UNIVERSIDADE D
COIMBRA

Mariana da Conceição Ferreira Abreu

MODELOS DE AVALIAÇÃO DE RISCO DE CRÉDITO

APLICAÇÃO DE *MACHINE LEARINING*

Trabalho de Projeto no âmbito do Mestrado em Economia, na especialização de Economia Financeira, orientada pelo Professor Doutor Pedro Manuel Cortesão Godinho e apresentado à Faculdade de Economia da Universidade de Coimbra.

Fevereiro de 2020

FACULDADE DE ECONOMIA
UNIVERSIDADE D
COIMBRA

Mariana da Conceição Ferreira Abreu

Modelos de Avaliação de Risco de Crédito:
Aplicação de *Machine Learning*

Trabalho de Projeto de Mestrado em Economia, na especialização de Economia Financeira, apresentado à Faculdade de Economia da Universidade de Coimbra para a obtenção do grau de Mestre.

Orientador: Professor Doutor Pedro Manuel Cortesão Godinho

Fevereiro de 2020

Agradecimentos

Com elevada estima e admiração, agradeço ao meu orientador, Professor Doutor Pedro Manuel Cortesão Godinho, pela inigualável dedicação e disponibilidade, pela sua compreensão e encorajamento e por todos os conselhos e sugestões sábias da sua orientação.

À minha Mãe e ao meu Pai, um agradecimento especial, pelo apoio incondicional, pelo incentivo e pela força enorme que me dedicaram desde sempre. São os meus maiores mestres.

Aos meus Amigos, que desde o meu primeiro ano da faculdade foram incrivelmente leais, disponíveis e sinceros comigo. Vocês são como uma segunda família para mim.

À Margarida, à Marisa e a Sara, pelos conselhos. Às minhas colegas de casa, pelo ombro amigo e pela companhia. E, ao João, pela enorme paciência, pelo carinho e pelo tempo que me dedicou. Tenho-vos muita estima.

A todos os meus professores da Faculdade de Economia da Universidade de Coimbra, por todos os conhecimentos, orientações e valores académicos que me transmitiram durante todo o meu percurso académico.

À Faculdade de Economia da Universidade de Coimbra, por me ter proporcionado a oportunidade de me formar na área de economia.

O que prevemos raramente ocorre; o que menos esperamos geralmente acontece.

Benjamin Disraeli

Resumo

Existem vários métodos que ao longo dos anos tem sido empregues na avaliação de risco de crédito, sobretudo, metodologias tradicionais como o Modelo de Análise Discriminante (ADi), Modelo Logit e Modelo Probit, e metodologias mais sofisticadas de *Machine Learning*, como Árvores de Classificação (AC), *Random Forests* (RF), Redes Neuronais (RN) e *Support Vector Machines* (SVM). Na revisão de literatura são apresentados alguns estudos que recorrem a metodologias tradicionais e a metodologias de *Machine Learning*. Estas últimas não só se apresentam teoricamente como são estudadas na prática para avaliar diferentes aplicações de risco de crédito, sendo aplicados a duas bases reais, disponíveis publicamente, uma referente ao cumprimento de pagamento de cartões de crédito em Taiwan e outra referente ao risco de crédito na Alemanha. Ambas as bases de dados incluem uma variável de resposta binária relativa ao risco de crédito.

Em cada modelo experimentaram-se alguns meta-parâmetros, tendo a devida precaução na sua seleção, de forma a não repeti-los nas diferentes combinações do mesmo modelo e, consequentemente, de forma a evitar o *overfitting*.

Este estudo efetua uma análise do desempenho dos modelos de *Machine Learning* individuais e também do desempenho de uma técnica de *Ensemble* baseada nos resultados obtidos pelos diferentes modelos, com intuito de determinar qual destes revela um melhor desempenho na avaliação de risco de crédito. A maioria dos resultados deste estudo empírico permitem concluir que os desempenhos da técnica de Ensemble são superiores aos dos modelos individuais. Também o modelo *Random Forest* realçou os melhores desempenhos de entre todos os modelos individuais.

Palavras-Chave: Machine Learning; Credit Scoring; Ensemble.

Classificação JEL: G21; C44; C45.

Abstract

There are several methods that over the years have been used in credit risk assessment, especially traditional methodologies such as the Discriminant Analysis Model (ADi), Logit Model and Probit Model, and more sophisticated Machine Learning methodologies, such as Classification Trees (AC), Random Forests (RF), Neural Networks (RN) and Support Vector Machines (SVM). In the literature review presents some studies that use traditional methodologies and Machine Learning methodologies. This last not only present themselves theoretically, but are studied in practice to evaluate different applications of credit risk, being applied to two real bases, publicly available, one referring to the fulfillment of credit card payments in Taiwan and the other referring to credit risk. in Germany. Both databases include a binary response variable for credit risk.

In each model, some meta-parameters were experimented, taking due care in their selection, so as not to repeat them in the different combinations of the same model and, consequently, in order to avoid overfitting.

This study performs an analysis of the performance of the individual Machine Learning models and also of the performance of an Ensemble technique based on the results obtained by the different models, in order to determine which one shows a better performance in the credit risk assessment. Most of the results of this empirical study allow us to conclude that the performances of the Ensemble technique are superior to those of the individual models. Also the Random Forest model highlighted the best performances among all individual models.

Key-Word: Machine Learning; Credit Scoring; Ensemble.
JEL Codes: G21; C44; C45.

Lista de Siglas e Acrónimos

A

AC – Árvores de Classificação

AD – Árvores de Decisão

ADi – Modelo de Análise Discriminante

ANS – Aprendizagem Não Supervisionada

AS – Aprendizagem Supervisionada

AUC – *Area Under the Curve*

C

CV – *Cross-Validation*

K

k-NN – *K-Nearest Neighbors*

L

Logit – *Logistic Regression*

LS-SVM – *Least Square-Support Vecto Machine*

P

PMC – *Perceptron* Multicamadas

R

RC – Risco de Crédito

RF – *Random Forest*

RN – Redes Neuronais

S

SVM – *Support Vector Machine*

W

Weka – *Waikato Environment for Knowledge Analysis*

Índice

Índice de Figuras

Índice de Tabelas

1. Introdução

A tomada de decisão sobre empréstimos por parte das instituições financeiras e outros credores é muito importante, uma vez que más decisões podem levar a severas consequências (Tsai & Wu, 2008). Por isso é que, ao longo das últimas décadas, os modelos de *Credit Scoring* têm sido tão amplamente analisados em vários estudos para a avaliação de pedidos de crédito. Além disso, há outros motivos como a possibilidade de tomadas de decisão mais precisas e mais rápidas, a tentativa de diminuição de custos de análise de crédito, prevenção do risco de incumprimento, que também contribuem para a crescente atenção sobre o tema de *Credit Scoring*.

Mas afinal o que é o *Credit Scoring*? Mester (1997) e Chye (2004) definem formalmente *Credit Scoring* como "um método estatístico (ou quantitativo) que é usado para prever a probabilidade de um requerente de empréstimo ou de um mutuário entrar em incumprimento"[1]. Os modelos de *Credit Scoring* produzem uma "pontuação" que ajuda as instituições financeiras e outros credores a avaliar e classificar o risco de crédito para diversas aplicações, seja em pedidos de empréstimo por particulares (crédito à habitação, crédito pessoal, crédito automóvel, entre outros), seja em crédito concedido através de cartões ou crédito a empresas. Basicamente, esses modelos classificam o mutuário em categorias, de acordo com a probabilidade de cumprir os pagamentos futuros relativos ao empréstimo e, assim, as entidades credoras podem avaliar o risco associado à concessão de crédito ao cliente. Para produzir um modelo de *Credit Scoring* há variadas características que devem ser tidas em consideração, como o estado civil, a idade, emprego e a duração dele, a condição económica, o motivo do pedido de empréstimo, o histórico de empréstimos anteriores, os ativos e os passivos na posse do cliente, entre muitos outros fatores. Mester (1997) menciona que todos os fatores potenciais que podem estar relacionados com o desempenho do empréstimo devem ser considerados e podem ser utilizados nos modelos de *Credit Scoring*.

Os modelos utilizados pelas instituições financeiras para avaliar o risco de crédito em empréstimos sofreram uma grande evolução. Tradicionalmente, utilizavam-se análises

[1] Tradução livre da autora. No original "Credit scoring is a statistical method used to predict the probability that a loan applicant or existing borrower will default." (Mester, 1997)

essencialmente subjetivas para avaliar o risco de crédito. Posteriormente, em resposta às alterações sofridas no mercado financeiro, desenvolveram-se métodos estatísticos mais objetivos de *Credit Scoring*, como o modelo *Logit*, o modelo *Probit*, e o modelo de Análise Discriminante Linear. Atualmente, empregam-se novas metodologias, nomeadamente modelos mais sofisticados de *Machine Learning*.

A finalidade deste trabalho de projeto é determinar qual dos diferentes modelos de *Machine Learning* evidencia um melhor desempenho na avaliação de risco de crédito. No entanto, o resultado a prever não é certo. Tal como refere o autor Mester (1997), mesmo os modelos de *Credit Scoring* que apresentem o melhor desempenho "não preveem com certeza absoluta o desempenho de um empréstimo individual, somente fornecem uma previsão bastante precisa da probabilidade de um requerente do empréstimo, com certas caraterísticas, entrar em incumprimento."[2] Portanto, com base na literatura existente inferiu-se quais os modelos de *Machine Learning* mais utilizados e pretende-se de certa forma complementar os estudos existentes reforçando a ideia que estes modelos evidenciam bons resultados na avaliação do risco de crédito e, adicionalmente, determinando qual o melhor modelo.

O trabalho de projeto está estruturado da seguinte forma. Primeiro apresenta-se a revisão de literatura, na qual se aborda a evolução dos modelos de *Credit Scoring*, desde dos modelos mais primordiais até aos modelos de *Machine Learning*. De seguida, descreve-se a metodologia utilizada, apresentando os conceitos básicos dos principais modelos de *Machine Learning* a serem considerados no trabalho. Posteriormente, descrevem-se os dados utilizados e o programa aplicado, bem como alguns aspetos da abordagem seguida. Segue-se uma secção onde se apresentam os principais resultados obtidos com a análise empírica. Por fim, apresenta-se a análise dos resultados seguida da conclusão.

[2] Tradução livre da autora. No original "Even a good scoring system won't predict with certainty any individual loan's performance, but it should give a fairly accurate prediction of the likelihood that a loan applicant with certain characteristics will default." Mester (1997)

2. Revisão da Literatura

Por muitos anos, as instituições de crédito dependiam quase exclusivamente de uma abordagem subjetiva, realizada por um especialista, a diversas informações sobre as características do mutuário. Segundo Altman e Saunders (1997), características, como o caráter do mutuário (reputação), o capital (alavancagem), a capacidade de "reembolsar" (volatilidade dos ganhos) e das garantias eram denominados de 4 "C's". Este especialista decidia com base nestas caraterísticas se o banco devia ou não conceder crédito ao cliente, sendo este, geralmente, um processo demorado. Esta abordagem subjetiva, por si só, não garantia uma boa avaliação do risco de crédito de um cliente, tendo por isso sido desenvolvidos modelos mais objetivos e formais.

Inicialmente, começou-se por utilizar modelos econométricos de escolha binária, como os modelos *Logit* (*Logistic Regression*), *Probit* e Análise Discriminante Linear (ADiL). Posteriormente, os avanços da tecnologia de computação e o crescimento exponencial de grandes bancos de bases de dados impulsionaram o desenvolvimento de modelos de *Credit Scoring* mais sofisticados. Surgem então várias técnicas de *Machine Learning* que começam a ser utilizadas em diversas áreas como, por exemplo, Árvores de Classificação (AC), *Random Forests* (RF), Rede Neuronais (RN) e *Support Vector Machines* (SVM).

Na literatura mais recente sobre modelos *Machine Learning*, todas estas técnicas mencionadas anteriormente têm, de facto, vindo a demonstrar bons resultados em aplicações de *Credit Scoring*, e também se pode constatar que há diversos autores que tem vindo a comparar os seus resultados com os resultados de modelos tradicionais.

West (2000) realizou uma análise comparativa entre cinco métodos de RN e os seguintes métodos tradicionais: ADiL, *Logit*, *K-Nearest Neighbors* (k-NN) e AD. Os resultados obtidos demonstram que os modelos de RN podem obter boas previsões em aplicações de *Credit Scoring*. O modelo *Logit* é considerado o mais preciso dos métodos tradicionais e é uma boa alternativa às RN.

Baesens *et al*., (2003) estudaram o desempenho de vários modelos com base em dois critérios: percentagem dos casos classificados corretamente e a curva AUC (*Area Under the Curve*)[3]. O método RN e LS-SVM (*Least Square*-SVM) demonstraram um bom desempenho nos dois critérios, tal como o método ADiL e o método *Logit*. Os autores sugerem o estudo futuro da combinação de métodos.

No estudo de Galindo e Tamayo (2000) realizou-se uma análise comparativa entre um modelo estatístico de Análise de Regressão (*Probit*) e modelos de *Machine Learning* atuais, como AD, RN e k-NN, para classificar empréstimos hipotecários. Os autores concluíram que o método AD fornece os melhores resultados de entre todos os métodos, com a taxa de erro mais baixa, seguida do modelo RN, do k-NN e, finalmente, do modelo *Probit*.

Wang *et al*. (2011) evidenciam que as AD apresentam resultados fracos comparativamente a outros métodos individuais, como consequência de estas serem afetadas por dados com ruído e por atributos redundantes. Nesta pesquisa utilizaram-se então duas estratégias de *Ensemble* de AD, o *Bagging* e *Random subspace*, que permitiram corrigir em parte as duas causas do fraco desempenho das árvores e possibilitaram que estas apresentassem melhores resultados que os outros métodos individuais, nomeadamente, *Probit*, ADiL, e Redes Neuronais.

Também Wang *et al*. (2012) exploraram o desempenho de métodos de *Ensemble*, ou seja, o desempenho da combinação de múltiplos modelos, neste caso do conjunto dos quatro modelos seguintes: *Logit*, AC, RN, SVM. O estudo a duas aplicações reais de *Credit Scoring* revelou que os três métodos de *Ensemble Bagging*, *Boosting* e *Stacking* analisados produzem melhores resultados do que os métodos individuais, especialmente, o método *Bagging*, que obteve melhor desempenho do que o método *Boosting*. Para além disso, método *Bagging* AC e *Stacking* apresentaram os melhores desempenhos nos três critérios analisados: precisão média, erro do tipo I e erro do tipo II.

