgan Kaufmann (The Morgan Kaufmann Series in Data Management Systems), pp. 147–187. doi: https://doi.org/10.1016/B978-0-12-374856-0.00005-5.

Witten, I. H. *et al.* (2017) «Chapter 4 - Algorithms: The basic methods», em Witten, I. H. et al. (eds.). Morgan Kaufmann, pp. 91–160. doi: https://doi.org/10.1016/B978-0-12-804291-5.00004-0.

Wu, X. *et al.* (2008) «Top 10 algorithms in data mining», *Knowledge and Information Systems*, 14(1), pp. 1–37. doi: 10.1007/s10115-007-0114-2.

Zareapoor, M. e Shamsolmoali, P. (2015) «Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier», *Procedia Computer Science*, 48, pp. 679–685.

doi: 10.1016/j.procs.2015.04.201.