U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

APLICAÇÃO DE IA NA OTIMIZAÇÃO DE ESTRUTURAS METÁLICAS PRÉ-FABRICADAS EM COBERTURAS E FACHADAS

FRANCISCO ANTÓNIO MACHADO DA COSTA PONTES

Dissertação submetida para satisfação parcial dos requisitos do grau de

MESTRE EM ENGENHARIA CIVIL — ESPECIALIZAÇÃO EM ESTRUTURAS E GEOTECNIA

Orientador: Professor Dr. João Manuel Pires Macedo

JUNHO DE 2025

MESTRADO EM ENGENHARIA CIVIL 2024/2025

DEPARTAMENTO DE ENGENHARIA CIVIL

Tel. +351-22-508 1901

✉ m.ec@fe.up.pt

Editado por

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Rua Dr. Roberto Frias

4200-465 PORTO

Portugal

Tel. +351-22-508 1400

✉ feup@fe.up.pt

http://www.fe.up.pt

Reproduções parciais deste documento serão autorizadas, na condição de que seja mencionado o Autor e feita referência a *Mestrado em Engenharia Civil - 2024/2025 - Departamento de Engenharia Civil, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2025*.

As opiniões e informações incluídas neste documento representam unicamente o ponto de vista do respetivo Autor, não podendo o Editor aceitar qualquer responsabilidade, legal ou outra, em relação a erros ou omissões que possam existir.

Este documento foi produzido a partir de versão eletrónica fornecida pelo respetivo Autor.

À minha irmã Carolina, pelo exemplo vivo de que o compromisso com a academia resulta, por fim, na expressão máxima da Engenharia, que é o aperfeiçoamento da sociedade em que vivemos.

The greater our knowledge increases, the greater our ignorance unfolds

John F. Kennedy

AGRADECIMENTOS

Partilho aqui os meus agradecimentos, em especial à minha família, por todo o suporte afetivo, financeiro e psicológico que se prestaram fazer nos últimos 5 anos da minha vida para que concluísse com sucesso esta etapa que aqui finda.

À Universidade do Porto, e a todos os docentes do Departamento de Engenharia Civil da Faculdade de Engenharia pela disponibilidade que sempre demonstraram na partilha de conhecimentos e de experiências detidas pelos mesmos. Esta será a escola que com orgulho tratarei sempre como uma casa.

Ao Orientador da presente Dissertação, Professor Doutor João Manuel Pires Macedo, pela grandeza de tudo o que me ensinou. O meu mundo é hoje bem maior.

RESUMO

A Inteligência Artificial (IA) é, na atualidade, uma das tecnologias onde se desenvolvem mais projetos, sejam de proponente empresarial ou académica. A aplicação da mesma à Engenharia Estrutural tornou-se, por isso, num campo de investigação desafiante, com a IA a poder ser integrada no processo de análise, monitorização e dimensionamento estrutural.

Ainda que, na atualidade, a maior parte dos softwares para cálculo estrutural se apoiem na utilização do Método dos Elementos Finitos (FEM), a dificuldade na sua aplicabilidade, nomeadamente na análise numérica não linear de estruturas, bem como a necessidade de utilizar grandes recursos computacionais, oferecem à IA uma oportunidade para produzir novas técnicas, particularmente com a utilização de modelos de Machine Learning (ML) e Deep Learning (DL), bastante mais rápidos, mas igualmente precisos e corretos.

O objetivo da presente dissertação passa por aplicar a IA como passe de arranque na Otimização de Estruturas Metálicas Pré-Fabricadas em Coberturas e Fachadas, designadamente com a criação de um modelo de ML, recorrendo à linguagem de programação Python, que permita acelerar o processo do dimensionamento otimizado de elementos estruturais em aço. Com recurso a um DataBase (DB), perfazendo um total de 60 000 (sessenta mil) casos de estudo distintos, foi construída uma Rede Neuronal Artificial (RNAs**),** com um total de 3 (três) camadas, onde, posteriormente, recorrendo à utilização da aprendizagem supervisionada, a própria rede foi treinada, com o objetivo final de esta ser capaz de, perante um conjunto de dados distintos dos dados categorizados previamente fornecidos à rede, generalizar o conhecimento adquirido e conseguir prever, de forma precisa, as dimensões de elementos estruturais cumprindo o regulamento em vigor para o dimensionamento de estruturas metálicas, EN 1993-1-1:.. Projeto de Estruturas de Aço (EC-3).

Os dados que serviram como input da Rede Neuronal foram fracionados em 3 (três) características principais, sendo elas: a exigência de que o perfil utilizado para solução fosse do tipo Madre e que a secção de solução selecionada fosse de 100 mm de diâmetro.

Para conferir capacidade de cálculo não-linear ao modelo construído, foi utilizada a Função de Ativação Rectified Linear Unit (ReLU); para garantir a performance do modelo, foi utilizada a Função de Perda, Mean Squared Error (MSE); e, para assegurar a minimização da Função de Perda, foi utilizada a Função de Otimização Adaptive Moment Estimation (ADAM).

No final, foi discutida a precisão e fiabilidade do modelo construído, tendo por base as diferenças entre os valores obtidos no modelo e no simulador de cálculo, seguindo métricas de avaliação como o Erro Absoluto Médio (MAE) e o Erro Quadrático Médio (RMSE), e extraindo tabelas e gráficos comparativos entre os resultados "reais" obtidos pelo algoritmo e os resultados obtidos pela execução da generalização do modelo, deixando, por último, considerações finais e sugestão para trabalho futuro.

PALAVRAS-CHAVE: INTELIGÊNCIA ARTIFICIAL, DIMENSIONAMENTO DE ESTRUTURAS METÁLICAS PRÉ-FABRICADAS, MACHINE LEARNING, PYTHON, REDE NEURONAL ARTIFICIAL.

ABSTRACT

Artificial Intelligence (AI) is currently one of the technologies with the most developed projects, whether from corporate or academic proponents. Its application to Structural Engineering has therefore become an interesting field of research, with AI potentially being integrated into the processes of structural analysis, monitoring, and design.

Although most structural analysis software currently relies on the Finite Element Method (FEM), the challenges in its applicability, particularly in the nonlinear numerical analysis of structures, as well as the need for significant computational resources, present an opportunity for AI to develop new techniques. Specifically, the use of Machine Learning (ML) and Deep Learning (DL) models offers faster, yet equally precise and accurate solutions.

The objective of this dissertation is to apply AI as the initial step in he Optimization of Prefabricated Metallic Structures in Roofs and Facades, particularly through the creation of an ML model, using the programming language Python, that enables the optimized design of steel structural elements. Using a Database (DB), comprising a total of 60 000 (sixty thousand) distinct case studies, an Artificial Neural Network (ANN) with a total of 3 (three) layers was constructed. Subsequently, through supervised learning, the network was trained with the goal of being able to generalize the acquired knowledge and accurately predict the dimensions of structural elements when presented with a new set of data, distinct from the categorized data previously provided to the network. The predictions comply with the current regulations for the design of metallic structures, EN 1993-1-1: Design of Steel Structures (EC-3).

The data used as input for the neural network was divided into two main characteristics: the requirement that the profile used for the solution be of the Madre type and that the selected solution section be 100 mm in diameter.

To confer nonlinear calculation capability to the model, the Rectified Linear Unit (ReLU) activation function was used. To ensure model performance, the Mean Squared Error (MSE) loss function was employed, and to minimize the loss function, the Adaptive Moment Estimation, (ADAM) optimization function was applied.

In the end, the accuracy and reliability of the constructed model were discussed based on the differences between the values obtained from the model and those from the simulator, following rigorous assessment metrics such as the the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE), and extracting comparative tables and graphs between the 'real' results obtained by the algorithm and the results obtained by executing the model generalization, leaving final considerations and suggestions for future work for last..

KEYWORDS: ARTIFICIAL INTELLIGENCE, DESIGN OF PREFABRICATED METALLIC STRUCTURES, MACHINE LEARNING, PYTHON, ARTIFICIAL NEURAL NETWORK.

ÍNDICE GERAL

AGRADECIMENTOS I
RESUMO III
ABSTRACT V
ÍNDICE DE FIGURAS X
SÍMBOLOS, ACRÓNIMOS E ABREVIATURAS XIII
DEFINIÇÕES XVI

1. INTRODUÇÃO 1
1.1. CONSIDERAÇÕES INICIAIS 1
1.2. OBJETIVOS 2
1.3. ESTRUTURA DA DISSERTAÇÃO 3

2. ENQUADRAMENTO TEÓRICO 5
2.1. INTELIGÊNCIA ARTIFICIAL 5
2.1.1. REACTIVE MACHINES 6
2.1.1.1. Características 7
2.1.2. LIMITED MEMORY 7
2.1.2.1. Características 7
2.1.3. THEORY OF MIND 8
2.1.3.1. Características 8
2.1.4. SELF AWARE AI 9
2.1.4.1. Características 9
2.2. MACHINE LEARNING 10
2.2.1. A APRENDIZAGEM EM MODELOS DE MACHINE LEARNING 10
2.2.1.1. Relevância dos Dados no contexto do ML com a definição da Terminologia utilizada 12
2.2.1.2. Dados Classificados e Não- Classificados 12
2.2.1.3. O que é a Tarefa em ML 13
2.2.1.4. Algoritmos de ML 13
2.2.1.5. Funções de Ativação e Otimização 32
2.2.1.6. Métricas de Avaliação em algoritmos de ML 40

3. DESENVOLVIMENTO DO MODELO DE MACHINE LEARNING 45

3.1. OBJETIVO DO MODELO DE MACHINE LEARNIG 45

3.1.1. APRESENTAÇÃO E EXPLICITAÇÃO DAS CARACTERÍSTICAS DO PERFIL TIPO MADRE 46

3.1.2. APRESENTAÇÃO E EXPLICITAÇÃO DAS CARACTERÍSTICAS DO PERFIL TIPO OMEGA 49

3.1.3. EXPLICAÇÃO DO MÉTODO DE OPERACIONALIDADE DO SIMULADOR DE CÁLCULO ESTRUTURAL 50

3.1.3.1. Variáveis de Input necessárias para o desenvolvimento do cálculo dentro do Simulador 51

3.1.3.2. Variáveis de output que constituem a solução do perfil calculado pelo Simulador 59

3.2. EXPLICITAÇÃO DA CONSTRUÇÃO DO MODELO DE MACHINE LEARNING 65

3.2.1. EXPLICITAÇÃO DA OPERACIONALIDADE DO PROGRAMA P1 67

3.2.1.1. Construção e Import da Base de Dados 67

3.2.1.2. Estruturação dos Dados 67

3.2.1.3. Normalização e Seleção dos dados de treino e validação 69

3.2.1.4. Arquitetura da Rede Neuronal 70

3.2.1.5. Treino da Rede Neuronal 72

3.2.1.6. Armazenamento da Matriz de Pesos 73

3.2.2. EXPLICITAÇÃO DA OPERACIONALIDADE DO PROGRAMA P2 74

3.2.2.1. Construção e Import da Base de Dados 74

3.2.2.2. Definição da Arquitetura do modelo 75

3.2.2.3. Import da Matriz de Pesos e Previsão de Soluções 75

3.2.2.4. Pós Processamento dos Dados e Outputs de Resultados 75

4. ANÁLISE DOS RESULTADOS 76

4.1. A ANÁLISE DE RESULTADOS EM ML 76

4.1.1. ANÁLISE DOS VALORES OBTIDOS PARA AS MÉTRICAS MSE E MAE NO P1 76

4.1.2. ANÁLISE COMPARATIVA ENTRE AS SOLUÇÕES CALCULADAS PELO SIMULADOR DE CÁLCULO E AS SOLUÇÕES PREVISTAS PELO MODELO DE ML 77

4.1.2.1. Comparação de Soluções Previstas e Reais 77

4.1.2.2. Distribuição do Erro percentual por Intervalos percentuais do Erro 78

4.1.2.3. Análise do Erro percentual em função do Comprimento do Vão 79

4.1.2.4. Precisão das Estimativas: Alinhamento entre Medidas Reais e Previstas 80

4.1.2.5. Distribuição Espacial dos Erros Percentuais 81

4.1.2.6. Avaliação da Capacidade do modelo na Previsão dos valores das restantes variáveis-alvo 82

5. CONCLUSÃO E TRABALHO FUTURO 84

REFERÊNCIAS BIBLIOGRÁFICAS 86

ÍNDICE DE FIGURAS

Figura 1- História do Desenvolvimento da AI 5

Figura 2- Diferentes tipos de AI baseados na Funcionalidade 10

Figura 3 - Mapa de Conceitos em Inglês onde estão referidas as semânticas do ML.....10