O estudo realizado por Brown e Mues (2012) para prever o incumprimento em empréstimos através da análise ao desempenho de várias técnicas num conjunto de dados não balanceados,

[3] Curva AUC é uma medida que indica o poder discriminatório de um modelo de classificação binário. Mede a área abaixo da curva (ROC) *Receiver Operating Characteristic* (Brown e Mues, 2012).

recorreu ao critério da curva AUC. Os resultados obtidos indicam que o método RF e *Gradient Boosting* lidam muito bem com conjuntos de dados desequilibrados, enquanto, o algoritmo de árvore de decisão (C4.5), Análise Discriminante Quadrática (ADiQ) e k-NN obtiveram os piores desempenhos. ADiL e *Logit* apresentaram resultados razoáveis. A experiência também demonstra que o método *LS*-SVM não obtém bons resultados em bases de dados com um desequilíbrio muito grande entre as classes de dados.

Outros modelos foram implementados no estudo do risco de crédito, como *Random Forests* (RF). Por exemplo, os autores Malekipirbazari e Aksakalli (2015) como forma de identificar se os clientes constituem um bom ou mau risco de crédito na plataforma *social lending*[4], compararam os seguintes modelos de *Machine Learning*: RF, SVM, *Logit* e k-NN. O método RF obteve um excelente desempenho a identificar os clientes que constituem um bom risco de crédito e o método k-NN não ficou atrás das RF. Quanto ao SVM, os autores recomendam o teste de outros meta-parâmetros, como outro tipo de funções *kernel,* de forma a verificar se esse ajuste permite a melhoria do seu desempenho.

A pesquisa realizada por Louzada *et al*. (2016) centra-se na revisão da literatura de 187 artigos científicos correspondentes a um período de 1992 a 2015. A literatura analisada baseia-se nas técnicas de classificação binária utilizadas para análises financeiras de *Credit Scoring.* Nesta investigação ainda se realiza uma análise comparativa dos métodos que popularmente tem sido investigados, com auxílio de três conjuntos de dados de *Credit Scoring*. O estudo destaca o método SVM e os métodos baseados em lógica Fuzzy[5] como os dois melhores métodos deste estudo, visto que apresentam o maior desempenho preditivo, com base nos resultados de duas medidas: *Approximate Correlation* e *F 1-Score Measure* sobre três conjuntos de dados de *Credit Scoring*. De seguida, em terceiro lugar apresentam-se as Árvores, independentemente de os

[4] Social Lending – "…also known as peer-to-peer (P2P) lending, is emerging as an alternative to banks where individual members lend and borrow money using an online trading platform without the help of official financial intermediaries such as banques." (Malekipirbazari & Aksakalli, 2015)

[5] Fuzzy logic foi introduzido por Zadeh, "…é um sistema matemático que lida com informações imprecisas na forma de termos linguísticos, fornecendo uma reposta aproximada a uma questão baseada em conhecimentos imprecisos, incompletos e não totalmente confiáveis." (Louzada *et al*., 2016)

dados serem ou não balanceados. Contrariamente, nos dados não balanceados, às RN apresentam um fraco desempenho. A Regressão Linear, *k*-NB e ADiL revelam um comportamento razoável. Por fim, quando há dados não balanceados os modelos com piores desempenhos são métodos baseados em Programação Genética[6] e o método *Logit*.

Note-se que não há convergência de entre os resultados dos estudos realizados sobre o tema de avaliação de risco de crédito, relativamente ao método que permite obter os melhores resultados.

[6] Programação Genética tem objetivo de criar uma população de respostas possíveis para um problema, que passa por um processo de evolução através da aplicação de operações como: mutação, reprodução e cruzamento (Louzada *et al*.,2016).

3. Metodologia

Primeiramente, é importante definir o conceito de *Machine learning*. Os autores Han *et al.* (2011) definem este conceito como sendo a área de pesquisa que utiliza "programas de computador que automaticamente aprendem a reconhecer padrões complexos e a tomar decisões inteligentes com base em dados" [7].

Neste capítulo, pretende-se analisar os seguintes modelos de *Machine Learning*: Árvores de Decisão (AD), *Random Forests* (RF), Redes Neuronais (RN) e os *Support Vector Machine* (SVM). Todos os modelos descritos anteriormente são classificados como modelos de aprendizagem supervisionada. A Aprendizagem Supervisionada (AS) é uma abordagem poderosa do *Machine Learning* que permite partir de exemplos em que o valor do *output*[8] é conhecido, e treinar estes dados através do uso de um modelo que aprende a relacionar e a mapear um conjunto de variáveis *input*[9] (também denominadas de atributos[10]) até à sua respetiva variável *output*. De acordo com os valores output obtidos, a AS pode ser dividida em duas principais subcategorias: problemas de classificação e problemas de regressão. Os problemas de classificação ocorrem quando o *output* que se pretende prever é categórico ou qualitativo, enquanto que os problemas de regressão ocorrem quando o valor do output previsto é um valor real ou contínuo. O objetivo desta abordagem é conseguir classificar/prever corretamente o valor das variáveis *output* de um novo conjunto de dados.

Para além da abordagem de AS, existe a abordagem Aprendizagem Não Supervisionada (ANS). Contrariamente a AS, ANS não parte de exemplos em que o valor do *output* é conhecido, tem a finalidade de encontrar associações e padrões dentro o conjunto de *inputs*. Segundo Rokach e

---

[7] Tradução livre da autora. No original "…computer programs to automatically learn to recognize complex patterns and make intelligent decisions based on data." (Han *et al*., 2011)

[8] *Output* é a variável (resultado) que se pretende estudar - ou a classificação, no caso de métodos de regressão ou o valor a prever, no caso de métodos de regressão.

[9] *Input* é a variável de entrada, que representa uma observação do conjunto de dados.

[10] Atributo (também conhecido como variável) é uma observação ou uma característica de um determinado objeto dos dados (como por exemplo, de um cliente). Podem ser numéricos (ou seja, apresentam um valor), nominais (isto é, representam uma categoria) ou binários (ou seja, possuem dois valores possíveis).

Maimon (2008), esta abordagem "refere-se técnicas de aprendizagem que agrupam instâncias sem um atributo dependente pré-especificado"[11]. Brownlee (2016) menciona que os problemas de ANS podem ser agrupados em dois tipos: *Clustering* e a Associação. *Clustering* é uma técnica que tenta encontrar subgrupos ou *clusters* de conjuntos de observações que são semelhantes entre si. A técnica de Associação tem objetivo de encontrar regras de associação que descrevam a maioria do conjunto de dados.

Este trabalho apenas aborda problemas de classificação.

3.1. Árvores de decisão

Uma árvore de decisão é um modelo hierárquico de decisões e consequências que pode ser usado tanto para representar modelos de regressão como modelos de classificação.

Tal como a própria denominação indica, os modelos de árvores de decisão apresentam uma estrutura em forma de árvore (Figura 1). Essa estrutura é composta por uma raiz, por ramos, por nós internos (também designados por nós não terminais/teste) e por folhas (também conhecidos por nós terminais).

As folhas ou nós terminais representam o final da cadeia, ou seja, o último nó antecedido por um último ramo. Cada nó não terminal representa testes em um ou mais atributos, os ramos representam um intervalo de valores dos atributos e as folhas representam os resultados do processo de decisão (Zhao e Zhang, 2008).

A construção das árvores de decisão baseia-se num processo que é realizado recursivamente, de forma a particionar os subconjuntos (nós) em novos nós até encontrar a melhor divisão destes e, assim, sucessivamente, até se chegar aos nós terminais, as folhas, isto é, até não se encontrar divisões mais significativas (Biggs *et al*., 1991). Daí advém o nome de Particionamento Recursivo para este método que permite a criação de árvores de decisão.

[11] Tradução livre da autora. No original "… learning techniques that group instances without a prespecified dependent attribute". (Rokach e Maimon, 2008)

Existem ainda vários benefícios em usar modelos de árvores de decisão, tal como: são de fácil interpretação e implementação; possibilitam uma análise eficiente e objetiva, conseguem lidar com uma variedade de tipos de dados (sejam nominais, numéricos ou textuais); permitem um elevado desempenho e são capazes de processar dados que possuem *outliers* e ruídos, com auxílio da técnica de poda.

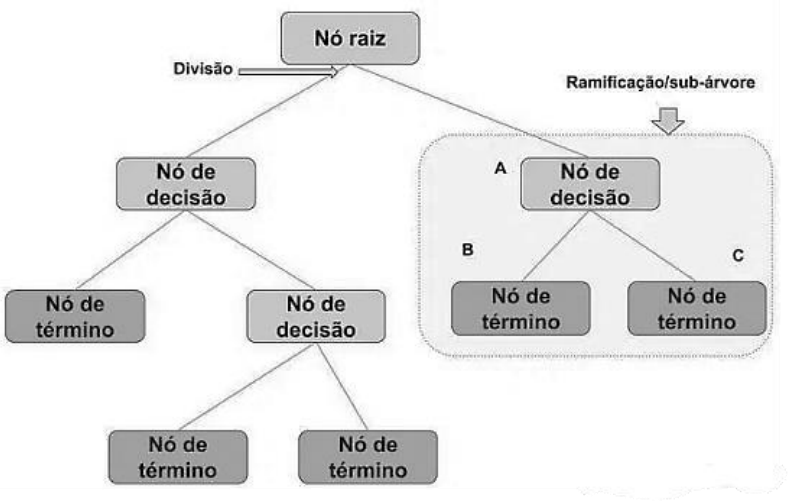
A técnica de poda (também conhecida por "*pruning*") é muito importante na criação de árvores, visto que permite detetar e excluir as "impurezas" das árvores de decisão, tais como *outliers* e ruídos[12] (Han *et al*., 2011). Este processo geralmente usa medidas estatísticas para remover os ramos menos confiáveis e produzir árvores mais pequenas e menos complexas. Também resolvem o problema de *overfitting*[13] dos dados.

Há duas abordagens comuns para podar árvores: *prepruning* e *postpruning.* Na abordagem *prepruning,* as árvores são "podadas" durante a sua construção, ou seja, decidindo não dividir (nó folha) ou dividir mais o subconjunto. A segunda abordagem é mais comum e permite remover as subárvores de uma árvore completa, isto é, após a criação da árvore (Han *et al*., 2011).

[12] *Outliers*- são objetos que não estão em conformidade com o comportamento ou modelo geral dos dados. Enquanto, um Ruído "é um erro aleátorio ou variação em uma variável medida" (Han et la., 2011).

[13] *Overfitting*- é um problema do *Machine Learning*, em que um determinado modelo se ajusta de forma excessiva a todas as observações de uma amostra (até mesmo *outliers* e ruídos), e, consequentemente, mostra-se ineficaz por não refletir a verdadeira tendência geral dos dados.

Figura 1. Exemplo da estrutura de uma Árvore de decisão.



Fonte: Baseada numa imagem retirada de https://www.vooo.pro/insights/um-tutorial-completo-sobre-a-modelagem-baseada-em-tree-arvore-do-zero-em-r-python/.

3.2. *Random Forest*

Breiman (2001, *apud* Rokach e Maimon, 2008: 108) propôs o popular modelo *Random Forest* (RF), que é um modelo composto por um conjunto de árvores que podem ser de classificação ou de regressão. O número de árvores que compõe o conjunto é definido à priori.

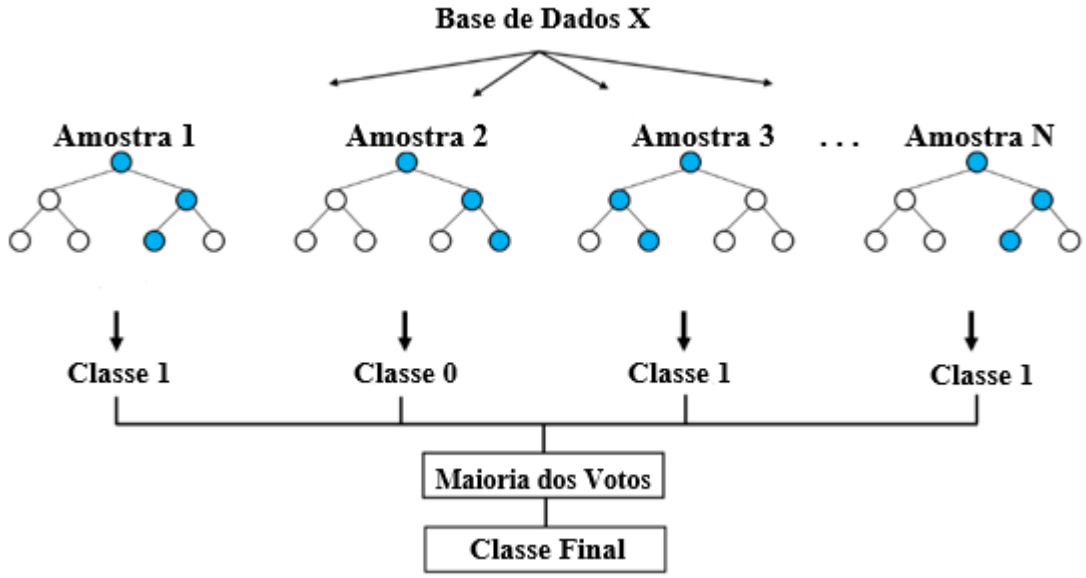
Random Forest recorre à técnica de *bootstrap aggregation*, também conhecida como *bagging*. A aplicação desta técnica nas *random forests* consiste na criação de árvores de decisão a partir da seleção de subamostras aleatórias da amostra de dados originais. Em cada nó das árvores de decisão criadas é selecionado aleatoriamente um subconjunto de atributos. Cada subamostra é criada com reposição, isto significa, que se pode repetir a escolha da mesma observação numa subamostra. Este processo é repetido até criar o número de árvores definido *a priori*, formando uma floresta (Figura 2). É importante referir que a escolha aleatória de amostras do conjunto de treino e dos atributos usados em cada nó de cada árvore de decisão permitem reduzir o problema de *overfitting* (conceito abordado com mais detalhe na seção 4.2.).

Quanto à previsão do modelo RF, esta é, geralmente, obtida através do voto da maioria das árvores de decisão para a classificação, ou calculada através da média das previsões individuais

das árvores, para a regressão. Tipicamente não se utiliza o método de *prunning* neste modelo, ou seja, as árvores criadas não são podadas.

Em suma, as combinações das previsões das árvores de decisão geralmente produzem um desempenho substancialmente melhor ou uma melhor precisão do que às árvores de decisão individuais (Zhao e Zhang, 2008).