Figura 4 - Aplicação da Aprendizagem em contexto de ML, para a previsão de comportamentos.....12

Figura 5 - Diagrama representativo das diferenças entre Dados Classificados e Não-Classificados ..13

Figura 6 - Modus Operandi de um modelo de ML para dar resposta a um problema de Classificação 15

Figura 7 - Exemplos de 2 tipos de Regressão. Regressão linear à esquerda e Regressão múltipla à direita. 15

Figura 8 - Algoritmos de ML agrupados por tipo de aprendizagem e problema 16

Figura 9 - Exemplo de uma solução de um modelo de Regressão Linear 17

Figura 10 - Regressão Multivariada.....18

Figura 11 - Representação Gráfica de uma Rede Neuronal.....19

Figura 12 - Esquema Representativo de Random Forest 20

Figura 13 - Representação Gráfica de Random Forests 23

Figura 14 - Representação Gráfica do Método Operacional de uma Decision Tree 24

Figura 15 - Exemplo Explicativo da Utilização do Erro de Classificação na Construção de uma Decision Tree 25

Figura 16 – Exemplo gráfico do funcionamento teórico de um Boosting Alghoritm.....26

Figura 17 - Representação Gráfica do processo exposto em cima para criação de um algoritmo de Boosting para dar resposta a um problema de classificação.....27

Figura 18 - Representação Gráfica de um SVM em que procura solucionar um problema simples de classificação binária 29

Figura 19 - Ilustração de um algoritmo kNN uniforme para regressão para (a) k=2 e (b) k=5 32

Figura 20 - Tabela sumário das Funções de Ativação 35

Figura 21 - Exemplo gráfico de uma Confusion Matrix.....41

Figura 22 – Esquema estrutural da cobertura ou fachada, composto por 8 vãos e 2 consolas, conforme definido pelas restrições aplicadas aos dados 66

Figura 23 - Distribuição dos Erros Percentuais dos Valores Previstos 78

Figura 24 - Gráfico de Dispersão Erro Percentual em função do Comprimento dos Segmentos.....79

Figura 25 - Gráfico Scatter Plot de comparação entre os valores Reais e Previstos.....80

Figura 26 - Mapa de Calor: Erro Percentual por Posição na Estrutura 81

SÍMBOLOS, ACRÓNIMOS E ABREVIATURAS

AI – Inteligência Artificial (Artificial Inteligence)

FEM – Método dos Elementos Finitos (Finite Element Method)

ML - Aprendizagem Automática (Machine Learning)

DL – Aprendizagem Profunda (Deep Learning)

DB – Base de Dados (DataBase)

RNAs – Redes Neuronais Artificiais

EC-3 - EN 1993-1-1: Projeto de Estruturas de Aço

ReLU - Rectified Linear Unit

ADAM - Adaptive Moment Estimation

MSE – Erro quadrático médio (Mean Squared Error)

χ – Fator de Redução da Resistência à Encurvadura por Flexão

χLT – Fator de Redução da Resistência à Encurvadura Lateral

MAE - Erro Absoluto Médio (Mean Absolute Error)

RMSE – Erro Quadrático Médio (Root Mean Square Error)

NLP – Processamento de Linguagem Natural (Natural Language Processing)

CV – Visão Computacional (Computer vision)

MDPs – Processos de Decisão de Markov (Markov Decicion Processes)

OOB – Erro fora de bolsa (Out – Of – Bag)

mm – Milímetros

SVM – Support Vector Machine

kNN – k-nearest Neighbors

tanh – Hyperbolic Tangent

PReLU – Parametric Rectified Linear Unit

ELU - Exponencial Linear Unit

SELU - Scaled Exponential Linear Unit

GM - Geometric-Mean

SGD – Stochastic Gradient Descent

AdaGrad – Adaptive Gradient Algorithm

ADAM - Adaptive Moment Estimation

m – Metros

m/s – metros por segundo

mrad – milirradianos

kg/m – quilogramas por metro

kg- quilogramas

MPa – Mega Pascal

DEFINIÇÕES

Backpropagation – Backpropagation é um algoritmo utilizado no treino de RNAs, primariamente para gradientes computacionais de funções representadas por gráficos computacionais. Utiliza a regra matemática de cadeia para ajustar de forma eficiente os parâmetros de um modelo, entre os quais o Peso e as Bias, durante o decorrer de um processo de treino. Este método é crucial no ML, em particular, por otimizar redes neuronais [17].

Hiper-parâmetros – Hiper- parâmetros são o conjunto de parâmetros de um modelo de ML que são fixos e, por isso, não ajustáveis durante o processo de treino [21].

Teste t – O teste t é um teste estatístico bivariado tradicionalmente empregue em situações onde o objetivo passa por aferir as médias de 2 (dois) grupos, para estabelecer se são substancialmente distintos um do outro. Na maioria das vezes, é aplicado para estudar dados contínuos [22].

Teste $\chi 2$ – O teste $\chi 2$ (qui quadrado) é tradicionalmente empregue em situações em que o objetivo passa por avaliar a relação entre 2 (duas) variáveis expostas, através de uma tabela de contingência. A invariância do teste $\chi 2$ (qui quadrado) permite que a forma do teste permaneça inalterada sobre diferentes modelos probabilísticos [23].

Método de Bagging – O método de Bagging consiste na criação de versões distintas de um preditor, a partir de um conjunto de amostras bootsrap do conjunto de treino original, com o objetivo de alcançar um modelo preditor mais agregador. A média das agregações é utilizada para prever dados de saída, que estão representados sob a forma numérica. Por sua vez, na previsão de uma classe, o método executa uma votação maioritária. A eficácia do método de Bagging é respeitante à instabilidade do conjunto de treino original; em casos onde se verifiquem pequenas variações no conjunto de treino que possam resultar em previsões distintas, o uso do Baggin reduz a variância e melhora a precisão. O método de Bagging é, por isso, extremamente útil para algoritmos instáveis, em que pequenas perturbações nos dados de treino afetam consideravelmente o resultado de saída. [28]

1
INTRODUÇÃO

1.1. CONSIDERAÇÕES INICIAIS

Num mundo global cada vez mais desenvolvido, a simplificação de processos complexos, mantendo a precisão e a exatidão da decisão, torna-se cada vez mais uma necessidade competitiva nos mercados nacionais e internacionais. A crescente evolução tecnológica está conectada com a utilização da AI nas diferentes áreas e setores profissionais, oferecendo oportunidades de diferenciação competitiva muito importantes.

Neste sentido, a integração de AI, DL e ML em processos de análise, dimensionamento, manutenção, orçamentação e construção de Estruturas de Engenharia Civil tornou-se uma mais valia para os Engenheiros Civis [1] não só no ganho temporal que advém do rápido processo de cálculo das RNAs, muitas vezes milhares de vezes mais rápido do que os softwares tradicionais e os processos de cálculo analítico [2], mas também, pela possibilidade da utilização de Aprendizagem Automática, um subcampo de AI que possibilita que os sistemas interiorizem padrões a partir de uma base de dados fornecida, e que a partir daí, consigam efetuar previsões e decisões para casos de estudo desconhecidos do próprio modelo [3].

Isto torna-se particularmente útil para o Dimensionamento de Estruturas Metálicas Pré-Fabricadas. Apesar de maioritariamente utilizados, os softwares de cálculo que operam com as bases matemáticas do FEM apresentam uma série de limitações e dificuldades que podem ser diminuídas ou até ultrapassadas com a utilização de RNAs, sejam em modelos de ML independentes, incorporados num software de análise estrutural que utilize o FEM, ou incorporados em algoritmo de cálculo estrutural.

As dificuldades associadas à utilização de softwares que recorrem ao Método dos Elementos Finitos (FEM) verificam-se, essencialmente, no tratamento de incertezas nos parâmetros de entrada, nas condições de carregamento, nas alterações dos materiais e na definição do contorno [3], fatores que comprometem a exatidão dos resultados obtidos, especialmente em cenários de análise não linear. Especificamente na análise de estruturas metálicas pré-fabricadas, a principal dificuldade reside na escolha da curva de encurvadura, sendo o fator global de redução considerado o valor mínimo entre χ (encurvadura por flexão) e χ_{LT} (encurvadura lateral). Esta abordagem conduz a um valor conservador da resistência, que, por sua vez, não permite a utilização plena da capacidade resistente real do material, encarecendo o custo financeiro do projeto. Importa ainda salientar a adversidade associada à consideração de imperfeições, que dificulta a modelação das estruturas, bem como os desafios na definição da

classe da secção, na seleção das curvas de encurvadura e na identificação das secções críticas para o dimensionamento [4].

1.2. OBJETIVOS

Com base no exposto acima, a presente dissertação tem como objetivo criar um modelo de ML que permita otimizar o processo de Dimensionamento de Estruturas Metálicas Pré-Fabricadas em Coberturas e Fachadas.

O processo iniciou-se pela implementação de um algoritmo que permite, perante o input de dados de entrada, o cálculo do perfil metálico a ser aplicado nas coberturas e/ou fachadas do projeto pretendido, determinando não só as dimensões dos elementos constituintes das coberturas e das fachadas, como também as zonas de corte para produção, em fábrica, dos mesmos, garantindo, ao mesmo tempo, o cumprimento do regulamento exposto no EC-3.

No entanto, considerando o momento inicial de definição geométrica da estrutura, os cálculos iterativos do algoritmo atuam de forma repetitiva na procura das soluções que, além de cumprirem toda a regulamentação requerida, estejam otimizadas. Como o algoritmo consegue oferecer mais do que uma solução para cada cenário definido pelas variáveis de entrada, o processo de obtenção dessas soluções é um processo demorado e exigente do ponto de vista computacional.

É, por isso, recomendável acelerar este processo de cálculo, com a intenção de diminuir o tempo necessário para a obtenção dos elementos estruturais, o que, por consequência, diminuirá o esforço computacional exigido, resultando em ganhos temporais e financeiros substanciais.

É com este pressuposto que a criação de um modelo preditivo em ML, programado em Python, se torna relevante, porquanto permite maior velocidade nos processos de cálculo, sem que, com isso, surjam erros significativos. Para a criação do modelo, vão ser carregados os dados dos 60 000 (sessenta mil) casos de estudo, fracionados em 2 (duas) condições de controlo, sendo elas: a exigência de que o perfil selecionado seja do tipo Madre e que a secção de solução seja a de 100 (cem) mm. Os dados carregados e estratificados irão englobar não só os inputs iniciais, como as soluções (s0) otimizadas que o algoritmo calculou.

Através de treino, seguindo os princípios da Aprendizagem Supervisionada, perante o conjunto de dados de entrada a que estará associado um conjunto de dados de saída, será possível ao modelo ajustar os seus parâmetros, tendo por base o erro entre as previsões, o que resultará na capacitação do mesmo para prever soluções, neste caso específico associadas a um modelo de regressão. [3]

O objetivo da presenta Dissertação passa pela criação deste modelo, que permitirá acelerar o processo de obtenção do dimensionamento da estrutura (s0) e que entrará no simulador para obtenção da estrutura otimizada. O processo tornar-se-á assim mais rápido, requerendo menor utilização de esforço computacional, porque o modelo de ML já treinado será capaz de identificar uma lei de regressão entre os parâmetros de entrada e de saída, que tornará possível a partida diretamente da definição dos parâmetros de entrada para a solução (s0) de forma mais rápida e sem que isto origine erros significativos. Será discutida no capítulo 5º (quinto), Conclusão e Trabalho Futuro a expansão que permitirá a implementação deste modelo no simulador como auxílio ao algoritmo atual com ganhos temporais e financeiros significativos, derivados à potencialidade e velocidade dos processos de cálculo em modelos de AI e à economização da utilização do CPU respetivamente.

1.3. ESTRUTURA DA DISSERTAÇÃO

A presente Dissertação encontra-se dividida em 5 (cinco) capítulos, nos quais serão explicitadas e referenciadas todas as considerações científicas tomadas como verdadeiras:

O 1º (Primeiro) Capítulo é um capítulo introdutório, onde se encontram as Considerações Inicias, os Objetivos e a Estrutura da Dissertação. Apresenta, de forma resumida, a pertinência do tema que será tratado, os Objetivos que se pretendem atingir e a Estrutura da Dissertação, nomeadamente os conteúdos de cada capítulo e o que neles será exposto.

No 2º (Segundo) Capítulo é apresentado o Enquadramento Teórico, onde será exposta a base teórica e conceptual que suporta o trabalho. São descritas definições e enquadramentos científicos, sustentados por literatura assente em evidências e metodologia rigorosa.