Figura 2. Exemplo do processo de *Bootstrap Aggregation* num problema de classificação.



Fonte: Baseada numa imagem retirada de https://medium.com/@ar.ingenious/applying-random-forest-classification-machine-learning-algorithm-from-scratch-with-real-24ff198a1c57, com adaptações feitas pela autora.

3.3. *Support Vector Machine*

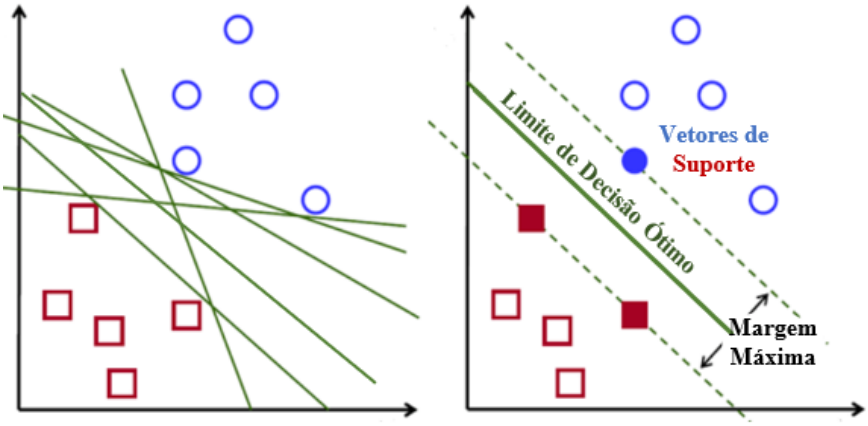
Support Vector Machine (SVM) foi proposto por Vapnik *et al*. (1992, *apud* Han *et al*., 2011: 408). É um método supervisionado utilizado para problemas de classificação e regressão.

SVM tem finalidade de produzir uma margem máxima (isto é, a maior margem) que permita separar linearmente um conjunto de dados do seu limite de decisão. Basicamente, de todos os limites de decisão possíveis, escolhe-se o que é mais distante dos pontos mais próximos do

limite de decisão (Figura 3). Daí advém o nome de margem máxima. Esses pontos mais próximos são denominados de vetores de suporte e são estes que definem a margem e o limite de decisão. O limite de decisão já referido pode ser um ponto, uma linha, um plano ou um hiperplano consoante a dimensão do espaço. Por exemplo, num espaço bidimensional, o limite de decisão é uma linha, enquanto, num espaço tridimensional é um plano. Com o propósito de se compreender melhor o conceito de SVM, apresenta-se na Figura 3, um espaço linear de duas dimensões, no qual as duas classes são apresentadas por quadrados e círculos. Existem diversos limites de decisão (linhas) traçados na figura da esquerda.

No entanto, apenas existe um limite de decisão ótimo, representado por uma linha, que é a bissetriz do conjunto de vetores suporte mais próximos. Os vetores de suporte são representados por quadrados e um círculo preenchidos e a margem ótima garante a máxima separação ou separação ótima e generalizada das classes.

Figura 3. Exemplo da aplicação do modelo SVM num espaço linear de duas dimensões, com diferentes limites de decisão e com o limite de decisão ótimo.

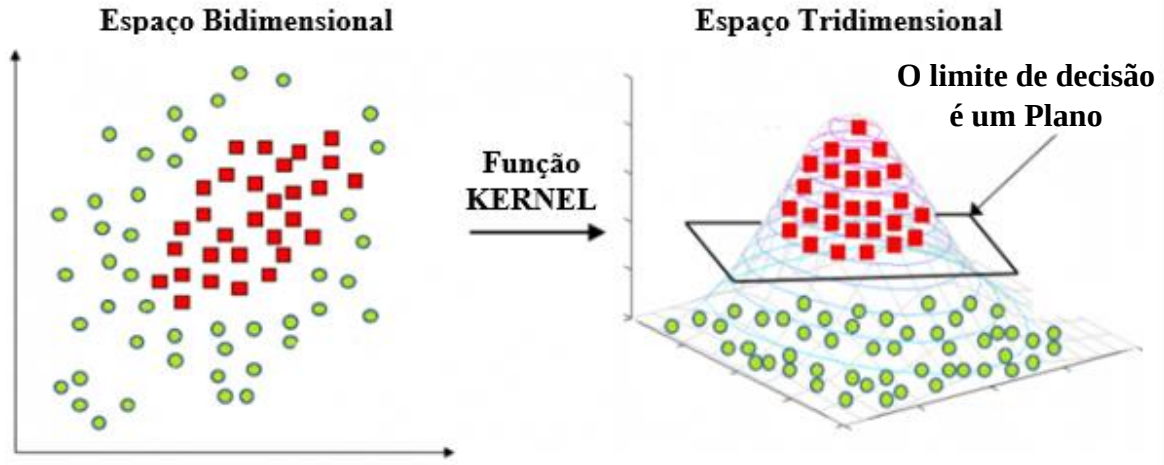


Fonte: Baseada numa imagem retirada de https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c, com adaptações feitas pela autora.

Evidentemente, que quanto mais longe estão os vetores suporte do limite de decisão, provavelmente mais dados se encontrarão corretamente classificados ou, como também se pode dizer, os erros de generalização serão minimizados.

Na realidade, há conjuntos de dados que não podem ser linearmente separáveis no seu espaço de dimensão original. Neste caso, a ideia subjacente ao SVM é mapear (transformar) o conjunto de dados para um espaço de dimensão superior à dimensão original para que se possa separar linearmente os dados (tal como apresenta a Figura 4). Para tal, utilizam-se as funções *kernel* para calcular as coordenadas dos pontos/observações num espaço de dimensão superior.

Figura 4. Exemplo de um caso não linear mapeado para uma dimensão tridimensional.



Fonte: Baseada numa imagem retirada de https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d , com adaptações feitas pela autora.

Devido à existência de *outliers*, erros de classificação e/ou outras caraterísticas dos dados, há conjuntos de dados que não são linearmente separáveis, mesmo depois de serem mapeados para um espaço de dimensão superior. Por isso, os vetores suporte permitem a utilização das margens suaves, estas correspondem a situações em que se tenta encontrar um limite de decisão que faça separação dos dados, mas que permite que alguns pontos fiquem do lado errado desse limite.

3.4. Rede Neuronal

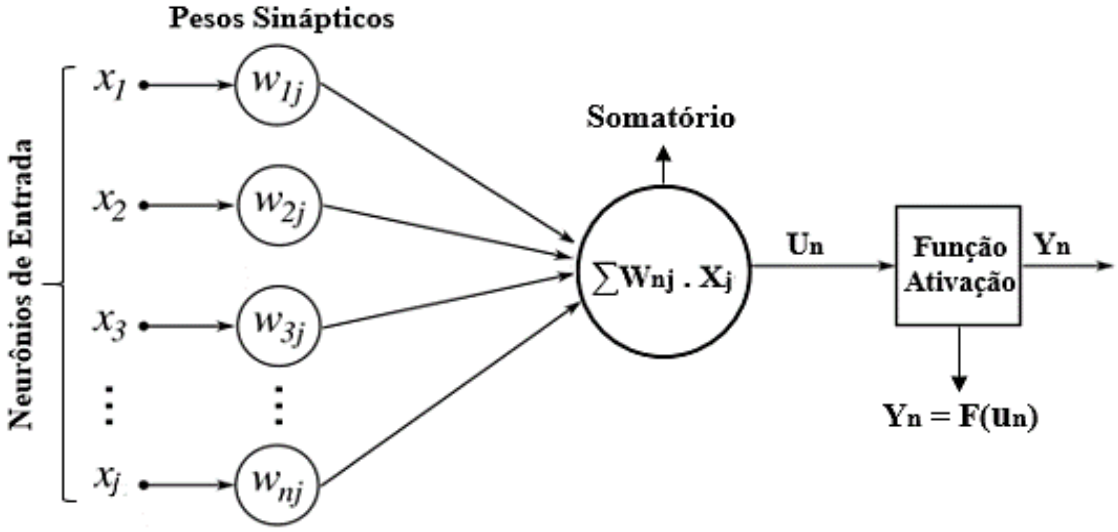
McCulloch e Pitts (1943, *apud* Torgo, 2011: 126) propuseram o primeiro modelo de Redes Neuronais ou *Artificial Neural Networks*. O modelo foi inspirado no processamento de

informação no cérebro humano, por isso, é um modelo análogo ao processo de transmissão de um impulso nervoso. A estrutura de uma rede neuronal é formada por um conjunto de unidades de computação denominados de neurónios artificiais ou nós conectados entre si.

De forma a se compreender melhor este conceito Rede Neuronal, primeiramente, aborda-se a Rede Neuronal mais simples de uma só camada, designada por *perceptron*. A primeira camada e única é composta por vários neurónios de entrada, as variáveis, em que cada uma apresenta um valor, também conhecido por um sinal. Cada uma destas variáveis é multiplicada por um peso ou peso sináptico. Por fim, a soma do produto de cada um desses valores ($\mathbf{x_j}$) multiplicado pelo seu peso respetivo ($\mathbf{w_{nj}}$), origina um único neurónio, ou seja, resulta num único valor ($\mathbf{u_n}$). Este valor ($\mathbf{u_n}$) é transformado a partir de uma função de ativação que dá origem ao valor $\mathbf{y_n}$, que indica qual a classificação a atribuir.

A seguir considera-se que a Figura 5 pode representar uma das aplicações estudadas neste trabalho, em que se pretende prever o cumprimento ou não dos pagamentos por parte de um cliente. Isto é efetuado, substituindo as características do candidato nas variáveis ($\mathbf{x_j}$) e multiplicando pelo respetivo peso ($\mathbf{w_{nj}}$) do neurónio correspondente, originando-se assim o valor $\mathbf{u_n}$. A partir deste, a função de ativação permite calcular o valor final do output, **y**.

Figura 5. Exemplo de um *Perceptron*.



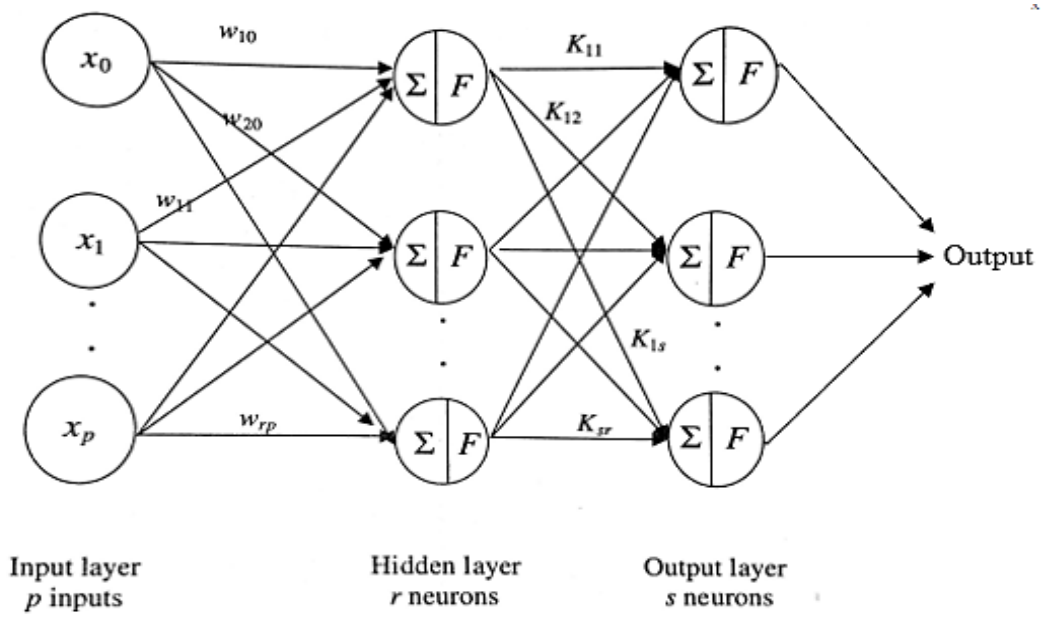
Fonte: Baseada numa imagem retirada de https://nl.wikipedia.org/wiki/Perceptron, com adaptações feitas pela autora.

O algoritmo *perceptron* tem uma limitação de não conseguir classificar todo o tipo de casos, particularmente nos casos em que estes não são linearmente separáveis. Para estes casos, surgiu o algoritmo *Perceptron* Multicamadas (PMC), que apresenta a vantagem de conseguir lidar com problemas altamente não lineares. Este algoritmo apresenta uma estrutura típica de uma Rede Neuronal só que com mais camadas. PMC é constituído por uma primeira camada de variáveis, em que cada uma possui um peso e está conectada a uma ou mais camadas intermédias ocultas e estes, entre si, estão conectadas às últimas camadas dos neurónios *output*.

Tal como o algoritmo *perceptron*, cada um dos neurónios da rede constitui um valor que multiplicado pelo respetivo peso e somado aos produtos dos outros neurónios da mesma camada que estejam conectados ao mesmo neurónio da camada a seguir, produzem o respetivo valor desse neurónio. O processo repete-se até se chegar a classificação do output final, que fornece uma previsão.

Na seguinte Figura 6 representa-se o PMC com uma camada intermédia (*hidden layer*) e os respetivos pesos (designados por "$\mathbf{w_{rp}}$" e "$\mathbf{k_{sr}}$") associados as conexões entre camadas.

Figura 6. Exemplo de um *Perceptron* de Multicamadas, com apenas uma camada intermédia (*hidden layer*).



Fonte: Baseada numa imagem retirada do livro "*Credit Scoring and Its Applications*" do autor Thomas (2002), com adaptações feitas pela autora.

4. Dados e Abordagem de Análise

Neste trabalho, utilizou-se duas bases de dados reais retiradas do conjunto de dados do repositório do *UCI Machine learning.* A primeira base de dados refere-se ao risco de incumprimento em pagamentos de cartões de crédito por parte de consumidores em Taiwan[14]. Esta base de dados é constituída por um total de 30 000 instâncias, possui uma variável resposta binária, que corresponde ao (in)cumprimento em pagamentos e um conjunto de 23 variáveis explicativas, sendo estas variáveis socioeconómicas, demográficas e relativas a empréstimos pendentes (ver Apêndice A). Os dados dizem respeito a um período de abril a setembro de 2005. A segunda base de dados diz respeito ao Risco de Crédito (RC) na Alemanha[15]. Esta contém informação sobre 1000 pedidos de empréstimos (instâncias) classificados em bom ou mau risco de crédito. Cada pedido é descrito por 20 atributos diferentes, tais como atributos demográficos, sociais, económicos, relativas ao plano de prestações e ao histórico de crédito (ver Apêndice B). Cada observação é confidencial e por isso, as instituições de crédito são anónimas.