O 3º (Terceiro) capítulo, Desenvolvimento do Modelo de Machine Learning, é onde será apresentado o procedimento para o desenvolvimento do modelo de ML, sendo explicadas todas as componentes de decisão relevantes e necessárias.

No 4º (Quarto) capítulo, Análise dos Resultados, serão apresentados e discutidos os resultados obtidos, que sustentarão a avaliação de precisão do modelo. A avaliação seguirá métricas de análise comparativa, como o erro percentual absoluto médio (MAE) e o Erro Quadrático Médio (RMSE), e serão expostas tabelas e gráficos informativos, onde são comparados os resultados obtidos pelo modelo de ML e os valores obtidos pelo desenvolvimento numérico do algoritmo.

No 5º (Quinto) capítulo, Conclusão e Trabalho Futuro, é exposta a conclusão da Dissertação e sugestões para futuras pesquisas sobre o tema abordado.

2

ENQUADRAMENTO TEÓRICO

2.1. INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (AI) começou a ser desenvolvida na década de 1950 (mil novecentos e cinquenta); no entanto, só se tornaria uma área comercial em 1980 (mil novecentos e oitenta). [6] As primeiras descobertas no campo da AI são, a título de exemplo, a criação da primeira rede Neuronal e a Aprendizagem por Reforço (RL), como exposto na Figura 1.

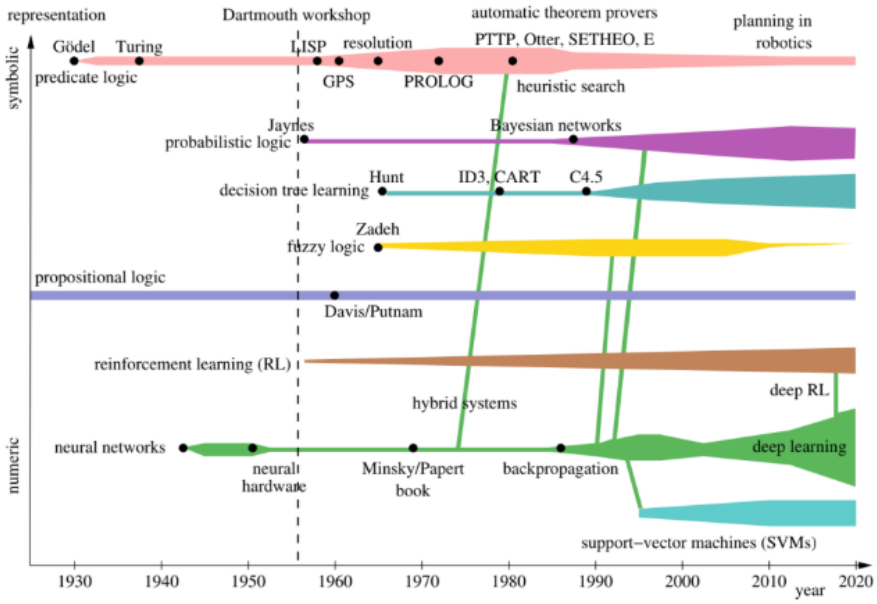


Figura 1- História do Desenvolvimento da AI

Apesar de, legalmente, a definição exata de AI ser algo ambígua [5], é possível defini-la, no contexto da presente dissertação, como um conceito que possibilita a criação de sistemas através de algoritmos que atuam na previsão de relações ou modelos. [1]

Fundamentalmente, o raciocínio da AI baseia-se no pensamento lógico de dedução, que reflete os processos cognitivos utilizados pelos seres-humanos. Da mesma forma que os indivíduos analisam e ava-

liam a informação que detém para formar decisões informadas, os sistemas de AI analisam os dados de input para, posteriormente, conseguirem tirar partido das informações dos conhecimentos adquiridos e, através de treino, obterem conclusões racionais. Este processo permite criar soluções ou estratégias viáveis para cumprir um ou mais objetivos específicos. A essência da AI é deter capacidade de raciocínio e de resolução de problemas, que irá funcionar como um mecanismo dinâmico na mecânica central da AI, capacitando os sistemas da lógica necessária para enfrentar os problemas a que os mesmos serão expostos, assegurando, ao mesmo tempo, eficácia e exatidão científica em problemas tão diversos como a automatização de veículos, robótica ou o diagnóstico de perturbações ou disfunções nos organismos humanos. [7]

No campo da Engenharia Civil, a AI aparece como uma oportunidade, pois pode aprimorar os processos de decisões na otimização, análise e processo de dimensionamento estrutural. A automatização de processos, como a capacidade de produzir modelos preditivos, permite que as aplicações de AI sejam uma importante ferramenta na melhoria da eficácia, precisão e criatividade das obras de Engenharia Civil. Ao permitir o desenvolvimento de algoritmos potentes, é possível, através de técnicas de ML ou DL ,produzir modelos que permitem maior interdisciplinaridade no seio da Engenharia Civil, ao mesmo tempo que produz vantagens significativas para tarefas como a identificação de fendas em pavimentos e de danos estruturais em estruturas de betão – armado, aperfeiçoando a manutenção das estruturas e garantindo maior segurança. [8]

A AI pode ser categorizada em 3 (três) tipos, baseados nas suas capacidades: Narrow AI ou Week AI, que é quando um sistema é construído e treinado com intervenção humana para a resolução de um exercício em particular; General AI ou Strong AI, que é quando sistemas possuem capacidades cognitivas humanas generalizadas que permitem encontrar soluções sem intervenção humana [9]; e Super AI, que é quando um sistema consegue superar a própria capacidade de raciocínio e inteligência humana, permitindo-lhe alcançar um nível tal de capacidade cognitiva que irá permitir ao sistema não só realizar tarefas altamente complexas, como demonstrar uma compreensão excecional do mundo [10].

Baseado na funcionalidade, a AI pode ser categorizada em 4 (quatro) tipos, Reactive Machines, Limited Memory, Theory of Mind e Self-Aware AI [11].

2.1.1. Reactive Machines

Reactive Machines têm como objetivo criar sistemas de AI que podem atuar na resolução de tarefas específicas e funcionar assentes em regras pré-definidas e respostas programadas. Estes sistemas não possuem capacidade para aprender ou adaptar-se a situações diferentes, dado que o comportamento do sistema por inteiro é determinado pelas instruções e regras previamente fornecidas aos mesmos. [11]

Reactive Machines estão limitadas a resolver tarefas ou problemas particulares dentro das fronteiras para que foram instruídas, através do recurso à programação. Esta AI não detém capacidade de memória ou dados para se desenvolver, nem permite aprendizagem com ações passadas. Apesar de serem úteis para problemas bem estruturados e tarefas repetitivas, estas restrições tornam as Reactive Machines inadequadas para situações dinâmicas, incertas ou envolvendo ambientes onde a aprendizagem e a adaptabilidade são essenciais. [11]

2.1.1.1. Características

Reactive Machines seguem uma série de regras predefinidas ou instruções meticulosamente criadas com recurso à programação. Estas regras são o guia base sobre o qual a máquina irá responder a diferentes situações e aos diversos inputs. [11]

Em contraste com outros tipos de AI, como por exemplo modelos de ML, Reactive Machines não têm capacidade de aprender através de experiências passadas ou dados inseridos, bem como Reactive Machines não tem habilidade de superação em situações em que necessitam de alterar o seu comportamento. [11]

A nível operativo, Reactive Machines atuam de forma determinística, o que significa que, para um dado input inicial, irão produzir sempre o mesmo output ou resposta final, seguindo o conjunto de regras predefinidas. [11]

Tradicionalmente, este tipo de AI é aplicado na indústria e na robótica, onde são bastante úteis para aprimorar linhas de produção, equipamentos de soldadura precisa ou sistemas de transporte de materiais. A capacidade de atuar de forma sequencial torna as Reactive Machines altamente eficazes e precisas, o que as torna profícuas em situações onde a precisão e a velocidade são fatores determinantes. [11]

2.1.2. Limited Memory

O objetivo desta forma de AI é desenvolver sistemas que possam tomar decisões baseadas em dados históricos e em experiências passadas. Em contraste com as Reactive Machines, a Limited Memory permite a incorporação, dentro dos modelos, de algum grau de informação passada para aprimorar a tomada de decisão do modelo. Estes sistemas conseguem aprender com interações anteriores ou observações, sendo altamente utilizados para problemas que requerem adaptabilidade e capacidade de decisão assente em padrões e/ou tendências nos dados de input. [11]

Do ponto de vista operacional, sistemas de Limited Memory são capazes de armazenar dados e/ou previsões, para posteriormente serem capazes de tomar decisões corretas seguindo as relações que identificaram. No entanto, apresentam limitações na capacidade de armazenamento dos dados que utilizam para funcionar, o que limita a sua utilização a problemas específicos que exijam a percepção de resultados passados, mas sem compreender aprendizagem complexa ou adaptabilidade, características de sistemas de AI mais avançados. [11]

2.1.3. Características

Sistemas de AI de Limited Memory são tradicionalmente desenvolvidos para solucionar situações onde o sistema de AI necessita de tomar decisões ao longo de um curso de tempo, recorrendo apenas a uma quantidade limitada de decisões ou experiências passadas. [11]

Estes sistemas utilizam sumários condensados da informação mais significativa de iterações antigas para conseguir gerir a capacidade de memória do modelo, assegurando a manutenção dos aspetos mais importantes do problema. [11]

Limited Memory é um dos sistemas de AI que opera com a framework Markov Decision Processes (MDP). Numa MDP, o agente assume decisões baseadas no estado atual no momento de decisão e numa parte limitada da história de estados e ações passadas. Os sistemas de Limited Memory enfrentam escolhas entre a quantidade de informação que conseguem armazenar e a capacidade de tornar a

decisão o mais eficaz possível. Isto obriga a uma relação de compromisso entre a capacidade de armazenamento de memória do sistema e o esforço computacional necessário para o fazer correr na medida em que maior capacidade de armazenamento requer maior esforço computacional. [11]

Tradicionalmente, este tipo de AI é empregue em sistemas que pretendam dar resposta a problemas complexos, onde o armazenamento de toda a informação no que respeita à história dos dados do problema é ou demasiado excessiva ou não essencial para uma tomada de decisão precisa e eficaz. A Limited Memory é correntemente utilizada, a título de exemplo, na robótica e na autonomização de veículos. Ao garantir um sumário de experiências passadas em representações compactas, estes sistemas podem construir uma relação equilibrada entre eficácia e qualidade na decisão. [11]

2.1.4. Theory of Mind

A Theory of Mind é um conceito na Psicologia e na Ciência cognitiva que retrata a capacidade de o ser humano entender e atribuir estados emocionais a si próprio e aos outros. O objetivo de um sistema de AI funcionalmente baseado na Theory of Mind prende-se com a pretensão de dar à máquina um amplo e profundo conhecimento do comportamento humano. [11]

A nível operacional, a Theory of Mind, no contexto da AI, pretende capacitar sistemas para estarem aptos a atribuir estados mentais, intenções, crenças e emoções a eles mesmos ou a outros. Isto permite à AI assimilar que diferentes indivíduos podem ter diferentes perspetivas, conhecimentos e emoções, o que tem por consequência interações e respostas mais sofisticadas em interações humano-máquina. A essência da Theory of Mind é conceder ao modelo capacidade para simular o processo cognitivo associado à inteligência emocional humana, tornando o sistema capaz de interpretar, prever e responder ao comportamento humano de forma mais compreensiva, calculista e de acordo com o contexto em que a máquina se encontra. [11]

2.1.4.1. Características

A Theory of Mind aplicada a um Sistema de AI concede ao sistema capacidade para percecionar a perspetiva psicológica e emocional de um indivíduo, sendo assim possível a compreensão do raciocínio e emoção do mesmo, tendo como objetivo último facilitar a capacidade do ser humano percorrer interações sociais e reações com maior empatia e conhecimento. [11]

Com a utilização da Theory of Mind, é possível educar respostas sobre o que um indivíduo poderá estar a sentir, alicerçadas em comportamentos e expressões decorrentes dos acontecimentos de uma determinada situação. A Theory of Mind está intimamente conectada com a capacidade de sentir empatia, já que é a empatia que concede habilidade ao ser humano de perceber o que outros possam estar a sentir e consequentemente estruturar respostas ou sugestões empáticas que demonstrem uma aproximação compreensiva nas nossas interações e relações. [11]

A capacidade de dominar sistemas de AI, que se desenvolvem funcionalmente segundo a Theory of Mind, permite percorrer as complexidades das interações sociais, capacitando o sistema para gerir relações e tomar previsões informadas relativas à forma como um indivíduo poderá responder e reagir em diferentes contextos sociais. [11]

No entanto, a implementação da Theory of Mind em sistemas de AI está sujeita a um número amplo de dificuldades e desafios. A necessidade de modelar o comportamento intrínseco e multifacetado da capacidade cognitiva do ser humano ser altamente complexa, a juntar ao facto de não existirem evidências nítidas que possibilitem a análise da real capacidade e precisão destes sistemas tecerem con-

clusões sobre estados da mente Humana, são 2 (dois) dos principais desafios a que a implementação está sujeita.

Adicionalmente, a ambiguidade e imprevisibilidade do comportamento Humano cria uma grande dificuldade para a interpretação precisa e fiável de intenções e emoções. [11]

2.1.5. Self Aware AI

O objetivo de Sistemas de AI que funcionem assente nos princípios do Self Aware AI é conceder à Máquina capacidade de autoconsciência e consciência análoga à dos Seres- Humanos. Este objetivo torna substancialmente mais complexo este tipo de AI, já que o sistema terá de ser capaz de atuar alicerçado em regras pré-definidas e no reconhecimento de padrões nos dados, mas sem que exista autoconsciência ou alguma experiência subjetiva. A funcionalidade do Self Aware AI é que os sistemas consigam refletir sobre a sua própria existência, possuam identidade, emoções e consciência, e que porventura ainda sejam sistemas introspetivos e que procurem a evolução individual. Neste tipo de AI, que à data é apenas uma hipótese de investigação, a máquina irá ter consciência não só das emoções e estados psicológicos de outros como dos seus próprios também. [11]

Apesar de não existir nenhum princípio operacional estabelecido devido à enorme complexidade envolvente à criação de um sistema de AI com este tipo de funcionalidade, irá, com grande probabilidade, envolver o desenvolvimento de algoritmos e arquiteturas que irão potenciar a perceção, a interpretação subjetiva do mundo, a introspeção e a capacidade de evolução própria. [11]

No entanto, a criação deste tipo de sistemas de AI é ainda algo distante, devido à questionabilidade do ponto de vista ético, e a questões de natureza da consciência, identidade própria e de consciência ética para criar entidades com consciência própria. [11]

2.1.5.1. Características

Sistemas de AI de Self Aware possuem capacidade de reflexão, sendo possível que expressem opiniões sobre pensamentos próprios, experiências anteriores e identidade pessoal, adicionando capacidade de definição pessoal própria à AI. [11]

Os projetos iniciais para o desenvolvimento deste tipo de sistemas oferecem, no entanto, algumas questões e dilemas éticos, nomeadamente na perspetiva dos sistemas de AI expressarem emoções ou até a eventualidade de conseguirem sentir emoções caracterizadas pela consciência. [11]

Apesar do exposto em cima, sistemas de Self Aware AI ainda não são alcançaveis, dada a dificuldade que é a construção de modelos que possuem consciência e identidade pessoal. Os sistemas de AI baseiam a sua operação essencialmente em algoritmos e dados concretos, o que adiciona especial dificuldade à construção de modelos de Self Aware AI, não deixando, no entanto, de ser um conceito visionário que está atualmente a ser estudado no mundo profissional e académico, sendo particularmente incerto se ou quando será possível a construção e utilização deste tipo de AI. [11]

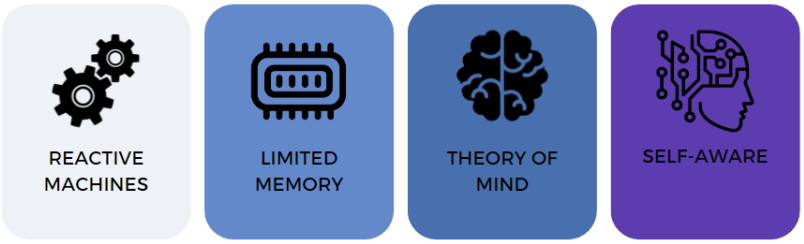


Figura 2- Diferentes tipos de AI baseados na Funcionalidade

2.2. MACHINE LEARNING

ML é um campo científico que estuda algoritmos e modelos estatísticos a que os sistemas computacionais recorrem para resolver tarefas ou problemas, sem que para isso tenham que ser explicitamente programados para o efeito. A principal vantagem da utilização de sistemas de ML é que, uma vez treinado o algoritmo para percecionar de que forma agir perante os dados de input, o sistema irá automaticamente fazer sair uma solução sem necessitar de programação específica para estes tipos de dados. [12]

Uma vez que o ML é um subcampo da AI e, portanto, uma forma de aplicar a AI, o grande foco deste campo da ciência computacional é a criação e execução de algoritmos preexistentes para atingir modelos generalistas capazes de identificar e prever padrões e diretrizes entre os dados. Algoritmos de ML são alicerçados em modelos estatísticos e matemáticos e utilizam a relação histórica entre os dados para simular o comportamento de raciocínio humano. Estes modelos atuam na base do ajuste automático e da aprendizagem para maximizar a previsibilidade e o desempenho das suas configurações, de forma a assegurar elevada eficácia nas previsões que o sistema entrega ao utilizador. [13]

O ML é de forma resumida um mecanismo para a busca de padrões e inteligência, construída numa máquina que lhe concede a capacidade de aprender, implicando que aprimore soluções como resultado da própria experiência do sistema. [14]

Na utilização de modelos de ML, é recomendável iniciar a sua aplicação com modelos mais simples, que permitam análises menos demoradas, e avaliar a necessidade de implementação de modelos mais complexos em função da eventual necessidade de um ajuste mais correto aos dados, que garanta uma resposta mais precisa ao problema ao qual é pretendido obter solução.[15]

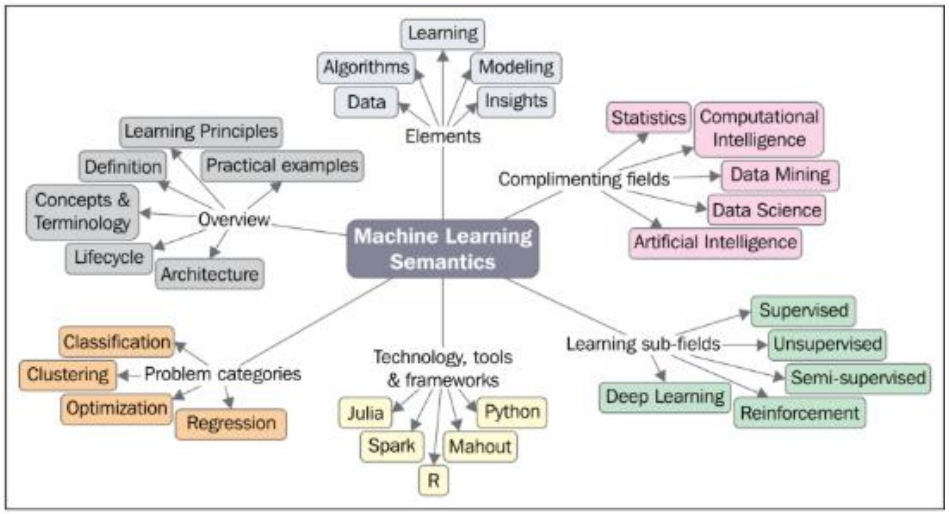


Figura 3 - Mapa de Conceitos em Inglês onde estão referidas as semânticas do ML

2.2.1. A APRENDIZAGEM EM MODELOS DE MACHINE LEARNING

Em contexto de ML, a aprendizagem pode ser compreendida como o ato de utilizar observações ou dados passados para prever relações que possibilitem alcançar uma etapa em direção a um objetivo

final concreto. É mandatário, para um sistema inteligente, ter capacidade de aprendizagem, sendo que a definição do problema deve dirigir-se pelas considerações seguintes:

1) Definir o que o sistema de ML deve aprender e a necessidade no contexto do problema dessa aprendizagem.
2) Definir a base de dados e as fontes de onde se obteve a mesma.
3) Definir se o sistema deve utilizar a base de dados como um todo, ou apenas um substrato para o processo de aprendizagem. [16]

Comumente, para o desempenho dos sistemas de ML, existem 2 (dois) tipos de base de dados primordiais. A 1ª (primeira) é, em geral, manualmente estruturada facultando ao sistema tanto os dados de input como os dados de output esperados. É imprescindível que a cada dado de input corresponda um dado de output, porque é com base neste princípio que será possível ao sistema de ML determinar a relação entre dados. A 2ª (segunda) é quando é fornecido ao sistema um dado de input e é requerido ao mesmo a previsão do output esperado. [16]

No início do processo, os dados de entrada são segregados em 3 (três) conjuntos de dados distintos, designados por Treino, Validação e Teste. Não existe nenhuma regra que restringe a percentagem de dados que deve ser restrita a treino, validação e teste, sendo usualmente utilizadas as relações de 70% (setenta) – 10% (dez) – 20% (vinte), 60% (sessenta) – 30% (trinta) – 10% (dez), 50% (cinquenta) – 25% (vinte- cinco) – 25% (vinte-cinco) respetivamente.

O conjunto de dados restritos ao Treino serve o propósito de conceder ao sistema a capacidade de aprendizagem que o possibilita identificar as relações entre os dados e criar um modelo de classificação para os referidos. Por sua parte, o conjunto de dados cedidos para o processo de validação servem para confrontar o modelo produzido no processo de Treino e verificar a precisão no processo de determinação do output. Por último, os dados utilizados no processo de Teste irão permitir aferir o desempenho do modelo de classificação.

O processo pode ser melhor explicado desta forma:

1) Fase 1 (um) – Fase de Treino: Esta fase é caracterizada pela utilização dos dados para treinar o modelo, ao emparelhar os dados de entrada com os respetivos dados de saída esperados. O output desta fase é o próprio modelo de aprendizagem.
2) Fase 2 (dois) – Fase de Validação: Esta fase é caracterizada pela apreciação da capacidade de aprendizagem do modelo e pela aferição das propriedades do mesmo, nomeadamente a medição do erro e da precisão, entre outros. O output desta fase é o modelo de aprendizagem já aprimorado.
3) Fase 3 (três) – Fase de Teste: Esta fase é caracterizada pela exposição do modelo a dados de entrada cujos dados de saída são desconhecidos, sendo depois requeridas soluções do problema, de acordo com as relações já identificadas e aprimoradas nas fases 1 (um) e 2 (dois), respetivamente. No fundo, expor o modelo de aprendizagem construído, e já validado, a dados que este desconhece e ao qual necessitará de dar uma resposta, dentro de parâmetros de precisão pré-definidos. [16]

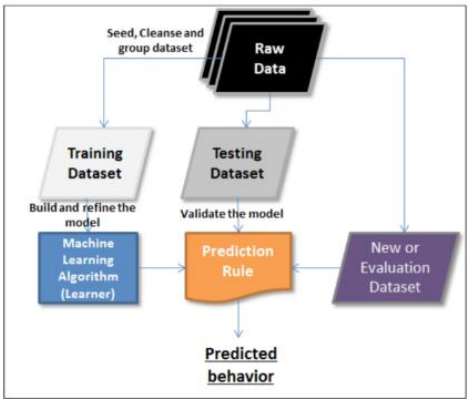


Figura 4 - Aplicação da Aprendizagem em contexto de ML, para a previsão de comportamentos

2.2.1.1. Relevância dos Dados no contexto do ML com a definição da Terminologia utilizada

A Base de Dados é a principal fonte de suporte à aprendizagem em ML. Os dados podem ser entregues aos modelos em qualquer formato, frequência ou tamanho, sendo que o 1º (primeiro) fator de relevância dos Dados é a dimensão. Tanto em bases de dados estruturadas como desestruturadas, é inevitável a presença de linhas e colunas, como a construção de matrizes exige.

É também indispensável, para a correta construção de modelos de ML, conhecer a terminologia utilizada, nomeadamente o propósito e a definição dos termos tradicionalmente empregues na construção dos modelos. [16]

- Feature, attribute, field e variable: É uma coluna singular referenciada pelo algoritmo de aprendizagem. Alguns Features podem ser o input em algoritmos de aprendizagem, sendo outros o output.
- Instance: Uma linha singular de dados presente na base de dados.
- Feature vector ou tuple: Um conjunto de Features.
- Dimension: Subconjunto dos atributos utilizados na descrição das propriedades dos dados.
- Dataset: Conjunto de linhas de dados, sendo que no contexto do ML existem diferentes tipos de Datasets para utilização em situações distintas. Um algoritmo é executado em diferentes Datasets em diferentes níveis para aferição da precisão do modelo. Como já descrito em cima, os três tipos de Datasets são: Treino, Teste e Validação.
- Data Types: Features ou Attributes podem ser de vários tipos, sendo os mais usuais:
  - Categorical (Exemplo: Novo, Velho)
  - Ordinal (Exemplo: 0, 1)
  - Numeric (Exemplo: 1.3, 2.1, 3.2)
- Coverage: A percentagem de Datasets utilizados para a construção de uma previsão na qual o modelo de ML está alicerçado. Determina a confiança no modelo de previsão. [16]

2.2.1.2. Dados Classificados e Não- Classificados

Os dados, em contexto de ML, podem ser Classificados ou Não-classificados. Dados Não-Classificados são normalmente todos aqueles que não estão atribuídos, tradicionalmente disponíveis em largas quantidades. Esta estrutura dos dados não apresenta explicação do significado em causa.