4.1. Software *WEKA*

Neste trabalho utilizou-se o programa *Weka* (*Waikato Environment for Knowledge Analysis*), versão 3.6.0, que foi desenvolvido na Universidade de Waikato, na Nova Zelândia. É um *software* escrito na linguagem *Java*. Para além de fornecer uma variedade de algoritmos de *Machine Learning*, também inclui vários métodos e ferramentas de pré-processamento de dados, que permitem resolver problemas de *Machine Learning* (Witten e Frank, 2005).

A Tabela 1 apresenta as implementações do programa *Weka* utilizadas neste trabalho.

Tabela 1. Designação dos Algoritmos utilizados neste trabalho.

Modelos de Machine Learning	**Designação dos Algoritmos no Programa Weka**
Árvores de Classificação	Algoritmo J4.8
Random Forest	Algoritmo Random Forest

[14] Ver https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients retirado em (18/09/2019).

[15] Ver https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data) retirado em (18/09/2019).

Support Vector Machine	Algoritmo Libsvm
Rede Neural	Algoritmo Multilayer Perceptron

Fonte: Elaboração Própria Autora

4.2. Seleção de Meta-parâmetros

Para cada um dos modelos, escolheram-se alguns meta-parâmetros potencialmente mais relevantes, configuraram-se diferentes valores destes meta-parâmetros e experimentaram-se diversas combinações desses valores, de forma a obter as melhores configurações e a otimizar o desempenho de cada modelo. Passa a descrever-se os meta-parâmetros escolhidos para cada modelo.

Nas árvores de classificação, considerou-se o meta-parâmetro *Minimum number of Instances*, que representa o número mínimo de instâncias por folha e o meta-parâmetro *Confidence Factor*, que define "o limiar do erro permitido no conjunto de treino durante o processo de poda de árvores de decisão" [16]. Ao reduzir este limiar, a poda é efetuada com mais frequência gerando modelos mais gerais (Stiglic *et al*.,2012).

Para o método *Random Forest*, foram escolhidos os seguintes meta-parâmetros: *Max Depth*, que representa a profundidade/tamanho máximo de cada árvore; *Nº Features*, que é o número de atributos escolhidos aleatoriamente e *Nº Iterations*, que é o número de árvores.

No método SVM escolheu-se o meta-parâmetro *cost*, que é uma medida de penalização dos casos de observações que se encontram no lugar "errado" da margem, ou seja, os erros da classificação; a função *Kernel*, que define a função usada para mapear os dados num espaço de dimensão superior, considerando-se quatro tipos de função: linear, polinomial, sigmoide e radial. Escolheu-se ainda o *gamma*, que é um parâmetro da função *kernel.* É importante mencionar que se tentou testar o meta-parâmetro *kernel* para a função do tipo polinomial, no entanto, não foi possível a sua utilização devido aos limites de capacidade de processamento

[16] Tradução livre da autora: "confidence factor that represents a threshold of allowed inherent error in data while pruning the decision tree" (Stiglic *et al*., 2012).

dos computadores usados. Para além disso, o meta-parâmetro *gamma* apenas foi testado para todos os seus valores na base de dados da Alemanha. Nesta constatou-se que, independentemente do valor que se configurasse para o parâmetro *gamma*, não haveria qualquer influência no resultado do *output*. Por isso, no estudo da base de dados de Taiwan, apenas se configurou o parâmetro *gamma* para o valor igual a 1.

Na rede neuronal, o meta-parâmetro *Hidden layers* define as camadas ocultas entre neurónios *input* e *output* e quantos nós (neurónios) cada *hidden layer* contém. Um único valor representa apenas uma camada e esse valor dita o número de neurónios em cada camada intermédia. Dois números separados por vírgula representam duas camadas intermédias e cada número representa o número de neurónios que cada camada possui. O *training time* define o número de passagens pela base de dados, com o fim de atualizar os pesos e o *learning rate* é um meta-parâmetro que define a amplitude das variações que ocorrem quando os pesos são atualizados.

Como exemplo da escolha de valores para meta-parâmetros, refira-se que, no método de árvore de classificação, se escolheram os seguintes valores: *Minimum number Instances* com os valores 2, 5 e 8 e o *Confidence Factor* com os valores 0.1, 0.25 e 0.5.

Na Tabela 2 apresenta-se o resumo dos valores testados para os meta-parâmetros selecionados de cada modelo.

Tabela 2. Meta-parâmetros selecionados para análise dos modelos preditivos.

Modelos	Meta-Parâmetros		
Árvores de Classificação	***Minimum number of instances***		***Confidence Factor***
	2 / 5 / 8		0.1 / 0.25 / 0.5
Random Forest	***Max depth***	***Nº features***	***Nº iterations***
	3 / 10 / 20	3 / 5 / 10	100 / 1000 / 2000
Support Vector Machine	***Cost***	***Kernel Type***	***Gamma***
	0.01 / 1 / 1000	Linear/ Radial/Sigmoide	0.0001 / 0.01 / 1
Rede Neuronal	***Hidden layers***	***Training time***	***Learning rate***

	15 / 31 / 15,15 / 31,31	200 / 500 / 1000	0.001 / 0.3 / 0.5

Fonte: Elaboração e Cálculos Próprios.

Resumidamente, podemos indicar que se testou, para o método de árvore de decisão, nove combinações de meta-parâmetros, ou seja, 3^2. Em cada teste de combinações de meta-parâmetros de cada método teve-se em atenção a matriz confusão obtida por cada output e o cálculo de uma função custo, a que posteriormente se dará particular atenção.

Em cada base de dados, mais especificamente na subamostra treino, cada modelo foi treinado com todas as suas combinações possíveis dos diversos valores dos meta-parâmetros apresentados na Tabela 2. Posteriormente, reuniram-se todos os desempenhos dos diversos meta-parâmetros de cada método calculando-se a função custo.

4.3. Matriz Confusão

A matriz confusão avalia o desempenho do método testado, e, também, permite o cálculo de outras medidas importantes, como: *accuracy* ou precisão, a função custo, entre outras (Haldankar & Bhowmick, 2016).

Uma matriz confusão possui (*n* x *n*) dimensões, em que *n* é o número de classes diferentes. Geralmente, numa matriz confusão, as linhas representam o valor das classes reais e as colunas o valor das classes previstas.

Tabela 3. Matriz Confusão.

		Previsão	
		Classe = 1 (Positiva)	Classe = 0 (Negativa)
Atual (Valores verdadeiros)	Classe = 1 (Positiva)	Verdadeiro Positivo (VP)	Falso Negativo (FN) **Tipo 2**
	Classe = 0 (Negativa)	Falso Positivo (FP) **Tipo 1**	Verdadeiro Negativo (VN)

Fonte: Elaboração Própria.

Na Tabela 3, procede-se ao exemplo geral de uma matriz de duas classes (classe 0 e 1), em que na diagonal principal se encontram os números de instâncias corretamente classificadas, que são os VP e VN. Na diagonal secundária encontram-se os tipos de erros, ou seja, os números de instâncias incorretamente classificadas. Por exemplo, no caso do erro do tipo 2, um falso negativo sucede quando uma instância é classificada/prevista incorretamente como classe negativa (classe 0), quando na realidade pertence à classe positiva ou classe 1.

No caso de um erro do tipo 1, falso positivo, ocorre quando o resultado é incorretamente previsto como 1 (classe positiva) e na realidade a classe correta seria a classe 0, a classe negativa.

Tendo em conta as denominações apresentadas na Tabela anterior, a medida *accuracy* é calcula a partir de (VP+VN) / (VP+VN+FN+FP), ou seja, é a frequência com que o método prevê o resultado correto.

Após a apresentação da matriz confusão, aplicar-se-á este conceito às bases de dados estudadas neste trabalho. A base de dados da Alemanha apresenta duas classes: a classe 1 (também designada por positiva), que representa os clientes que constituem um bom risco de crédito para o banco e a classe 2 (negativa), os clientes que representam um mau risco de crédito para o banco.

Na descrição da base de dados da Alemanha é sugerida uma aplicação de custos, e optou-se por incorporar esses custos na análise dos modelos preditivos. Nomeadamente, é sugerido o peso 1 para os erros do tipo 2, ou seja, para as instâncias em que os clientes constituem um bom risco de crédito para o banco, mas são incorretamente classificados como um mau risco de crédito para este. É também sugerido o peso de 5 para o tipo de erro 1, isto é, para as instâncias em que os clientes na realidade constituem um mau risco de crédito para o banco, no entanto, são previstos como bom risco de crédito para este. Fará sentido diferentes pesos em cada tipo de erro? Sim. Para as instituições financeiras é pior classificar um mau devedor como um bom devedor do que classificar um bom devedor como um mau devedor, visto que conceder crédito a um cliente que vai entrar em incumprimento é mais arriscado para o banco do que recusar crédito a um cliente que cumprisse o pagamento do crédito. Por isso, os diferentes erros de classificação têm pesos diferentes. Este é também um dos motivos pelos quais a função custo é tão importante.

Na Tabela 4 encontra-se uma matriz custo da base de dados da Alemanha com os respetivos pesos atribuídos aos tipos de erros, em que nas instâncias corretamente classificadas não se atribui nenhum peso.

Tabela 4. Matriz Custo da Base de Dados da Alemanha.

Base de dados da Alemanha		**Previsão**	
		Classe = 1 (Bom Risco de Crédito)	Classe = 2 (Mau Risco de Crédito)
Atual (Valores verdadeiros)	Classe = 1 (Bom Risco de Crédito)	0	1
	Classe = 2 (Mau Risco de Crédito)	5	0

Fonte: Elaboração Própria.

Para incorporar os pesos referidos anteriormente, na avaliação dos modelos foi necessário recorrer a um método do programa *WEKA*, o método *cost sensitive classifier*. O propósito deste método é atribuir custos diferentes aos diferentes tipos de erros e construir um modelo com o mínimo custo relativo às instâncias classificadas incorretamente (Haldankar e Bhowmick, 2016).

Quanto a base de dados do Taiwan decidiu-se atribuir os mesmos pesos sugeridos na base de dados da Alemanha, para os erros com caraterísticas semelhantes. Todavia, a matriz de custo do Taiwan é inversa, porque as classes estão definidas de forma diferente, tal como se pode verificar na Tabela 5. A classe igual a 0 corresponde ao cumprimento de pagamentos em cartões de crédito por parte do cliente e a classe igual a 1 corresponde ao incumprimento de pagamentos por parte do cliente.

Tabela 5. Matriz Custo da Base de Dados Taiwan.

Base de dados de Taiwan		**Previsão**	
		Classe = 1 (Clientes Incumpridores)	Classe = 0 (Clientes Cumpridores)
Atual (Valores verdadeiros)	Classe = 1 (Clientes Incumpridores)	0	5
	Classe = 0 (Clientes Cumpridores)	1	0

Fonte: Elaboração Própria.

4.3. Problema do *Overfitting*

Os modelos analisados neste trabalho utilizam meta-parâmetros que definem a sua estrutura. Os valores dos meta-parâmetros que produzem melhores resultados podem levar a resultados "excessivamente" bons, devido a erros aleatórios. Em causa está um problema usual do *Machine Learning*, denominado por *overfitting*. Este problema ocorre quando se utilizam algoritmos baseados em estruturas flexíveis, que permitem um ajustamento muito forte aos dados que são utilizados para os treinar. Este ajustamento pode não se dever a uma efetiva capacidade de explicação do comportamento dos dados, mas apenas à flexibilidade da estrutura utilizada – o algoritmo está a ajustar-se mais a componentes aleatórias (como por exemplo os ruídos) dos dados do que à verdadeira estrutura subjacente a estes. Assim, quando o modelo estimado é aplicado a uma nova amostra, a sua capacidade de previsão não será boa.

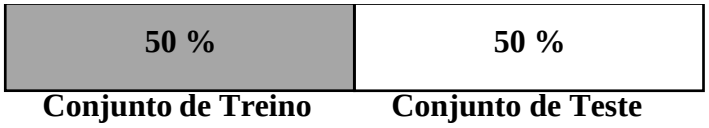
Segundo Rokach e Mainon (2008), *overfitting* "é a situação em que o algoritmo de indução gera um classificador que ajusta perfeitamente os dados de treino, mas que perdeu a capacidade de

generalizar para instâncias não apresentadas durante o treino"[17]. Por outras palavras, criou-se um modelo que conhece muito bem o conjunto de treino, mas não poderá prever corretamente um novo conjunto de dados que esteja fora do conjunto de treino, visto que, o classificador apenas memorizou as instâncias de treino.

Devido ao *overfitting* obtém-se uma capacidade de previsão elevada no conjunto de treino. No entanto, os modelos não possuem realmente essa capacidade de previsão, o que acontece é um ajustamento excessivo aos dados. De certa forma, este "excesso de otimismo" ou elevada capacidade de previsão de modelos pode ser corrigido por duas técnicas: a técnica *K-Fold Cross-Validation* (CV) ou validação cruzada e a divisão da amostra.

4.4.1. Divisão da Base de dados: Treino e Teste

Figura 7. Separação da base de dados total em dois subconjuntos: Treino e Teste.



Fonte: Elaboração Própria.

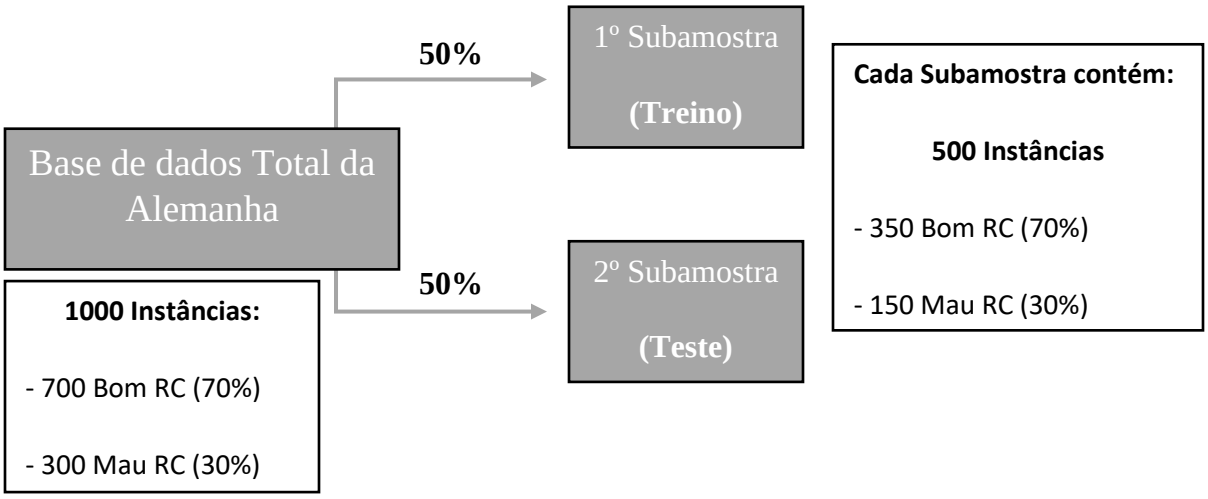
Porque é que a divisão da base de dados poderá ser uma tentativa de evitar o problema de *overfitting*? A divisão da amostra permite garantir que a escolha dos meta-parâmetros não é enviesada para os meta-parâmetros que apresentam melhores resultados devido a motivos puramente aleatórios.

De forma a pôr em prática esta técnica procedeu-se a divisão aleatória de cada base de dados original em duas subamostras de igual dimensão: numa primeira subamostra, treino, e numa segunda subamostra, teste (representado na Figura 7).

[17] Tradução livre da autora. No original "*It refers to the situation in which the induction algorithm generates a classifier which perfectly fits the training data but has lost the capability of generalizing to instances not presented during training*." (Rokach e Mainon, 2008)

Além disso, cada subamostra é representativa da amostra total, ou seja, a percentagem de elementos de cada classe em cada subamostra é idêntica ao seu peso na amostra total, tal como se pode verificar nas Figuras 8 e 9, respetivamente para a Alemanha e para Taiwan.

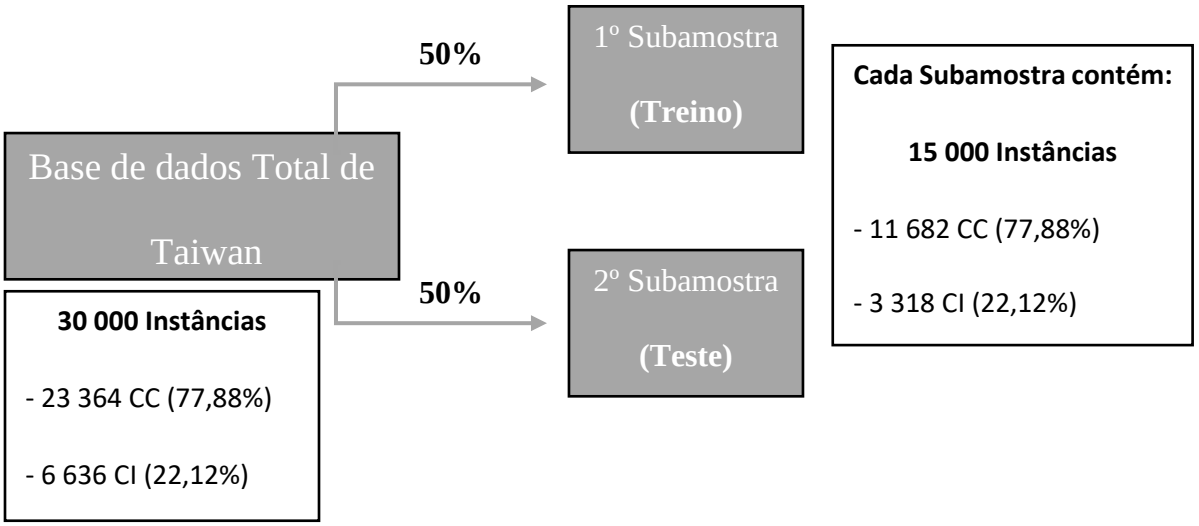
Figura 8. Divisão da Base de dados da Alemanha em duas subamostras: Treino e Teste.



Fonte: Elaboração Própria. Nota: Risco de Crédito é representado pela abreviatura de RC.

Através das Figuras 8 e 9 verificamos que, tanto a base de dados da Alemanha, como a base de dados Taiwan não são balanceadas, isto significa que ambas contêm mais instâncias de uma determinada classe. Neste caso, ambas possuem mais instâncias representativas da classe de bons devedores: a amostra de Taiwan é constituída por 77,88% de instâncias referentes a clientes cumpridores (ou seja, 23 364 em 30 000 instâncias representativas da classe 0) e a amostra da Alemanha tem 70% de instâncias referentes a clientes com bom RC (isto é, 700 de 1000 instâncias da classe 1).

Figura 9. Divisão da Base de dados de Taiwan em duas subamostras: Treino e Teste.



Fonte: Elaboração Própria. Nota: Clientes Cumpridores são representados pela abreviatura CC e Clientes Incumpridores por CI.

Na subamostra treino, cada modelo foi treinado com as diferentes combinações dos diversos valores dos meta-parâmetros. Posteriormente, os melhores conjuntos de meta parâmetros obtidos em cada modelo preditivo da subamostra treino serão usados na nova segunda subamostra de dados, de teste. A escolha das melhores combinações de meta-parâmetros teve em conta a tentativa de não repetir os mesmos meta-parâmetros nas três melhores combinações de cada modelo. Na seção 5.1 abordar-se-á com mais pormenor os procedimentos e testes realizados.

4.4.2. *Cross-Validation*

A técnica de Validação Cruzada ou *Cross-Validation* (CV) garante que os modelos são aplicados em novos dados diferentes dos utilizados para os treinar.

O único parâmetro deste processo é o *K-Fold*, que é o número de vezes em que uma amostra original é dividida aleatoriamente, em *k* subconjuntos ou *k folds* iguais. Também é o número de vezes que se realiza validação cruzada, tal como se irá descrever.

Podem-se considerar diversos valores para o parâmetro *k*, porém na análise de todos os modelos preditivos utilizados nas duas bases de dados deste trabalho considerou-se *k* igual a 4 *folds*.

De seguida, tendo em conta que o parâmetro *k*=4, descreve-se o procedimento de CV:

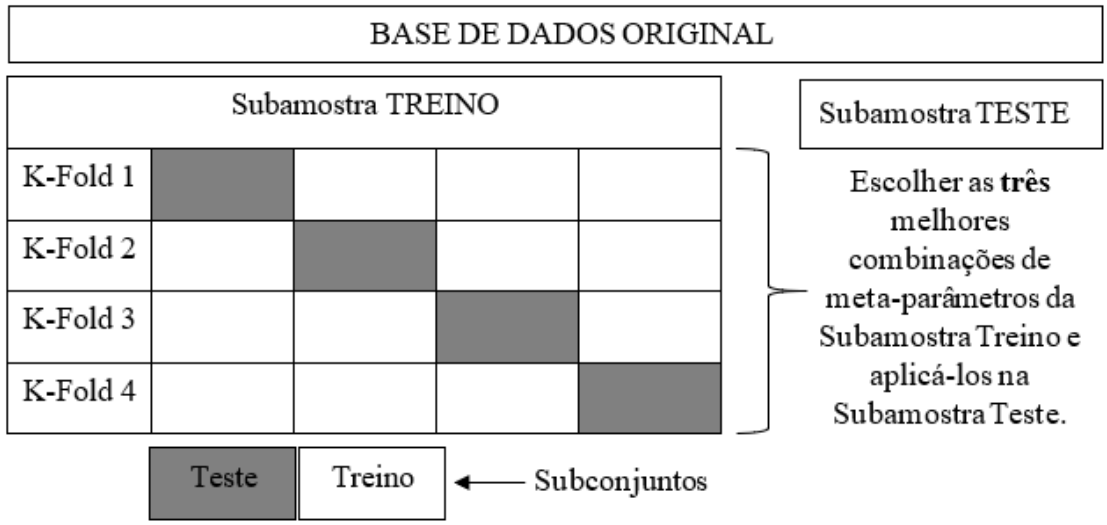
1º Dividiu-se a subamostra treino em 4 *folds* ou 4 subconjuntos iguais.

2º Existem sempre (k-1 = 3) *folds* que irão constituir um subconjunto de treino e 1 a subconjunto de teste.

3º Treinou-se sempre usando o subconjunto de treino (composto por 3 *folds*) e validou-se no subconjunto que ficou de fora, o subconjunto de teste.

4º Realizou-se este processo de *Cross-validation* para todas as combinações possíveis dos dois subconjuntos. Na figura 10 verifica-se que na primeira iteração, o treino usa os últimos subconjuntos e o teste é feito no primeiro subconjunto. Nas iterações posteriores é usada uma lógica semelhante. Nessa figura, observa-se ainda um quadrado denominado teste e outro denominado treino, que representam os subconjuntos.

Figura 10. Ilustração da Técnica *Cross-Validation* (K=4).



Fonte: Elaboração Própria.

Tal como está representado na Figura acima, é essencial relembrar que cada base de dados analisada neste trabalho foi dividida aleatoriamente em duas subamostras: treino e teste.

5. Resultados

Antes de descrever os resultados obtidos na análise, é necessário abordar quais os procedimentos adotados.

5.1. Procedimentos e Testes

Anteriormente, referiu-se que a matriz confusão permitiu calcular a função custo. Portanto, após a recolha de todas as matrizes confusão relativas a todas as combinações de meta-parâmetros de cada subamostra de cada base de dados, calculou-se a respetiva função custo para cada combinação de meta-parâmetros. A função custo é calculada a partir da ponderação de cada peso pelo respetivo erro, ou seja, pelo respetivo número de clientes classificados incorretamente noutra classe. Por exemplo, utilizando a denominação das variáveis apresentadas na Tabela 3 e assumindo os pesos atribuídos na base de dados da Alemanha, a fórmula geral da função custo seria 5x FP + 1xFN.

Esta função é fundamental por dois motivos. Um dos motivos refere-se à diferença de importância, para os bancos, dos dois tipos de erros, tal como foi descrito na seção 4.2. O outro motivo, já referido brevemente na seção 4.4.1., é porque as duas amostras analisadas neste trabalho não são balanceadas, ou seja, possuem mais instâncias de uma determinada classe. Aliás a base de dados do Taiwan é altamente não balanceada. Portanto, se atribuíssemos um peso igual para cada classe de cada amostra, haveria uma probabilidade forte de se obter uma taxa de precisão elevada simplesmente prevendo que todos os clientes são cumpridores, porque a percentagem de instâncias em que os clientes são considerados bons cumpridores ou que constituem um bom risco de crédito representam a maioria dos casos em cada amostra.

Desta forma, a utilização da função custo ou a atribuição de pesos para as instâncias que representam a classe com menos registos nas amostras permite contornar o problema do não balanceamento.

Em suma, reuniu-se toda a informação e analisou-se todas as funções custo das primeiras subamostras com o objetivo de selecionar apenas três combinações de meta-parâmetros por modelo. No apêndice C apresentam-se todos os resultados de todas as combinações de meta-parâmetros de cada modelo nas primeiras subamostras, antes da seleção das combinações. A seleção foi realizada com base em dois critérios: em primeiro lugar consideraram-se as

combinações em que se obteve um menor valor da função custo, em segundo lugar procurou-se evitar em escolher combinações que repitam os valores de meta-parâmetros. Obtiveram-se nas primeiras subamostras, as três melhores combinações de diferentes meta-parâmetros para cada um dos modelos considerados, sendo estas apresentadas na Tabela 6 e na Tabela 7.

Tabela 6. Função custo obtida em cada um dos algoritmos, com os melhores meta-parâmetros testados na 1º Subamostra da base de dados da Alemanha, 4-*Folds*.

Árvores de Classificação	
Meta-Parâmetros	**Função Custo**
Min. N. Inst.=2; *Confidence Factor* =0,1	284
Min.N.Inst.=5; *Confidence Factor* =0,1	281
Min.N.Inst.=8; *Confidence Factor* =0,1	**276**

Support Vector Machine	
Meta-Parâmetros	**Função Custo**
Cost =0.01; *Kernel Type* =*Linear* ; *Gamma* =1	**305**
Cost =0.01; *Kernel Type* = *Radial* ; *Gamma* =1	372
Cost =1000; *Kernel Type* =*Linear* ; *Gamma* =1	431

Random Forest	
Meta-Parâmetros	**Função Custo**
Max Dep.=10; Nº Feat.=10 Nº Iter.=100	253
Max Dep.=10; Nº Feat.=10; Nº Iter.=1000	**247**
Max Dep.=10; Nº Feat.=10; Nº Iter.=2000	252

Rede Neuronal	
Meta-Parâmetros	**Função Custo**
Train.Time=1000; Hid.Lay.= 15; Learn.Rt.=0.01	**283**
Train.Time=1000; Hid.Lay.=15, 15; Learn.Rt.=0.01	288
Train.Time=1000; Hid.Lay.=31,31; Learn.Rt.=0.01	301

Fonte: Elaboração e Cálculos Próprios. Nota: Min.N.Inst (*Minimum Number of Instances*); Max Dep.(*Max Depth*); Nº Feat (*NºFeatures*); NºIter.(*NºIterations*); Train. Time (*Training Time*); Hid.Lay. (*Hidden Layers*) e Learn.Rt.(*Learning Rate*).

Tabela 7. Função custo obtida em cada um dos algoritmos, com os melhores-meta parâmetros testados na 1º Subamostra da base de dados de Taiwan, 4-*Folds*.

Árvores de Classificação	
Meta-Parâmetros	**Função Custo**
Min. N. Inst.=2; *Confidence Factor* =0,1	**7 731**
Min.N.Inst.=5 ; *Confidence Factor* =0,1	7 655
Min.N.Inst.=8 ; *Confidence Factor* =0,1	7 585

Support Vector Machine	
Meta-Parâmetros	**Função Custo**
Cost =0.01; *Kernel* =*Linear* ; *Gamma* =1	11 314
Cost =0.01; *Kernel* = *Radial* ; *Gamma* =1	11 682
Cost =1000; *Kernel* =*Linear* ; *Gamma* =1	**11 265**

Random Forest	
Meta-Parâmetros	**Função Custo**
Max Dep.=10; Nº Feat.=10 Nº Iter.=100	6 927
Max Dep.=10; Nº Feat.=10; Nº Iter.=1000	**6 921**
Max Dep.=10; Nº Feat.=10; Nº Iter.=2000	6 940

Rede Neuronal	
Meta-Parâmetros	**Função Custo**
Train.Time=1000; Hid.Lay.= 15 ; Learn.Rt.=0.01	7 923
Train.Time=1000; Hid.Lay.=15, 15 ; Learn.Rt.=0.01	**7 843**
Train.Time=1000; Hid.Lay.=31,31 ; Learn.Rt.=0.01	7 890

Fonte: Elaboração e Cálculos Próprios. Nota: Min.N.Inst (*Minimum Number of Instances*); *Kernel* (*Kernel Type*); Max Dep.(*Max Depth*); Nº Feat (*NºFeatures*); NºIter.(*NºIterations*); Train. Time (*Training Time*); Hid.Lay. (*Hidden Layers*) e Learn.Rt.(*Learning Rate*).