Dados Classificados, em oposição, são todos aqueles que detêm explicação do que representam. Esta classificação é obrigatória para definir a relevância dos mesmos, sendo esta classificação usualmente

determinada e catalogada com intervenção do programador. Estes dados requerem significativamente mais esforço computacional para serem obtidos do que os anteriores.

Os modelos de aprendizagem podem ser aplicados igualmente a ambos os tipos de dados, sendo que os modelos mais precisos resultam da combinação dos 2 (dois). [16]

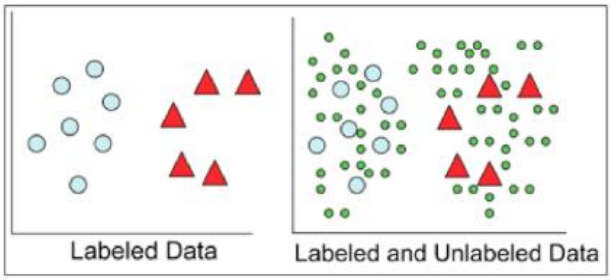


Figura 5 - Diagrama representativo das diferenças entre Dados Classificados e Não-Classificados

2.2.1.3. O que é a Tarefa em ML

A Tarefa é o problema a que a criação do algoritmo em ML pretende dar resposta. É indispensável a avaliação da confiabilidade do modelo para a resolução da Tarefa. No entanto, como diferentes algoritmos executados em diferentes bases de dados produzem modelos distintos, é relevante avaliar a consistência dos resultados com bases de dados e modelos descoincidentes e não resumir a avaliação à estrita comparação entre 2 (dois) modelos desiguais. [16]

2.2.1.4. Algoritmos de ML

A perceção clara do problema, a que é pretendido que o modelo de ML dê resposta, é o precursor para a avaliação da aplicabilidade e relevância dos dados e dos algoritmos a ser empregues no modelo. Existe uma variedade de algoritmos disponíveis, agrupados pelo tipo de aprendizagem ou tipo de problema requeridos pelo programador. Os algoritmos são aplicados de forma iterativa em diferentes bases de dados e modelos de output, que evoluem quando lhes são fornecidos novos dados. [16]

No subcampo da aprendizagem, os algoritmos são classificados em 3 (três) grandes grupos:

- Supervised Learning

Este tipo de aprendizagem é aplicado em contextos onde o objetivo de treino dos modelos passa pela utilização de um conjunto de dados caracterizados, em que para cada dado de input existe um output preciso e previamente conhecido que serve como diretriz. No decorrer do treino, o modelo conseguirá ajustar os seus parâmetros a partir da análise do erro entre as previsões consecutivas e os outputs corretos, com o recurso a técnicas como a "Backpropagation". O intuito é que o modelo, ao finalizar o treino, esteja capacitado para realizar generalizações a novos dados, desconhecidos durante o treino, executando previsões precisas. Este tipo de aprendizagem é extensivamente adotado em problemas de Classificação e Regressão, em que existe uma quantidade significativa de dados caracterizados à disposição que orientam o procedimento da aprendizagem. [3]

- Unsupervised Learning

Este tipo de aprendizagem é utilizado em modelos que dependem de dados que não se encontram caracterizados, o que exige que o modelo seja capaz de identificar relações e Padrões de forma indepen-

dente. Em contraste com a Supervised Learning, em que os algoritmos são direcionados dos dados de input para os de outputs especificados, em Unsupervised Learning o modelo processa dados primários e procura reconhecer padrões subjacentes, estruturas encobertas e ou relações particulares entre os dados. O propósito a servir é a descoberta de associações ou descomplexificar o dado em dimensões inferiores, de forma autónoma, sem uma diretriz específica e previamente identificada, para onde o modelo se deverá dirigir. [3]

- Reinforcement Learning

Este tipo de aprendizagem tem como foco a interação entre um sistema e o ambiente envolvente, em que o desígnio será potencializar uma retribuição cumulativa ao longo do tempo. De forma resumida, este tipo de aprendizagem permite ao sistema tomar decisões através de uma sequência de tentativa e erro, em que o sistema é penalizado ou recompensado com base nas ações que executa. O Reinforcement Learning é extremamente útil em tarefas em que existe a necessidade de controlar e otimizar sistemas, sendo que, no contexto da Engenharia Civil, é diretamente aplicável em situações diversas, mas também (incluindo) para a análise estrutural. [3]

Mesmo existindo a possibilidade de os modelos de aprendizagem serem construídos manualmente pelo programador, tradicionalmente, a importação de modelos já construídos é a técnica mais utilizada, essencialmente devido à existência de uma variedade de possibilidades de escolhas para cada um dos subcampos referidos e explicados anteriormente. [16]

No subcampo do tipo de problemas, os algoritmos são classificados em 4 (quatro) grandes capítulos:

- Classification Problems

Os modelos de Classificação são tradicionalmente empregues em situações onde o problema reside na identificação do conjunto previamente definido, a que um novo dado de input pertence, seguindo a premissa de que o conjunto de dados utilizado no treino contém observações em que são conhecidas as categorias a que pertencem. Num problema de Classificação, os dados de entrada podem ser divididos em 2 (duas) ou mais classes e o modelo deverá ser capaz de identificar, perante os inputs que está a receber pela 1ª (primeira) vez, a que classe pertencem os dados. [18]

- Regression Problems

Regressão é o procedimento no qual se procura encontrar a relação entre 2 (duas) ou mais variáveis de interesse, tendo por base os elementos originais que estão incorporados na base de dados. Os modelos orientam e ativam o comportamento de associação das variáveis de interesse que descrevem a relação prática entre as variáveis, tornando assim possível que o modelo consiga produzir previsões e projeções. É um método de análise e reconhecimento de relações entre 2 (duas) ou mais variáveis de interesse. Os métodos que permitem a execução de análises de regressão permitem a perceção dos aspetos mais significativos, dos mais insignificantes e de que forma o modelo promove um ou outro. [19]

Os modelos de Regressão são tradicionalmente empregues em situações onde o objetivo da resolução passa por encontrar um padrão matemático entre os dados, de forma que o modelo seja capaz de executar uma previsão do valor de uma variável. Este tipo de resolução é muito útil quando as variáveis do problema são contínuas e garante que, perante um conjunto de propriedades, o modelo consiga

executar uma previsão do valor para a variável que estiver em causa, sem que necessite de a restringir a determinados valores e concedendo alguma versatilidade na forma de a calcular.

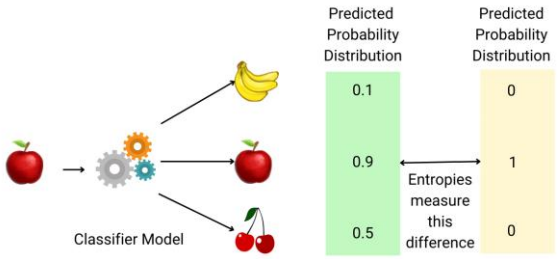


Figura 6 - Modus Operandi de um modelo de ML para dar resposta a um problema de Classificação

- Clustering Problems
- Optimization Problems

Como na presente Dissertação não serão utilizados modelos de resposta a tipos de problemas de Clustering e Optimization, estes dois tipos de problemas não serão explicados.

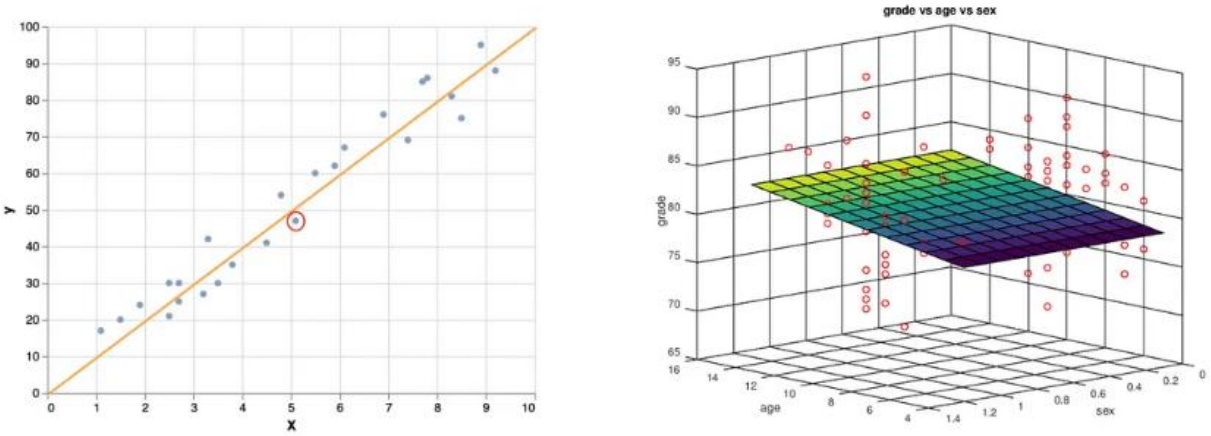


Figura 7 - Exemplos de 2 tipos de Regressão. Regressão linear à esquerda e Regressão múltipla à direita.

Existem, no entanto, uma diversidade significativa de algoritmos para solucionar cada tipo de problema, tendo em conta o tipo de aprendizagem selecionada. Cada um dos algoritmos, com os seus pontos mais e menos fortes, está pensado para dar resposta a diferentes tipos de métodos de aprendizagem e problemas, sendo, por isso necessário conhecer as propriedades dos mesmos, para que a seleção seja a mais correta. [20]

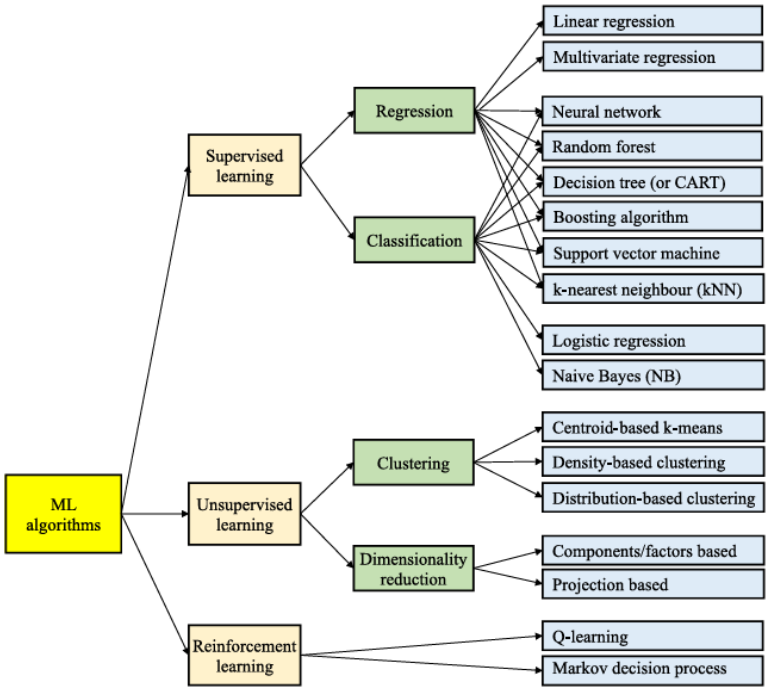


Figura 8 - Algoritmos de ML agrupados por tipo de aprendizagem e problema

Apenas os modelos que dão resposta a problemas de Regressão e Classificação foram aprofundados na realização da presente Dissertação:

- Linear Regression

Os modelos de Regressão linear são, tradicionalmente, os mais utilizados, pela facilidade na interpretação dos resultados. Este tipo de modelos só consegue dar resposta a problemas de regressão e procuram traduzir, numa equação, tendo em conta as dimensões da base de dados, a relação entre os dados num caso linear. Este tipo de modelos, de Regressão Linear, atua na procura da equação que minimize a distância dos mínimos quadrados, com o propósito de ajustar uma reta ou um plano com a máxima precisão possível. [16]

Os modelos têm como principal vantagem não exigir seleção de valores para os Hiper-parâmetros, ainda que seja possível utilizar uns, tendo como primeiro o "fit_intercept", em que o valor varia entre "true" ou "false", conforme o programador pretenda impor que para valores x=0 a base da equação seja 0 ou, em oposição, se é pretendido que o valor base seja calculado. [16]