De seguida, na segunda subamostra de cada base de dados testou-se apenas os três melhores meta-parâmetros selecionados anteriormente, apresentando-se os resultados na Tabela 7 e Tabela 8. Adicionalmente, repetiu-se o procedimento semelhante utilizado nas primeiras subamostras, desde a implementação no início do método *cost sensitive classifier*, a recolha das matrizes confusão até ao cálculo da função custo.

Tabela 8. Função custo obtida em cada um dos algoritmos, com os melhores meta-parâmetros testados na 2º Subamostra da base de dados da Alemanha, 4-*Folds*.

Árvores de Classificação	
Meta-Parâmetros	**Função Custo**
Min. N. Inst.=2; Confidence Factor=0,1	312
Min.N.Inst.=5; Confidence Factor=0,1	**257**
Min.N.Inst.=8; Confidence Factor=0,1	322

Support Vector Machine	
Meta-Parâmetros	**Função Custo**
Cost=0.01; Kernel Type=Linear; Gamma=1	**313**
Cost=0.01; Kernel Type= Radial; Gamma=1	391
Cost=1000; Kernel Type=Linear; Gamma=1	497

Random Forest	
Meta-Parâmetros	**Função Custo**
Max Dep.=10; Nº Feat.=10 Nº Iter.=100	**299**
Max Dep.=10; Nº Feat.=10; Nº Iter.=1000	318
Max Dep.=10; Nº Feat.=10; Nº Iter.=2000	321

Rede Neuronal	
Meta-Parâmetros	**Função Custo**
Train.Time=1000; Hid.Lay.= 15; Learn.Rt.=0.01	328
Train.Time=1000; Hid.Lay.=15, 15; Learn.Rt.=0.01	**320**
Train.Time=1000; Hid.Lay.=31,31; Learn.Rt.=0.01	325

Fonte: Elaboração e Cálculos Próprios. Nota: Min.N.Inst (*Minimum Number of Instances*); Max Dep.(*Max Depth*); Nº Feat (*NºFeatures*); NºIter.(*NºIterations*); Train. Time (*Training Time*); Hid.Lay. (*Hidden Layers*) e Learn.Rt.(*Learning Rate*).

Tabela 9. Função custo obtida em cada um dos algoritmos, com os melhores meta-parâmetros testados na 2º Subamostra da base de dados de Taiwan, 4-*Folds*.

Árvores de Classificação	
Meta-Parâmetros	**Função Custo**
Min. N. Inst.=2; *Confidence Factor* =0,1	7 805
Min.N.Inst.=5; *Confidence Factor* =0,1	**7 579**
Min.N.Inst.=8; *Confidence Factor* =0,1	7 608

Support Vector Machine	
Meta-Parâmetros	**Função Custo**
Cost =0.01; *Kernel* =*Linear* ; *Gamma* =1	15 498
Cost =0.01; *Kernel* = *Radial* ; *Gamma* =1	**11 682**
Cost =1000; *Kernel* =*Linear* ; *Gamma* =1	13 053

Random Forest	
Meta-Parâmetros	**Função Custo**
Max Dep.=10; Nº Feat.=10 Nº Iter.=100	**6 856**
Max Dep.=10; Nº Feat.=10; Nº Iter.=1000	6 865
Max Dep.=10; Nº Feat.=10; Nº Iter.=2000	6 891

Rede Neuronal	
Meta-Parâmetros	**Função Custo**
Train.Time=1000; Hid.Lay.= 15; Learn.Rt.=0.01	8 027
Train.Time=1000; Hid.Lay.=15, 15; Learn.Rt.=0.01	**8 008**
Train.Time=1000; Hid.Lay.=31,31; Learn.Rt.=0.01	8 146

Fonte: Elaboração e Cálculos Próprios. Nota: Min.N.Inst (*Minimum Number of Instances);* Kernel (*Kernel Type*); Max Dep.(*Max Depth*)*; Nº Feat* (*NºFeatures*); NºIter.(*NºIterations*); Train. Time (*Training Time*)*; Hid.Lay.* (*Hidden Layers)* e Learn.Rt.(*Learning Rate*).

Na segunda subamostra, para além de se analisar cada modelo individualmente, para cada cliente específico, também se realizou uma combinação de todos os modelos de forma a obter uma única previsão e com o objetivo de melhorar a capacidade de predição da classificação dos algoritmos de *Machine Learning*. Por outras palavras, referimo-nos a técnica *Ensemble*, ou seja, a combinação de previsões de vários algoritmos /classificadores com intuito de obter um melhor resultado em conjunto.

Breiman (1996) afirmou que muitos resultados experimentais também mostram que os resultados de classificação em *Ensemble* são frequentemente superiores aos resultados do melhor classificador individual"[18]. Pretende-se assim verificar se o *Ensemble* supera ou não a previsão ou o desempenho do melhor modelo individual. Para isso, realizaram-se alguns procedimentos com a finalidade de criar uma matriz custo que permitisse a comparação do *Ensemble* com os modelos individuais anteriormente analisados.

Por fim, é possível observar na Tabela 10, os resultados obtidos a partir da matriz criada pelo *Ensemble* em cada uma das segundas subamostras de cada base de dados.

Tabela 10. Resultados finais do *Ensemble* em cada 2º subamostra (teste).

	Ensemble **Taiwan**	*Ensemble* **Alemanha**
Função Custo	6151	310

Fonte: Elaboração e Cálculos Próprios.

Resumidamente, o procedimento consistiu em analisar, para cada elemento da segunda subamostra, a previsão gerada pelos modelos obtidos a partir dos meta-parâmetros que

[18] Tradução livre da autora. No original "*Many experimental results also show that classification results of ensembles are often superior to those of the single best classifier*" (Breiman, 1996).

conduziram a melhores resultados (os modelos utilizados são os evidenciados na Tabela 1). Depois verificou-se qual era a classe que a maioria dos modelos previa e de seguida definiu-se a previsão final do *Ensemble* (Ver Apêndice D).

6. Análise de Resultados

Nesta seção, dar-se-á particular atenção apenas ao melhor desempenho dos melhores meta-parâmetros de cada modelo. Antes de mais, é importante mencionar que os modelos que exibem um menor valor da função custo são aqueles que apresentam um melhor desempenho. Começando pela análise individual de cada modelo nas primeiras subamostras, podemos concluir que os modelos que obtém melhores desempenhos são as AC e as RF. Não só estes dois modelos obtém as melhores classificações nas primeiras subamostras treino, como também é notório o seu desempenho nas segundas subamostras teste. Observa-se que o modelo *Random Forest* se destaca como o modelo que tem, globalmente, os melhores resultados. Efetivamente, verifica-se que os melhores resultados individuais obtidos nas primeiras subamostras são os seguintes valores da função custo: 247 na Alemanha e 6 921 no Taiwan, os quais dizem respeito ao mesmo modelo, *Random Forest* e a mesma combinação de meta-parâmetros. O modelo de árvore classificação sobressai unicamente com um valor de 257 obtido na função custo da melhor combinação de meta-parâmetros na segunda subamostra da Alemanha, comparativamente com o valor da função custo de 299 da RF.

Em contraste, os modelos de SVM são os modelos com o pior desempenho quer na subamostra de treino quer na subamostra de teste. No entanto é de notar que a melhor combinação de meta-parâmetros do modelo SVM da subamostra teste da Alemanha foge a este resultado, obtendo um valor de 313 para a função custo. Este é o único valor melhor do que o apresentando pela melhor combinação de meta-parâmetros do modelo da Rede Neuronal, com um custo de 320. Perante a análise da segunda subamostra da Alemanha verifica-se que o valor obtido na função custo do *Ensemble* foi de 310. Apesar do valor ser inferior ao desempenho da maioria dos melhores modelos individuais, é superior ao valor 257 da função custo obtido através do melhor conjunto de meta-parâmetros do modelo individual AC e do valor 299 do melhor conjunto de meta-parâmetros do modelo individual RF. Relativamente a todos os outros resultados dos modelos, que foram piores, estes dois únicos valores podem ser um resultado excecional, que se destaca devido a efeitos aleatórios (um caso de "sorte"). Tendo em conta o valor obtido do *Ensemble* da segunda subamostra da base de Taiwan, que é 6 151, verifica-se que este é substancialmente inferior a função custo do modelo *Random Forest*, o melhor modelo individual

obtido na segunda subamostra, em que se obteve o valor de 6 856, e também inferior ao desempenho de todos os melhores modelos individuais.

7. Conclusão

Neste trabalho começou-se por realizar uma análise global da literatura existente relativamente a avaliação de risco de crédito, em que constatou que inicialmente se utilizavam métodos mais tradicionais, como *Logit*, *Probit*, Análise Discriminante Linear e *AC*. No entanto, atualmente dá-se particular atenção a métodos mais sofisticados, como os métodos de *Machine Learning*.

O presente trabalho de projeto tem como objetivo de comparar os diferentes modelos de *Machine Learning*, a fim de decidir qual dos métodos e quais os modelos que evidenciam um melhor desempenho na avaliação do risco de crédito. Considera-se ainda uma técnica de *Ensemble* que combina os resultados dos vários modelos.

Pode concluir-se a partir da análise concretizada que o modelo de RF é o modelo que, em geral, apresenta os melhores resultados, nas duas bases de dados. Contrariamente, o modelo SVM evidencia os piores desempenhos na análise dos dados.

O resultado empírico mais notório deste trabalho e que sobressai entre todos os resultados obtidos é que a técnica *Ensemble* permite produzir, em geral, uma melhor previsão, quer no caso da base de dados de Taiwan, quer no caso da base de dados da Alemanha.

Efetivamente, a utilização da técnica *Ensemble* no estudo da avaliação do risco de crédito é evidentemente uma boa opção para se aplicar nos modelos de *Credit Scoring*.

Bibliografia

Altman, E. I., & Saunders, A. (1997). Credit risk measurement: Developments over the last 20 years. *Journal of banking & finance*, *21*(11-12), 1721-1742.

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, *54*(6), 627-635.

Bequé, A., & Lessmann, S. (2017). Extreme learning machines for credit scoring: An empirical evaluation. *Expert Systems with Applications*, *86*, 42-53.

Biggs, D., De Ville, B., & Suen, E. (1991). A method of choosing multiway partitions for classification and decision trees. *Journal of applied statistics*, *18*(1), 49-62.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446-3453.

Brownlee, J. (2016). *Master Machine Learning Algorithms: discover how they work and implement them from scratch*. Machine Learning Mastery.

Chye, K. H., Chin, T. W., & Peng, G. C. (2004). Credit scoring using data mining techniques. *Singapore Management Review*, *26*(2), 25-48.

West, D. (2000). Neural network credit scoring models. *Computers & Operations Research*, *27*(11-12), 1131-1152.

Galindo, J., & Tamayo, P. (2000). Credit risk assessment using statistical and machine learning: basic methodology and risk modeling applications. *Computational Economics*, *15*(1-2), 107-143.

Haldankar, A. N., & Bhowmick, K. (2016, December). A cost sensitive classifier for Big Data. In *2016 IEEE International Conference on Advances in Electronics, Communication and Computer Technology (ICAECCT)* (pp. 122-127). IEEE.

Han, J., Pei, J., & Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

K. Bache, M. Lichman, UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Louzada, F., Ara, A., & Fernandes, G. B. (2016). Classification methods applied to credit scoring: Systematic review and overall comparison. *Surveys in Operations Research and Management Science*, *21*(2), 117-134.

Malekipirbazari, M., & Aksakalli, V. (2015). Risk assessment in social lending via random forests. *Expert Systems with Applications*, *42*(10), 4621-4631.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*(4), 115-133.

Mester, L. J. (1997). What's the point of credit scoring?. *Business review*, *3*(Sep/Oct), 3-16.

Rokach, L., & Maimon, O. Z. (2008). *Data mining with decision trees: theory and applications* (Vol. 69). World scientific. guent

Stiglic, G., Kocbek, S., Pernek, I., & Kokol, P. (2012). Comprehensive decision tree models in bioinformatics. *PloS one*, *7*(3).

Thomas, L. C., Edelman, D. B., & Crook, J. N. (2002). *Credit scoring and its applications*. Society for industrial and Applied Mathematics.

Torgo, L. (2011). *Data mining with R: learning with case studies*. Chapman and Hall/CRC.

Tsai, C. F., & Wu, J. W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert systems with applications*, *34*(4), 2639-2649.

Wang, G., Hao, J., Ma, J., & Jiang, H. (2011). A comparative assessment of ensemble learning for credit scoring. *Expert systems with applications*, *38*(1), 223-230.

Wang, G., Ma, J., Huang, L., & Xu, K. (2012). Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems*, *26*, 61-68.

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

Zhao, Y., & Zhang, Y. (2008). Comparison of decision tree methods for finding active objects. *Advances in Space Research*, *41*(12), 1955-1959.

Apêndices

A. Descrição da base de dados de Taiwan

A base de dados de Taiwan é constituída por 23 atributos, dos quais 20 são atributos categóricos e 3 são atributos numéricos. Possui ainda uma variável binária.

Tabela 11 Descrição dos atributos da base de dados da Alemanha.

AT	Designação de Atributos	Desagregação de Atributos Principais
AT1	**Montante de crédito concedido**	Inclui o crédito ao cliente e a família. **(Em NT dólar)** Nota: inclui o consumo de crédito individual do indivíduo e da sua família.
AT2	**Gênero**	1= Masculino, 2=Feminino.
AT3	**Educação**	1=Pós-Graduação, 2=Universidade, 3=Escola Secundária, 4=Outros.
AT4	**Estado Civil**	1=Casado, 2=Solteiro, 3=Outros.
AT5	**Idade**	(Em anos)
AT6 - AT11	**Histórico Mensal de Pagamentos Anteriores** **(De Abril a Setembro de 2005)**	AT6= Estado do reembolso em Setembro de 2005; AT7= Estado do reembolso em Agosto de 2005; …. AT11= Estado do reembolso em Abril de 2005; **A escala de medição para o estado do reembolso:** 0= Pagamento não está atrasado; 1= Pagamento atrasado por um mês; … 9= Pagamento atrasado por 9 meses ou mais.
AT12 - AT17	**Valor mensal dos extratos das faturas** **(De Abril a Setembro de 2005)**	AT12= Valor do extrato da conta em Setembro de 2005; … AT17=Valor do extrato da conta em Abril de 2005. **(Em NT dólar)**

AT18 - AT23	**Valor do pagamento anteriores (De Abril a Setembro de 2005)**	AT18=Valor pago em Setembro de 2005; … AT23 =Valor pago em Abril de 2005. **(Em NT dólar)**
Incumprimento de Pagamentos (Variável Binária)		**Classe 0**= Cumprimento, **Classe 1**= Incumprimento.