A fácil compreensão (até 3 (três) dimensões) e a não dependência de Hiper-parâmetros, o que permite que não seja mandatório adequar o modelo conforme o surgimento de novos dados, são as principais vantagens da utilização de modelos de Regressão Linear. Dado que, na Engenharia, em geral, existe um significativo conjunto de dados que são capazes de ser compreendidos e relacionados através de Regressões Lineares, a utilização deste tipo de modelos ganha especial relevância para solucionar estes casos. [16]

No entanto, o facto de este tipo de modelos poderem atribuir a mesma relação de regressão linear a 2 (dois) conjuntos de dados totalmente distintos, conjugado com a dificuldade em simplificar representações de dados mais complexos, representa as principais desvantagens da utilização deste tipo de

modelos, havendo também a possibilidade de existir um efeito majorativo negativo na previsão de um valor da variável dependente (y), em situações onde o valor da variável independente (x) seja elevado. [16]

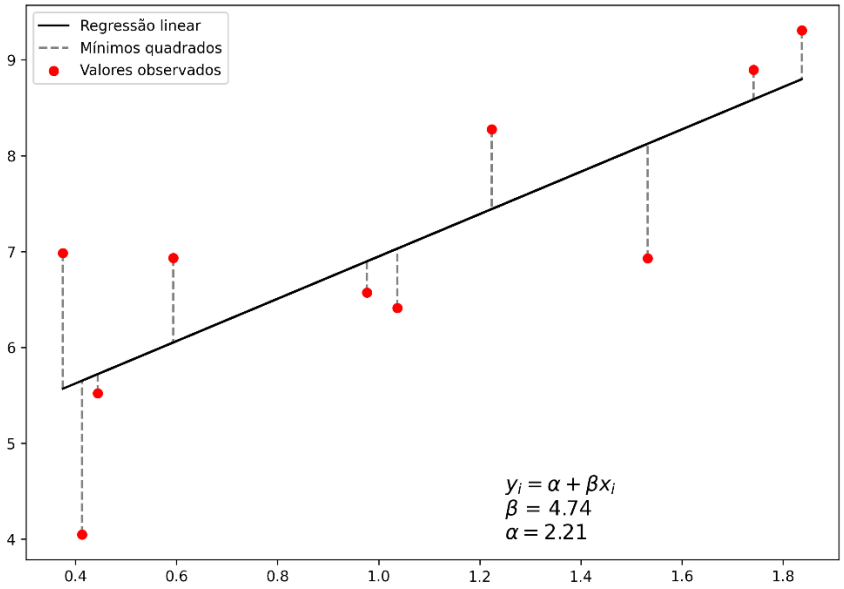


Figura 9 - Exemplo de uma solução de um modelo de Regressão Linear

- Multivariate Regression

Modelos de Regressão multivariada são tradicionalmente utilizados para dar resposta a problemas relacionados com o fator de confusão, em que a associação entre a variável independente e o resultado de interesse está distorcida por uma 3ª (terceira) variável que influencia tanto a variável dependente como a independente. Em oposição às técnicas bivariadas, como o teste t ou o teste χ^2 (qui quadrado), em que cada resultado pode ser condensado num único teste estatístico, cada modelo de Regressão produz um conjunto de resultados complexos difíceis de condensar. O ajuste de variáveis de confusão é uma das principais razões pelas quais os Métodos de Regressão Multivariada são selecionados. [24]

O objetivo em cada análise de dados é extrair, de dados primários, uma estimativa precisa. Neste contexto, a questão de maior relevância prende-se com a relação entre a variável de resposta (Y) e as variáveis explicativas (Xi), sendo uma solução para a obtenção desta relação a utilização de modelos de regressão. Existem vários tipos de análises por regressão, sendo que a escolha depende do tipo de distribuição da variável Y. Se é contínua e aproximadamente normal, é comum a utilização do modelo de Regressão Linear. Se a distribuição for dicotómica, a utilização recomendável é a do modelo de Regressão Logística. Se a distribuição seguir a Lei de Poisson ou for multinominal a análise deve ser log-linear (logarítmica – linear). Se for time-to-event data com a presença de casos censurados (tipo sobrevivência) deve ser utilizada a Regressão Cox como método de modelação. [25]

Com a utilização dos modelos, é possível prever um valor de saída (Y) baseado nos valores de um conjunto de variáveis de previsão (Xi). Estes métodos servem para avaliar o impacto de um diverso conjunto de variáveis (covariáveis e fatores) no mesmo modelo. [25]

É, no entanto, importante salientar que a utilização de modelos de regressão está tradicionalmente associada ao surgimento de alguns erros, entre os quais:

- Perda de tamanho da amostra devido à ausência de dados

- Na regressão logística, exclusão de resultados pelo facto de representar uma previsão perfeita do resultado
- Impossibilidade de grandes ou pequenas relações entre variáveis
- Valores de CIs implausíveis
- 95% do CI que inclui 1 (0 para modelos lineares) enquanto P < .05, ou vice-versa.

A Regressão é uma família de análise estatística que necessita de ser descrita com detalhe suficiente nos métodos e com a respetiva apresentação apropriada dos modelos construídos na secção de resultados. [24]

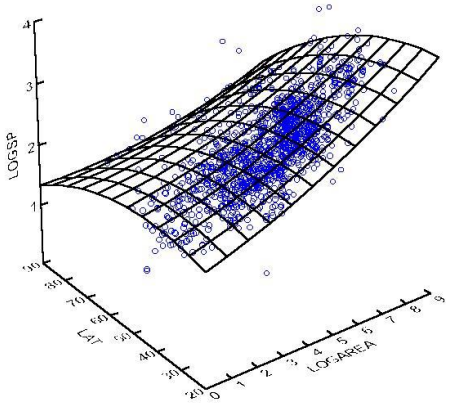


Figura 10 - Regressão Multivariada

- Neural Networks

RNAs são inspiradas na forma como o cérebro humano funciona, em especial na forma como assimila algo novo, como transmite informação e como consegue aprender a executar perante o aparecimento de uma situação ou informação que já conheceu. A composição das RNAs terá de cumprir, no mínimo, a existência de 3 (três) estratos: o estrato inicial, que é os valores de entrada que são fornecidos ao modelo, representados sob a forma de nós; uma camada intermédia, que capacita o modelo a depreender uma relação entre as variáveis que o conduzirá a um resultado; e, por fim, o estrato final, que é o valor de saída previsto e calculado pelo modelo em função dos valores colocados nos parâmetros de entrada. [2]

Cada uma das camadas é constituída por um número de nós (também denominados neurónios); estes são representados em valores e transmitem os próprios valores como sinais para os nós do estrado imediatamente a seguir aos que estão conectados pela composição das RNAs. Cada Neurónio recorre a uma Activation Function (Função de Ativação) para calcular o seu próprio valor, tendo em consideração o valor do peso e o valor de viés recebido do nó imediatamente anterior. A conexão entre Neurónios não segue uma regra fixa; no entanto, tradicionalmente, as RNAs são intensamente conectadas onde cada Neurónio de uma determinada camada está conectado a cada um dos Neurónios da camada seguinte, excetuando apenas a camada de output. O maior ou menor número de camadas intermédias de uma RNAs define, respetivamente, a maior ou menor profundidade da rede. [2]

Os Neurónios podem ter pesos negativos ou positivos, mediante estejam a excitar ou a inibir a influência na rede. Se o peso for 0 (zero), um dos neurónios não exerce influência sobre a conexão com outro neurónio. O conhecimento e a inteligência artificial que a RNAs possui estão armazenados nas conexões e pesos de cada um. [26]

A arquitetura de uma rede Neuronal reproduz o número de camadas intermédias; o número de nós ou neurónios em cada camada; as conexões entre os nós e entre as camadas em que estão inseridos; da mesma forma que as funções de ativação empregues na determinação dos pesos das conexões. A escolha da arquitetura correta, nomeadamente na determinação ótima do número de camadas intermédias e neurónios presentes em cada uma; na construção de uma RNAs, representa um importante fator de decisão, na medida em que tanto uma rede pouco profunda como uma rede exageradamente profunda originam erros: a 1ª (primeira) porque um número reduzido de camadas pode resultar em erros elevados no processo de treino; e a 2ª (segunda) porque uma rede extensa pode resultar em erros elevados na generalização de relações. [2]

O processo de treino de uma RNAs está divido em 2 (duas) fases:

- Propagação Direta:

Os neurónios utilizam os respetivos valores de peso e de viés e uma Função de Ativação, de forma a calcular os valores que, posteriormente, são transmitidos à camada seguinte , até que a camada de output produza um valor final.

- Backpropagation

Nesta fase, os neurónios ajustam os valores de peso e de viés a partir do erro entre o valor de saída e o valor real esperado. Este erro é calculado com recurso a métricas de avaliação. As 2 (duas) fases repetem-se até ao momento em que o erro seja suficientemente reduzido para que o resultado possa ser considerado válido. Cada ciclo completo deste processo é denominado como época (epoch). [2]

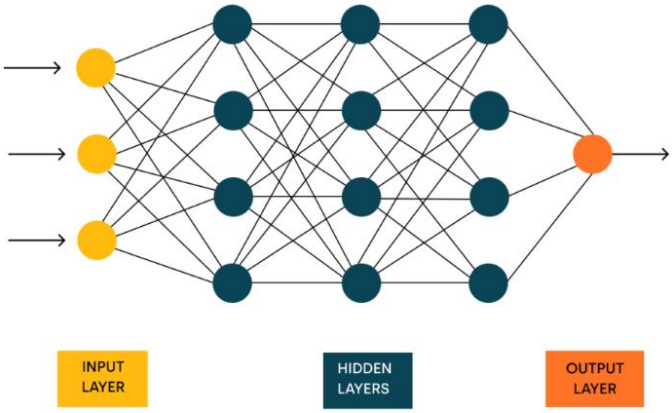


Figura 11 - Representação Gráfica de uma Rede Neuronal

- Random Forests

O algoritmo de Random Forests é um algoritmo de combinação, utilizado em modelos de ML, que agrega um conjunto de 3 classificadores. Cada ramificação contribui com 1 (um) voto unitário para a classe mais comum. Posteriormente, a combinação dos resultados resulta na classificação final. As vantagens da utilização de Random Forests estão associadas à elevada precisão nos resultados, à tolerância para valores desenquadrados e à não excedência no ajuste. [27].

A nível operacional, o Random Forests consiste num agrupamento de três classificadores estruturados, $\{h(x, \theta_k), k = 1, \dots\}$ em que $\{\theta_k\}$ são vetores aleatórios, independentes e identicamente distribuídos, e cada ramificação contribui com 1 (um) voto unitário para a classe mais frequente, correspondente ao

valor de entrada x. Como exposto acima, Random Forest é composta por uma série de classificadores estruturalmente representados sob a forma de uma ramificação de 3 (três) elementos, em que cada ramificação é construída a partir de um subconjunto de treino e de uma variável aleatória, sendo esta variável o elemento kth de uma ramificação representada pelo vetor θ_k. Isto resulta em classificadores do tipo $h(x, \theta_k)$, em que x é o vetor de entrada. Após a execução de um número k de iterações, é obtida uma sequência de classificadores $\{h_1(x), h_2(x), \dots, h_k(x)\}$ que irá compor um sistema de modelos de classificação, em que o valor de saída é decidido por votação maioritária simples entre as ramificações. A função de decisão é: [27]

$$H(x) = \arg\max_Y \sum_{i=1}^{k} I(h_i(x) = Y) \quad (2.\ 1)$$

Neste contexto, H(x) representa a combinação do modelo de classificação; h_i é um modelo único de árvore de decisão; Y é a variável de saída e I () é a função de indicatriz. [27]

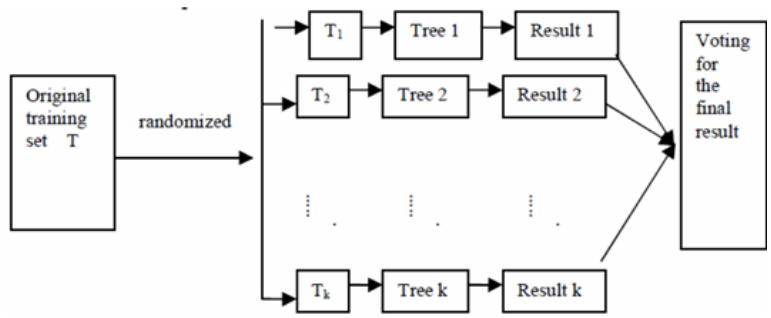


Figura 12 - Esquema Representativo de Random Forest

A função de margem quantifica, em média, o quanto a classe correta é selecionada em detrimento de uma outra qualquer classe que esteja incorreta para uma determinada entrada (X,Y). [27]

$$mg(X, Y) = \text{av}_k I(h_k(X) = Y) - \max_{j \neq Y} \text{av}_k I(h_k(X) = j) \quad (2.\ 2)$$