Fonte: Elaboração Própria, baseada na informação contida no repositório https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients. Nota: Todos os atributos numéricos apresentam-se no Novo dólar Taiwanês (NT).

B. Descrição da base de dados da Alemanha

A base de dados da Alemanha é constituída por 20 atributos, dos quais 13 são atributos categóricos e 7 são atributos numéricos. Possui ainda uma variável binária.

Tabela 12. Descrição dos atributos da base de dados de Taiwan.

Atributo(AT)	**Designação de Atributos**	**Desagregação de Atributos Principais**
AT1 (Qualitativo)	**Estado da Conta Corrente**	A11: … < 0 DM*; A12: 0 <= … < 200 DM; A13: … >= 200DM A14: Não possui conta corrente.
AT2 (Numérico)	**Duração (em meses) da conta no banco**	Valor > 0
AT3 (Qualitativo)	**Histórico de Crédito**	A30: Nenhum crédito retirado/Todos os créditos devolvidos devidamente; A31: Todos os créditos devidamente pagos neste banco; A32: Créditos existentes devolvidos devidamente até agora; A33: Atraso no pagamento no passado; A34: Conta crítica/ Outros créditos existentes (não neste banco).
AT4 (Qualitativo)	**Motivo do Crédito**	A40: Carro (novo); A41: Carro (usado);

		A42: Mobiliário/Equipamentos; A43: Rádio/Televisão; A44: Aparelhos Domésticas; A45: Reparações; A46: Educação; A47: Férias; A48: Reciclagem; A49 Negócios; A50: Outros.
AT5 (Numérico)	**Montante de Crédito (total)**	Valor > 0
AT6 (Qualitativo)	**Conta Poupança/Títulos**	A61: Saldo < 100 DM; A62: 100 DM<=Saldo<500 DM; A63: 500 DM<=Saldo<1000 DM; A64: Saldo >= 1000 DM; A65: Não possui conta poupança.
AT7 (Qualitativo)	**Presente Emprego (desde quando)**	A71: Desempregado; A72: Tempo<1 ano; A73: 1<= Tempo<4 anos; A74: 4<=Tempo<7 anos; A75: Tempo>=7 anos.
AT8 (Numérico)	**Taxa de prestações** **(em % rendimento disponível)**	Valor > 0
AT9 (Qualitativo)	**Estado Civil e Sexo**	A91: Masculino: Divorciado/ Separado; A92: Feminino: Divorciada/Separada/Casada; A93: Masculino e Solteiro; A94: Masculino: Casado/Viúvo; A95: Feminino e Solteira.
AT10 (Qualitativo)	**Outros fiadores (garantias)**	A101: Nenhum; A102: Co-candidato; A103: Fiador.
AT11 (Numérico)	**Duração da atual residência**	Valor > 0
AT12 (Qualitativo)	**Propriedade**	A121: Imóveis; A122: Se não estiver A121: Construção de acordo de sociedade de poupança; Seguro de vida; A123: Se não estiver A121 e A122:

		Carro ou outro; A124: Desconhecido/Sem propriedade.
AT13 (Numérico)	**Idade**	Valor (em anos) > 0
AT14 (Qualitativo)	**Outros planos de prestações**	A141: Banco; A142: Lojas; A143: Nenhum.
AT15 (Qualitativo)	**Habitação**	A151: Alugada; A152: Própria; A153: Gratuita.
AT16 (Numérico)	**Nº Créditos existentes neste banco**	Valor > 0
AT17 (Qualitativo)	**Emprego**	A171: Desempregado/Não Qualificado-Não Residente; A172: Não qualificado-Residente; A173: Funcionário Qualificado; A174: Administrador/ Conta Própria/ Funcionário Altamente Qualificado.
AT18 (Numérico)	**Nº de pessoas responsáveis por fornecer manutenção**	Valor > 0
AT19 (Qualitativo)	**Telefone**	A191: Nenhum; A192: Sim, registado na ficha do cliente.
AT20 (Qualitativo)	**Trabalhador Estrangeiro**	A201: Sim; A202: Não.
Risco de Crédito (Variável Binária)		**Classe 1**= Bom devedor, **Classe 2**= Mau devedor.

Fonte: Elaboração Própria, baseada na informação contida no repositório https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data). Nota: **Deusche Mark* (Marco Alemão).

C. Resumo dos resultados obtidos com todas as combinações de meta-parâmetros, nas primeiras subamostras

De seguida, apresenta-se os resultados obtidos de todas as combinações de meta-parâmetros em todos os modelos, nas primeiras subamostras de cada base de dados, por forma a analisar todos os resultados obtidos e a seleção das três melhores combinações em cada modelo.

Tabela 13. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método *Support Vector Machine*, na 1º Subamostra da Alemanha.

Support Vector Machine	Matriz Confusão		Função Custo
C= 0,01; K=LI; G=0,0001	75	275	**305**
	6	144	
C= 0,01; K=LI; G=0,01	75	275	305
	6	144	
C= 0,01; K=LI; G=1	75	275	305
	6	144	
C= 0,01; K=Rbf; G=0,0001	0	350	350
	0	150	
C= 0,01; K=Rbf; G=0,01	0	350	350
	0	150	
C= 0,01; K=Rbf; G=1	0	350	350
	0	150	
C= 0,01; K=Sig; G=0,0001	0	350	350
	0	150	
C= 0,01; K=Sig; G=0,01	0	350	350
	0	150	
C= 0,01; K=Sig; G=1	0	350	350
	0	150	
C= 1; K=LI; G=0,0001	203	147	372
	45	105	
C= 1; K=LI; G=0,01	203	147	372
	45	105	
C= 1; K=LI; G=1	203	147	**372**
	45	105	
C= 1; K=Rbf; G=0,0001	95	255	445
	38	112	
C= 1; K=Rbf; G=0,01	52	298	458
	32	118	
C= 1; K=Rbf; G=1	0	350	350
	0	150	

Support Vector Machine	Matriz Confusão		Função Custo
C= 1; K=Sig; G=0,0001	0	350	350
	0	150	
C= 1; K=Sig; G=0,01	0	350	350
	0	150	
C= 1; K=Sig; G=1	0	350	350
	0	150	
C= 1000; K=LI; G=0,0001	154	196	**431**
	47	103	
C= 1000; K=LI; G=0,01	154	196	431
	47	103	
C= 1000; K=LI; G=1	154	196	431
	47	103	
C= 1000; K=Rbf; G=0,0001	189	161	551
	78	72	
C= 1000; K=Rbf; G=0,01	80	270	455
	37	113	
C= 1000; K=Rbf; G=1	0	350	350
	0	150	
C= 1000; K=Sig; G=0,0001	0	350	350
	0	150	
C= 1000; K=Sig; G=0,01	0	350	350
	0	150	
C= 1000; K=Sig; G=1	0	350	350
	0	150	

*Fonte: Elaboração e Cálculos Próprios. Nota: C(*Cost*); K (*Kernel Type*); G(*Gamma*); LI(*Linear*); Rbf (*Radial Basis Function*); Sig (*Sigmoid*).*

Tabela 14. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método Árvore de Classificação, na 1º Subamostra da Alemanha.

Árvore de Classificação (J4.8)	Matriz Confusão		Função Custo
MinO.=2 ; Conf.F=0,1	171	179	
	21	129	**284**
MinO.=2 ; Conf.F=0,25	182	168	
	25	125	293
MinO.=2 ; Conf.F=0,5	204	146	
	27	123	**281**
MinO.=5 ; Conf.F=0,1	168	182	
	21	129	287
MinO.=5 ; Conf.F=0,25	179	171	
	21	129	**276**
MinO.=5 ; Conf.F=0,5	182	168	
	25	125	293
MinO.=8 ; Conf.F=0,1	165	185	
	21	129	290
MinO.=8 ; Conf.F=0,25	168	182	
	22	128	292
MinO.=8 ; Conf.F=0,5	170	180	
	24	126	300

Fonte: Elaboração e Cálculos Próprios. Nota: Min.N.Inst. (*Minimum Number of Instances*); Conf.F.(*Confidence Factor*).

Tabela 15. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método *Random Forest*, na 1º Subamostra da Alemanha.

Random Forest	Matriz Confusão		Função Custo
MaxD= 3; Feat=3; Itera.=100	126	224	279
	11	139	
MaxD= 3; Feat=3; Itera.=1000	130	220	260
	8	142	
MaxD= 3; Feat=3; Itera.=2000	129	221	261
	8	142	
MaxD= 3; Feat=5; Itera.=100	172	178	**253**
	15	135	
MaxD= 3; Feat=5; Itera.=1000	169	181	266
	17	133	
MaxD= 3; Feat=5; Itera.=2000	169	181	261
	16	134	
MaxD= 3; Feat=10; Itera.=100	201	149	269
	24	126	
MaxD= 3; Feat=10; Itera.=1000	206	144	264
	24	126	
MaxD= 3; Feat=10; Itera.=2000	206	144	264
	24	126	
MaxD= 10; Feat=3; Itera.=100	227	123	278
	31	119	
MaxD= 10; Feat=3; Itera.=1000	236	114	274
	32	118	
MaxD= 10; Feat=3; Itera.=2000	235	115	275
	32	118	
MaxD= 10; Feat=5; Itera.=100	240	110	270
	32	118	
MaxD= 10; Feat=5; Itera.=1000	242	108	253
	29	121	
MaxD= 10; Feat=5; Itera.=2000	243	107	**247**
	28	122	
MaxD= 10; Feat=10; Itera.=100	241	109	259
	30	120	
MaxD= 10; Feat=10; Itera.=1000	243	107	262
	31	119	
MaxD= 10; Feat=10; Itera.=2000	247	103	258
	31	119	
MaxD= 20; Feat=3; Itera.=100	230	120	270
	30	120	
MaxD= 20; Feat=3; Itera.=1000	233	117	277
	32	118	
MaxD= 20; Feat=3; Itera.=2000	235	115	280
	33	117	
MaxD= 20; Feat=5; Itera.=100	240	110	285
	35	115	
MaxD= 20; Feat=5; Itera.=1000	243	107	**252**
	29	121	
MaxD= 20; Feat=5; Itera.=2000	242	108	248
	28	122	
MaxD= 20; Feat=10; Itera.=100	241	109	259
	30	120	
MaxD= 20; Feat=10; Itera.=1000	243	107	262
	31	119	
MaxD= 20; Feat=10; Itera.=2000	247	103	258
	31	119	

Fonte: Elaboração e Cálculos Próprios. Nota: MaxD. (*Max Depht*); Feat. (*Nº Features*); Itera. (*Nº Iterations*).

Tabela 16. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método Rede Neuronal, na 1º Subamostra da Alemanha.

Rede Neuronal	Matriz Confusão		Função Custo
Train=200; Hid= 15 ; Lea.Rat.=0.01	217	133	**283**
	30	120	
Train=200; Hid= 15,15 ; Lea.Rat.=0.01	85	265	325
	12	138	
Train=200; Hid= 31 ; Lea.Rat.=0.01	217	133	**288**
	31	119	
Train=200; Hid= 31,31 ; Lea.Rat.=0.01	154	196	**301**
	21	129	
Train=200; Hid= 15 ; Lea.Rat.=0.3	292	58	433
	75	75	
Train=200; Hid= 15,15 ; Lea.Rat.=0.3	288	62	397
	67	83	
Train=200; Hid= 31 ; Lea.Rat.=0.3	279	71	371
	60	90	
Train=200; Hid= 31,31 ; Lea.Rat.=0.3	274	76	401
	65	85	
Train=200; Hid= 15 ; Lea.Rat.=0.5	267	83	403
	64	86	
Train=200; Hid= 15,15 ; Lea.Rat.=0.5	266	84	424
	68	82	
Train=200; Hid= 31 ; Lea.Rat.=0.5	283	67	377
	62	88	
Train=200; Hid= 31,31 ; Lea.Rat.=0.5	259	91	421
	66	84	
Train=500; Hid= 15 ; Lea.Rat.=0.01	246	104	354
	50	100	
Train=500; Hid= 15,15 ; Lea.Rat.=0.01	241	109	324
	43	107	
Train=500; Hid= 31 ; Lea.Rat.=0.01	244	106	341
	47	103	
Train=500; Hid= 31,31 ; Lea.Rat.=0.01	222	128	303
	35	115	
Train=500; Hid= 15 ; Lea.Rat.=0.3	273	77	422
	69	81	
Train=500; Hid= 15,15 ; Lea.Rat.=0.3	292	58	398
	68	82	
Train=500; Hid= 31 ; Lea.Rat.=0.3	278	72	367
	59	91	
Train=500; Hid= 31,31 ; Lea.Rat.=0.3	281	69	399
	66	84	

Rede Neuronal	Matriz Confusão		Função Custo
Train=500; Hid= 15 ; Lea.Rat.=0.5	269	81	396
	63	87	
Train=500; Hid= 15,15 ; Lea.Rat.=0.5	275	75	465
	78	72	
Train=500; Hid= 31 ; Lea.Rat.=0.5	280	70	380
	62	88	
Train=500; Hid= 31,31 ; Lea.Rat.=0.5	270	80	370
	58	92	
Train=1000; Hid= 15 ; Lea.Rat.=0.01	269	81	361
	56	94	
Train=1000; Hid= 15,15 ; Lea.Rat.=0.01	255	95	375
	56	94	
Train=1000; Hid= 31 ; Lea.Rat.=0.01	266	84	379
	59	91	
Train=1000; Hid= 31,31 ; Lea.Rat.=0.01	260	90	375
	57	93	
Train=1000; Hid= 15 ; Lea.Rat.=0.3	281	69	394
	65	85	
Train=1000; Hid= 15,15 ; Lea.Rat.=0.3	292	58	403
	69	81	
Train=1000; Hid= 31 ; Lea.Rat.=0.3	278	72	362
	58	92	
Train=1000; Hid= 31,31 ; Lea.Rat.=0.3	281	69	394
	65	85	
Train=1000; Hid= 15 ; Lea.Rat.=0.5	274	76	396
	64	86	
Train=1000; Hid= 15,15 ; Lea.Rat.=0.5	280	70	435
	73	77	
Train=1000; Hid= 31 ; Lea.Rat.=0.5	279	71	391
	64	86	
Train=1000; Hid= 31,31 ; Lea.Rat.=0.5	283	67	417
	70	80	

Fonte: Elaboração e Cálculos Próprios. Nota: Train. (*Training Time*); Hid. (*Hidden Layers*) e Lea.Rat.(*Learning Rate*).