A precisão da previsão de classificação está diretamente relacionada com o valor de margem obtido pela resolução da equação (2. 2), sendo que um maior valor de margem representa um nível mais alto de confiança nos resultados. O erro generalizado do classificador é obtido pela equação: [27]

$$PE^{x} = P_{X,Y}(mg(X,Y) < 0) \tag{2. 3}$$

Em situações onde o número de ramificações é significativo, a sequência das previsões converge frequentemente para um valor limite, em conformidade com a Lei dos grandes números. Isto permite que a Random Forest aplicada não sofra de sobreajuste e atinja um valor de erro de generalização estável. [27]

$$P_{X,Y}\left(P_{\theta}(h_k(X,\theta) = Y) - \max_{j \neq Y} P_{\theta}\left(h(X,\theta) = j\right) < 0\right) \tag{2. 4}$$

Existe um limite superior para o erro de generalização, o que significa que, independentemente das variações que possam ser impostas nos dados ou nas construções das ramificações, o erro não irá crescer indefinidamente: [27]

$$PE^{s} \leq \left(\rho(\overline{1 - s^2})\right)/s^2 \tag{2. 5}$$

O desempenho de uma Random Forest está diretamente relacionado com a força individual de cada ramificação no conjunto final que a constitui e o grau de correlação entre as próprias ramificações. A capacidade de generalização de respostas pelo modelo é tanto maior quanto menor for o erro de generalização; dito por outras palavras, quanto mais precisas forem as ramificações e menores forem as correlações entre elas. [27]

No processo de construção de uma Random Forest, cada ramificação é treinada num subconjunto distinto do conjunto de treino original. Este subconjunto nasce ao ser retirado do original, com a utilização do método de bagging. Existem essencialmente 2 (dois) grandes fatores pela utilização do método de baggin, sendo eles: a possibilidade de utilização de dados out-of-bag para estimar o valor do erro de generalização da Random Forest, e o facto de aprimorar a precisão do modelo. [27]

Dado um conjunto original de treino T que contém N amostras, o conjunto T_k é retirado de T com substituição por bagging, com cada T_k a conter um número N de amostras. A probabilidade de uma determinada amostra não estar contida no conjunto T_k é dada por:

$$(1 - 1/N)^{N} \tag{2. 6}$$

Quando N é suficientemente largo, a probabilidade converge para e^{-1}. Existe, portanto, a probabilidade de aproximadamente 36.8 (trinta e seis ponto oito) % de qualquer amostra individual que pertence a T não estar presente em T_k, o que caracteriza as amostras como amostras out – of – bag (OOB). Tradicionalmente, os dados OOB são empregues em situações onde o objetivo é estimar o valor do erro de generalização de forma eficiente, não havendo a necessidade de se efetuar um outro conjunto de testes de forma separada. O método OOB é não enviesado, comparável na previsão ao método da validação cruzada, mas consideravelmente mais veloz. [27]

Existem mais do que um tipo de abordagem para a construção de Random Forests, dependendo da seleção das variáveis de entrada. Dentro das abordagens, destacam-se 2 (duas) principais: o Forest -RI (Random Input) e o Forest -RC (Random Combination). Existe ainda uma 3ª (terceira) em que o conceito de operacionalidade é fundado nas variáveis categóricas, no entanto, não é tanto utilizada pela complexidade e desempenho comparativamente inferior às anteriores. [27]

O Forest-RI é considerado, das 2 (duas) a serem abordadas nesta Dissertação, a mais descomplicada, aplicando uma seleção aleatória de variáveis de entrada em cada nó de uma ramificação para as dividir. A estrutura das ramificações é construída assente nos princípios da metodologia CART e permite o crescimento máximo da árvore sem existir a necessidade de poda. [27]

O Forest-FC procura aumentar a diversidade dos atributos considerados para cada divisão. Esta abordagem é especialmente útil em situações em que o número de variáveis de entrada, *M,* é diminuto. Do ponto de vista operacional, o método consiste em combinar linearmente um número *L* de variáveis de entrada, escolhidas de forma aleatória, com coeficientes que variam entre os valores de -1 (menos um) e 1 (um), o que resultará num número *F* de variáveis sobre as quais serão, posteriormente, avaliadas as divisões nos nós. Os resultados experimentais indicam que o Forest-FC possui vantagens significativas, nomeadamente o facto de ser eficaz em cenários onde as variáveis sejam de escalas diferentes; apresentar um excelente desempenho para conjuntos de dados de entrada sintéticos e, em alguns casos, consegue superar o Forest-RI a nível de precisão. [27]

Além das estratégias que utilizam os dados de entrada, é possível construir Random Forests através da utilização de estratégias nos dados de saída. A Output Smearing (Perturbação de saída) adiciona ruído branco ou gaussiano ao valor do output, enquanto a Output Flipping (Modificação de Rótulos) altera aleatoriamente algumas das caracterizações de classes dos dados de saída, sendo que é consideravelmente importante que a variável se mantenha a mesma na secção de classificação. A vantagem significativa das 2 (duas) abordagens descritas no parágrafo anterior é a capacidade em aferir a importância relativa de cada variável de entrada, o que amplia a utilidade dos métodos tanto em problemas de classificação como em problemas de regressão. [27]

A atualização dos pesos é também uma estratégia utilizada durante a construção das Random Forests, ao simplificar, acelerar e possibilitar a implementação computacional de uma forma mais simples. No entanto, a atualização dos pesos depende das características do conjunto de dados e da aprendizagem adquirida, o que torna o método vulnerável à presença de ruídos. [27]

A Subspace Random Forest permite a construção de Random Forests a partir da seleção aleatória de subespaços de atributos, o que permite que, no mesmo espaço amostral, seja exequível construir um número ilimitado de ramificações, e, por isso, ser mais capaz em comparação a árvores individuais. O aumento na complexidade estrutural assegura um crescimento praticamente monotónico na precisão geral do modelo, sendo especialmente eficaz em cenários com mais do que uma árvore (multi-tree) em que tende a atingir um desempenho ótimo. [27]

Em problemas de Regressão, a construção de modelos que seguem os princípios inerentes à utilização de Random Forest garante que, dado um espaço amostral *X* e valores de output *Y,* a composição do modelo compreende mais do que uma ramificação de decisão construída a partir de variáveis aleatórias *Θ,* onde um preditor da árvore *h(x,Θ)* gera um resultado em formato numérico. O preditor é formado ao retirar a média de um número *k* de árvores *h(x,Θ).* [27]

A função de regressão das Random Forests é dada pela expressão;

$$Y = E_{\phi}(X, \theta_1) \tag{2. 7}$$

Em termos práticos, quando o valor de *k* é significativo, a expressão pode ser alterada para:

$$Y = a\, v_k\, h(X, \theta_k) \tag{2. 8}$$

Isto explicita que um valor de correlação baixo entre os erros residuais e os erros baixos nas ramificações resulta em Random Forest de elevado valor de precisão.

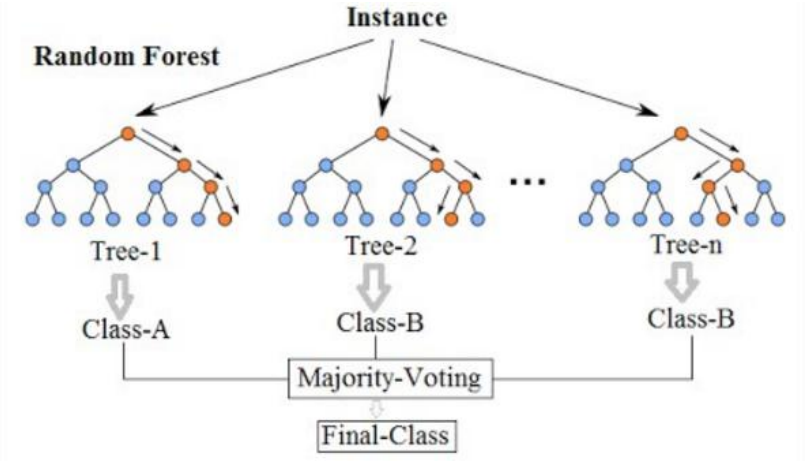


Figura 13 - Representação Gráfica de Random Forests

- Decision Tree ou CART

Decision Tree é um tipo de ML tradicionalmente empregue para solucionar problemas de classificação categórica e variáveis numéricas. É uma abordagem amplamente utilizada em data mining, em que o principal objetivo passa pela construção de um modelo capaz de prever o valor de uma variável alvo, tendo em conta mais do que uma variável de entrada. O processo de decisão do modelo funciona com base numa estrutura hierárquica similar a um fluxograma, em que a cada nó interno corresponde um atributo do conjunto de dados, e em que cada ramificação indica uma classe ou valor previsto. Decision Trees podem ser descritas, também, como a combinação de técnicas matemáticas e computacionais com o objetivo de categorizar, descrever e generalizar um determinado conjunto de dados base. [29]

A aprendizagem em Decision Tree, enquanto metodologia de ML, é frequentemente aplicada em modelos preditivos, pois permite a análise de padrões complexos. A definição dos pesos nas ramificações desenvolve-se pela análise quantitativa de múltiplos cenários, em que a seleção de atributos relevantes acontece assente em critérios estatísticos específicos. Este tipo de abordagem permite equilibrar as generalizações e previsões de variáveis, com a integração do algoritmo em conjuntos de dados específicos, a validar a eficácia em contextos reais. [29]

Os classificadores de uma Decision Tree são amplamente utilizados em situações em que a interpretabilidade é uma característica decisiva para o programador, sendo que o modelo toma decisões influenciadas, mediante as respostas que obtém a uma série de questões que coloca ao longo do processo, e que dependem das características que existem no conjunto de dados definidos para treino. É esta informação que capacita o modelo a inferir as categorias das amostras, independentemente de as variáveis serem categóricas ou numéricas. O algoritmo de decisão divide os dados a partir das características que maximizem o ganho de informação, e, posteriormente, por iteração, é possível repetir o processo em cada nó descendente, cessando quando todas as amostras que existem nesse nó, em específico, pertencem à mesma categoria ou classe. Para evitar o sobreajuste (overfitting) que pode resultar deste processo, frequentemente é definido um limite máximo para a profundidade da Decision Tree. [30]

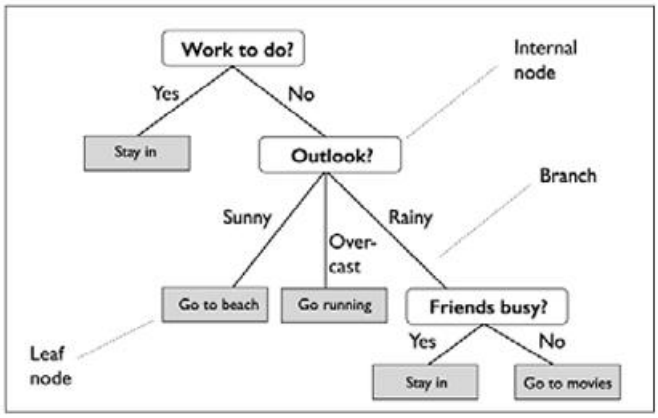


Figura 14 - Representação Gráfica do Método Operacional de uma Decision Tree

Com o objetivo de maximizar o ganho de informação é necessária a definição de uma função objetivo que permita a divisão dos nós nas suas características, de forma a contribuírem mais significativamente com informação. A função é otimizada pelo algoritmo de aprendizagem da Decision Tree. [30]

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \tag{2. 9}$$

Acima, *f* define a característica selecionada para efetuar a divisão; *Dp* e *Dj* são o conjunto de dados do nó anterior e do nó posterior, respectivamente; *I* é a medida de impureza e *Np* e *Nj* definem o número total de amostras no nó anterior e no nó posterior, respetivamente. O ganho de informação é obtido pela divisão das impurezas no nó anterior, com o somatório das impurezas nos nós posteriores, com o ganho a ser maior, consoante o menor grau de impureza nos nós posteriores. Na prática, a utilização de árvores de decisão binárias, onde cada nó anterior é dividido em 2 (dois) nós posteriores, garante uma redução de procura combinatória e por isso é maioritariamente utilizada. [30]

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right}) \tag{2. 10}$$

Os critérios de divisão maioritariamente utilizados em árvores de decisão binária são: Impureza de Gini (*IG*), entropia (*IH*) e o erro de classificação (*IE*). [30]