Tabela 17. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método Rede Neuronal, na 1º Subamostra de Taiwan.

Support Vector Machine	**Matriz Confusão**		**Função Custo**
C= 0,01; K=LI; G=1	2 247	1 071	**11 314**
	5 959	5 723	
C= 0,01; K=Rbf; G=1	3 318	0	**11 682**
	11 682	0	
C= 0,01; K=Sig; G=1	3 318	0	**11 682**
	11 682	0	
C= 1; K=LI; G=1	1 991	1 327	12 031
	5 396	6 286	
C= 1; K=Rbf; G=1	14	3 304	16 534
	14	11 668	
C= 1; K=Sig; G=1	3 314	4	11 691
	11 671	11	
C= 1000; K=LI; G=1	2 307	1 011	11 265
	6 210	5 472	
C= 1000; K=Rbf; G=1	15	3 303	16 531
	16	11 666	
C= 1000; K=Sig; G=1	3 314	4	11 691
	11 671	11	

Fonte: Elaboração e Cálculos Próprios. Nota: C(*Cost*); K (*Kernel Type*); G(*Gamma*); LI(*Linear*); Rbf (*Radial Basis Function*); Sig (*Sigmoid*).

Tabela 18. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método Árvore de Classificação, na 1º Subamostra de Taiwan.

Árvore de Classificação (J4.8)	**Matriz Confusão**		**Função Custo**
MinO.=2 ; Conf.F=0,1	2 407	911	**7 731**
	3 176	8 506	
MinO.=2 ; Conf.F=0,25	2 188	1 130	8 307
	2 657	9 025	
MinO.=2 ; Conf.F=0,5	2 031	1 287	8 905
	2 470	9 212	
MinO.=5 ; Conf.F=0,1	2 475	843	**7 655**
	3 440	8 242	
MinO.=5 ; Conf.F=0,25	2 368	950	7 918
	3 168	8 514	
MinO.=5 ; Conf.F=0,5	2 259	1 059	8 219
	2 924	8 758	
MinO.=8 ; Conf.F=0,1	2 499	819	**7 585**
	3 490	8 192	
MinO.=8 ; Conf.F=0,25	2 447	871	7 795
	3 440	8 242	
MinO.=8 ; Conf.F=0,5	2 345	973	8 198
	3 333	8 349	

Fonte: Elaboração e Cálculos Próprios. Nota: Min.N.Inst. (*Minimum Number of Instances*); Conf.F.(*Confidence Factor*).

Tabela 19. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método *Random Forest*, na 1º Subamostra de Taiwan.

Random Forest	Matriz Confusão		Função Custo
MaxD= 3; Feat=3; Itera.=100	2 663	655	8 355
	5 080	6 602	
MaxD= 3; Feat=3; Itera.=1000	2 680	638	8 224
	5 034	6 648	
MaxD= 3; Feat=3; Itera.=2000	2 666	652	8 236
	4 976	6 706	
MaxD= 3; Feat=5; Itera.=100	2 389	929	8 190
	3 545	8 137	
MaxD= 3; Feat=5; Itera.=1000	2 410	908	7 992
	3 452	8 230	
MaxD= 3; Feat=5; Itera.=2000	2 404	914	8 087
	3 517	8 165	
MaxD= 3; Feat=10; Itera.=100	2 118	1 200	8 371
	2 371	9 311	
MaxD= 3; Feat=10; Itera.=1000	2 113	1 205	8 326
	2 301	9 381	
MaxD= 3; Feat=10; Itera.=2000	2 106	1 212	8 341
	2 281	9 401	
MaxD= 10; Feat=3; Itera.=100	2 457	861	7 014
	2 709	8 973	
MaxD= 10; Feat=3; Itera.=1000	2 442	876	7 079
	2 699	8 983	
MaxD= 10; Feat=3; Itera.=2000	2 442	876	7 081
	2 701	8 981	
MaxD= 10; Feat=5; Itera.=100	2 433	885	7 029
	2 604	9 078	
MaxD= 10; Feat=5; Itera.=1000	2 426	892	7 003
	2 543	9 139	
MaxD= 10; Feat=5; Itera.=2000	2 428	890	6 994
	2 544	9 138	

Random Forest	Matriz Confusão		Função Custo
MaxD= 10; Feat=10; Itera.=100	2 433	885	**6 927**
	2 502	9 180	
MaxD= 10; Feat=10; Itera.=1000	2 428	890	**6 921**
	2 471	9 211	
MaxD= 10; Feat=10; Itera.=2000	2 422	896	**6 940**
	2 460	9 222	
MaxD= 20; Feat=3; Itera.=100	2 160	1 158	7 435
	1 645	10 037	
MaxD= 20; Feat=3; Itera.=1000	2 161	1 157	7 409
	1 624	10 058	
MaxD= 20; Feat=3; Itera.=2000	2 166	1 152	7 385
	1 625	10 057	
MaxD= 20; Feat=5; Itera.=100	2 167	1 151	7 387
	1 632	10 050	
MaxD= 20; Feat=5; Itera.=1000	2 164	1 154	7 360
	1 590	10 092	
MaxD= 20; Feat=5; Itera.=2000			
MaxD= 20; Feat=10; Itera.=100	2171	1147	7353
	1618	10064	
MaxD= 20; Feat=10; Itera.=1000	2161	1157	7373
	1588	10094	
MaxD= 20; Feat=10; Itera.=2000	2160	1158	7380
	1590	10092	

Fonte: Elaboração e Cálculos Próprios. Nota: MaxD. (*Max Depht*); Feat. (*Nº Features*); Itera. (*Nº Iterations*). Nota: Uma das combinações de Meta-Parâmetros apresentadas não se obteve resultados, devido a incapacidade de processamento dos computadores utilizados neste trabalho.

Tabela 20. Resultados obtidos de todas as combinações de Meta-Parâmetros do Método Rede Neuronal, na 1º Subamostra de Taiwan.

Rede Neuronal	Matriz Confusão		Função Custo
Train=200; Hid= 15 ; Lea.Rat.=0.01	2 438	880	8 176
	3 776	7 906	
Train=200; Hid= 15,15 ; Lea.Rat.=0.01	2 395	923	8 409
	3 794	7 888	
Train=200; Hid= 31 ; Lea.Rat.=0.01	2 445	873	8 336
	3 971	7 711	
Train=200; Hid= 31,31 ; Lea.Rat.=0.01	2 400	918	8 403
	3 813	7 869	
Train=200; Hid= 15 ; Lea.Rat.=0.3	2 407	911	8 660
	4 105	7 577	
Train=200; Hid= 15,15 ; Lea.Rat.=0.3	2 282	1 036	8 767
	3 587	8 095	
Train=200; Hid= 31 ; Lea.Rat.=0.3	2 331	987	8 589
	3 654	8 028	
Train=200; Hid= 31,31 ; Lea.Rat.=0.3	2 124	1 194	9 015
	3 045	8 637	
Train=200; Hid= 15 ; Lea.Rat.=0.5	2 407	911	8 660
	4 105	7 577	
Train=200; Hid= 15,15 ; Lea.Rat.=0.5	2 282	1 036	8 767
	3 587	8 095	
Train=200; Hid= 31 ; Lea.Rat.=0.5	2 331	987	8 589
	3 654	8 028	
Train=200; Hid= 31,31 ; Lea.Rat.=0.5	2 001	1 317	9 071
	2 486	9 196	
Train=500; Hid= 15 ; Lea.Rat.=0.01	2 419	899	8 085
	3 590	8 092	
Train=500; Hid= 15,15 ; Lea.Rat.=0.01	2 437	881	8 053
	3 648	8 034	
Train=500; Hid= 31 ; Lea.Rat.=0.01	2 453	865	8 033
	3 708	7 974	
Train=500; Hid= 31,31 ; Lea.Rat.=0.01	2 478	840	8 025
	3 825	7 857	
Train=500; Hid= 15 ; Lea.Rat.=0.3	2 394	924	8 174
	3 554	8 128	
Train=500; Hid= 15,15 ; Lea.Rat.=0.3	2 422	896	8 539
	4 059	7 623	
Train=500; Hid= 31 ; Lea.Rat.=0.3	2 273	1 045	8 683
	3 458	8 224	
Train=500; Hid= 31,31 ; Lea.Rat.=0.3	2 216	1 102	8 946
	3 436	8 246	
Train=500; Hid= 15 ; Lea.Rat.=0.5	2 314	1 004	8 423
	3 403	8 279	
Train=500; Hid= 15,15 ; Lea.Rat.=0.5	2 374	944	8 564
	3 844	7 838	
Train=500; Hid= 31 ; Lea.Rat.=0.5	2 163	1 155	8 772
	2 997	8 685	
Train=500; Hid= 31,31 ; Lea.Rat.=0.5	2 160	1 158	8 676
	2 886	8 796	
Train=1000; Hid= 15 ; Lea.Rat.=0.01	2 451	867	**7 923**
	3 588	8 094	
Train=1000; Hid= 15,15 ; Lea.Rat.=0.01	2 493	825	**7 843**
	3 718	7 964	
Train=1000; Hid= 31 ; Lea.Rat.=0.01	2 480	838	7 978
	3 788	7 894	
Train=1000; Hid= 31,31 ; Lea.Rat.=0.01	2 554	764	**7 890**
	4 070	7 612	
Train=1000; Hid= 15 ; Lea.Rat.=0.3	2 457	861	7 943
	3 638	8 044	
Train=1000; Hid= 15,15 ; Lea.Rat.=0.3	2 394	924	8 541
	3 921	7 761	
Train=1000; Hid= 31 ; Lea.Rat.=0.3	2 260	1 058	8 756
	3 466	8 216	
Train=1000; Hid= 31,31 ; Lea.Rat.=0.3	2 271	1 047	9 033
	3 798	7 884	
Train=1000; Hid= 15 ; Lea.Rat.=0.5	2 353	965	8 555
	3 730	7 952	
Train=1000; Hid= 15,15 ; Lea.Rat.=0.5	2 353	965	8 555
	3 730	7 952	
Train=1000; Hid= 31 ; Lea.Rat.=0.5	2 188	1 130	8 585
	2 935	8 747	
Train=1000; Hid= 31,31 ; Lea.Rat.=0.5	2 214	1 104	8 781
	3 261	8 421	

Fonte: Elaboração e Cálculos Próprios. Nota: Train. (*Training Time*); Hid. (*Hidden Layers*) e Lea.Rat.(*Learning Rate*).

D. Procedimentos realizados com a finalidade da criação do *Ensemble*

Primeiro, num ficheiro Excel juntaram-se todas as previsões dos melhores meta-parâmetros utilizados nas segundas subamostras.

De seguida, tal como podemos verificar na Figura 11, utilizou se a Função SE para verificar para cada instância, quantos modelos previam classe 1 (1º coluna) ou quantos previam classe 2 (2º coluna). Por exemplo, na primeira linha, 7 modelos preveem classe 1 e 5 preveem classe 2, o que perfaz um total de 12 modelos.

Figura 11. *Ensemble* na 2º Subamostra da Alemanha.

	1	2	0		310					
	01:01	02:02								
Npred1	**Npred2**	**Controlo**	**Real**	**Previsão**	**Penalização**	**Auxiliar (Real&Previsão)**				
7	5	0	1	1	0	11				
9	3	0	1	1	0	11	**Matriz Custo**			
1	11	0	1	2	1	12			**1**	**2**	**F. Custo**
2	10	0	1	2	1	12	**1**	135	215	**310**
8	4	0	1	1	0	11	**2**	19	131	
1	11	0	1	2	1	12				
8	4	0	1	1	0	11				
4	8	0	1	2	1	12				
3	9	0	1	2	1	12				

Fonte: Elaboração e Cálculos Próprios. Nota: F.Custo (Função Custo).

A terceira coluna é apenas para o controlo total das previsões, é a soma das duas primeiras colunas menos 12 modelos. Na coluna a seguir, denominada "Real" verifica-se o resultado real (o output). Por exemplo, no caso da Alemanha, verifica-se o cliente na base de dados indica um bom risco de crédito, e nesse caso atribuir-se-á o valor de 1, caso contrário atribuir-se-á o valor 2.

A quinta coluna, "Previsão" analisa-se na coluna "Npred1" se obteve mais de metade dos modelos, ou seja, mais do que 6 alcançaram a previsão de classe 1. Caso tal ocorra, considera-se prevista a classe 1. Caso contrário, a previsão será de classe 2.

A coluna "Penalização" avalia-se a "Previsão" é igual a coluna "Real" então obtemos a classificação de 0, caso contrário se a previsão "Real" é igual a 1, atribui-se o peso 1. Caso não seja, atribui-se o peso 5.

A última coluna "Auxiliar" é apenas a junção dos valores reais e das previsões.

A matriz custo é idêntica às matrizes apresentadas pelo *software Weka*. Esta matriz obtém-se através da Função Contar.se. Para o valor no canto superior esquerdo da matriz, conta-se, na última coluna, o número de instâncias em que a previsão é 1 quando e o seu valor real é também 1, que são 135.

A função custo é, neste caso, o resultado de 5*19+215=310.

Como podemos observar, o mesmo procedimento se repete para a segunda subamostra de Taiwan (Figura 12).

Figura 12. *Ensemble* na 2º Subamostra de Taiwan.

	1	2	0		6151					
	01:01	02:00								
Npred1	**Npred2**	**Controlo**	**Real**	**Previsão**	**Penalização**	**Auxiliar(Real&Previsão)**				
1	11	0	2	2	0	22				
7	5	0	1	1	0	11	**Matriz Custo**			
5	7	0	1	2	5	12		**1**	**2**	**F. Custo**
1	11	0	2	2	0	22	**1**	4254	1079	**6151**
2	10	0	2	2	0	22	**2**	756	8911	
3	9	0	2	2	0	22				
1	11	0	2	2	0	22				
4	8	0	2	2	0	22				
2	10	0	2	2	0	22				

Fonte: Elaboração e Cálculos Próprios. Nota: F.Custo (Função Custo)