A Impureza de Gini é uma estratégia que limita a incidência de erros de classificação:

$$I_G(t) = \sum_{i=1}^{c} p(i|t)\big(1 - p(i|t)\big) = 1 - \sum_{i=1}^{c} p(i|t)^2 \tag{2. 11}$$

A entropia avalia a dispersão de cada classe efetiva, em que a percentagem de exemplos atribuídos à classe *i* num nó *t* é expressa por *p*(*i*|*t*). A entropia atinge o valor mínimo 0 (zero), nas situações em que no nó apenas estão contidas amostras da mesma categoria. E atinge o seu valor máximo nas situações em que todas as classes estão representadas num nó. [30]

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p\,(i|t) \tag{2. 12}$$

No exercício prático a Impureza de Gini e a entropia habitualmente originam resultados semelhantes, não sendo necessário investir recursos computacionais significativos na análise de ramificações com critérios de heterogeneidade diferentes, em detrimento da avaliação da profundidade da árvore. [30]

O erro de classificação é relevante na simplificação da estrutura de uma Decision Tree. Mas dado que não responde de forma adequada a variações nas proporções das classes nos nós, não deve ser utilizado para o desenvolvimento da árvore. [30]

$$I_E = 1 - \max\{p(i|t)\} \tag{2. 13}$$

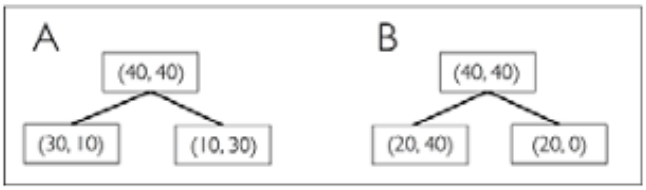


Figura 15 - Exemplo Explicativo da Utilização do Erro de Classificação na Construção de uma Decision Tree

- Boosting Algorithms

Boosting é um conjunto de técnicas de aprendizagem em modelos de ML, em que o modo de operacionalidade está firmado sobre o princípio de que a combinação de diversos classificadores simples pode resultar em desempenhos superiores em relação às situações em que cada um desses classificadores atua individualmente. Os classificadores simples são construídos por um algoritmo denominado de Weak Learner (WL) ou aprendiz fraco, que consegue produzir modelos em que o erro é ligeiramente inferior ao erro numa escolha aleatória (se a seleção for binária, ou seja, entre 2 (duas) variáveis, o erro é de 50% (cinquenta por cento)). Por oposição, um Strong Learner (SL) ou aprendiz forte é um algoritmo capacitado para criar modelos com erros significativamente mais baixos, próximos de 0 (zero), desde que exista quantidade suficiente de dados para treino. [31]

Um ensemble de classificadores é um modelo que resulta da junção e ajuste de múltiplos classificadores fracos. O boosting procura, portanto, construir mais do que um modelo simples e, posteriormente, integrá-los num singular com o propósito de alcançar, no pós-treino, um desempenho superior comparativamente ao que obteria se optasse por treinar um modelo mais complexo logo de início. Este tipo de estratégia é, em diversas situações, mais viável que treinar diretamente um classificador mais sofisticado. A título exemplificativo, em RNAs profundas, é, com alguma frequência, mais viável, a utilização de várias RNAs e subsequente ajuste dos outputs para produzir a predição final. Considerando Hm : X→{1,−1} como o m-ésimo classificador binário fraco (com m=1, …, M) e x ∈ X como uma entrada qualquer, é possível agregar as saídas H1(x), …, HM(x) numa única decisão de classe. [31]

Tendo em conta o facto de que os classificadores cometem erros nas determinações que executam quando atuam de forma independente, combinar as decisões do conjunto, com recurso ao voto maioritário, tenderá a reduzir a probabilidade de ocorrência de erros. Se for considerada uma combinação linear ponderada dos outputs dos classificadores menos complexos, normalmente denominados por classificadores fracos, é ainda possível definir uma função de predição conjunta para o sistema como um todo. [31]

$$H(x) = \text{sign}\left(\sum_{m=1}^{M} a_m H_m(x)\right) \tag{2. 14}$$

Com $\alpha_1 \ldots \alpha_m$ a tomarem o valor de um conjunto de pesos.

Quando todos os pesos atribuídos aos classificadores de um conjunto são iguais, o método de combinação resulta numa votação maioritária simples. O boosting destaca-se de outras técnicas de treino e integração de um conjunto de classificadores, não só por apresentar um desempenho mais robusto do ponto vista prático, mas também por ter integrado em si características teóricas e algorítmicas que o tornam atrativo. O princípio operacional do algoritmo é aplicar, de forma repetitiva, um classificador considerado como fraco, de acordo com a explicação exposta acima, sobre versões ponderadas do conjunto de treino, de forma que os classificadores criados em cada iteração priorizem a correção de exemplos anteriormente mal classificados. As diferentes variantes do Boosting distinguem-se, tanto pelo tipo de classificador base, quanto pelo critério adotado para a atualização repetitiva dos pesos de cada amostra. [31]

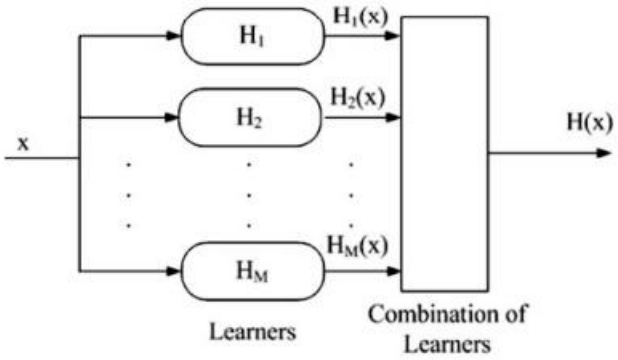


Figura 16 – Exemplo gráfico do funcionamento teórico de um Boosting Alghoritm

A terminologia do weak learning (aprendizagem fraca) e strong learning (aprendizagem forte) são centrais para a compreensão dos algoritmos de boosting. Tanto um como o outro são explicitados na teoria PAC (Probably Approximately Correct), onde é descrito uma estrutura formal de análise ao desempenho de algoritmos de aprendizagem. Um strong learning é um classificador capaz de, dado um conjunto de dados que sejam considerados suficientes, reproduzir um classificador em que o erro arbitrário seja pequeno e, por consequência, com elevada probabilidade de predição, considerando qualquer distribuição dos dados e qualquer função-alvo dentro de uma classe de hipóteses pré-definida. Por sua vez, um weak learning é descrito como um classificador que é capaz de, em situações de exigência limitada, reproduzir um classificador com desempenho ligeiramente melhor que uma escolha aleatória, restrito a um caso específico de distribuição dos dados e a uma função-alvo. [31]

O primeiro procedimento de boosting foi descrito por Schapire e é apresentado com o objetivo de demonstrar a possibilidade em alcançar um algoritmo de strong learning através da combinação de classificadores de aprendizagem reduzida, justificado com o facto de a aprendizagem em weak e strong learnings ser equivalente. [31]

No algoritmo proposto por Schapire, o conjunto de treino é dividido aleatoriamente, sem reposição, em 3 (três) subconjuntos. De início, 2 (dois) classificadores são treinados individualmente, e quando as previsões em ambos coincidem, essa é adotada como decisão final para o teste correspondente. Nos casos em que os 2 (dois) primeiros classificadores discordam, é criado um 3 (terceiro) subconjunto, que é utilizado para treinar um 3 (terceiro) classificador. Schapire conseguiu assim provar, que de

acordo com a definição da teoria PAC, este método é considerado como um método de strong learning. Se no final do teste ainda existir necessidade de fortalecer o desempenho final do modelo, o processo pode ser aplicado de forma recursiva, o que se traduz no aumento de capacidade de generalização e previsão do modelo. Posteriormente, Freund propôs uma nova variante do boosting, com melhorias significativas ao trabalho inicial de Schapire, em que a abordagem consiste na associação de vários classificadores binários, treinados em subconjuntos distintos do conjunto de dados original. No entanto, esta técnica ainda apresenta limitações significativas, como a necessidade de definir conjuntos de dados de treino volumoso de forma a garantir a diversidade dos subconjuntos utilizados. [31]

O procedimento de criação de um algoritmo de Boosting para dar resposta a um problema de classificação é descrito da seguinte forma:

Input: Dataset Z = { z_1, z_2, … z_N}, with $z_i = (x_i, y_1)$, where $x_i \in R$ e $y_i \in \{-1, +1\}$.

Output: Um classificador H : R → {-1, +1}

1) Seleciona aleatoriamente, sem substituição, $L_1 < N$ amostras de Z para obter Z^*_1
2) Executa o WL em Z^*_1, produzindo o classificador H_1.
3) Seleciona $L_2 < N$ amostras de Z, com metade das amostras mal classificadas de H_1 para obter Z^*_2.
4) Executa WL em Z^*_2, produzindo o classificador H_2.
5) Seleciona todas as amostras de Z onde H_1 e H_2 discordam, produzindo Z^*_3.
6) Executa WL em Z^*_3, produzindo o classificador H_3.
7) Produz o classificador final como votação maioritária: H(x) = $H(x) = \text{sign}\left(\sum_{b=1}^{3} H_b(x)\right)$.

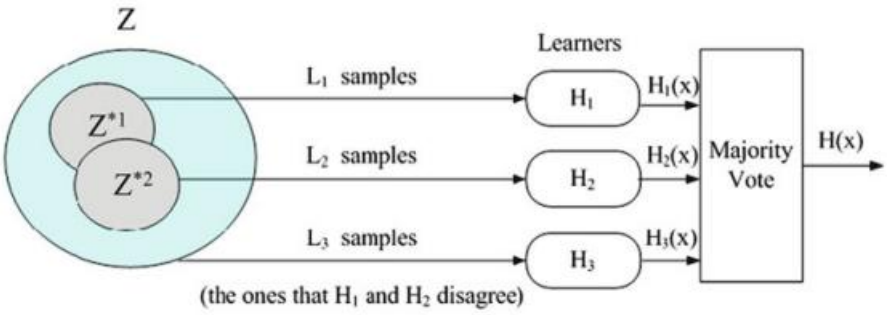


Figura 17 - Representação Gráfica do processo exposto em cima para criação de um algoritmo de Boosting para dar resposta a um problema de classificação

O algoritmo de Boosting é capaz também de dar resposta a problemas de Semi-Supervised Learning, em situações em que existe uma quantidade de dados muito significativa, mas apenas uma parte dos dados está caracterizada, com algoritmos como MixtBoost, SSMarginBoost, ASSEMBLE, SemiBoost, MultiClass Semi-Supervised Boosting, SERBoost, Information Theoretic Regularization Boosting, Multiview Boosting e Extensions on Semi-Supervised Boosting Algorithms, que não vão ser desenvolvidos pela irrelevância no contexto da presente dissertação. [31]

- Support Vector Machine

O Support Vector Machine é um algoritmo matemático que procura maximizar uma função específica, tendo em consideração um conjunto de dados fornecidos. Apesar da operacionalidade do algoritmo estar assente em fundamentação complexa, os princípios centrais do SVM podem ser compreendidos de forma intuitiva sem a necessidade de compreender formalmente o total das equações envolvidas. Para entender o funcionamento do SVM, é suficiente dominar quatro noções fundamentais do processo, o conceito de Separating Hyperplane (Hiperlano Separador), o Maximum-margin Hyperplane (Hiperlano de margem máxima), a soft margem (margem flexível) e a função kernel. [32]

A compreensão do Separating Hyperplane fica facilitada com a descrição de um exemplo teórico. Se existir um espaço em que as dimensões do mesmo são dependentes da quantidade de variáveis, se existir apenas uma variável, as distribuições dos dados ocorrem ao longo de uma linha, que pode ser divida em 2 (duas) partes por 1 (um) único ponto. A 2 (duas) dimensões, a divisão é executada através de uma linha reta. A 3 (três) dimensões a divisão é executada por um plano. Este raciocínio pode ser estendido, matematicamente, para espaços de dimensões superiores, em que a separação de classes é realizada por um hiperplano – uma generalização de linhas e planos que permite distinguir entre diferentes categorias. [32]

A lógica de representar objetos a serem classificados como pontos num espaço multidimensional e separá-los através de uma linha ou de um hiperplano não é exclusiva do algoritmo SVM. O que o diferencia de outros algoritmos de operacionalidade assente em hiperplanos é o critério a utilizar para escolher a separação adequada. Em problemas de classificação, é possível traçar mais do que uma linha para efetuar a separação de classes. A questão prende-se agora com a escolha mais adequada para criar o melhor classificador. A solução intuitiva seria a escolha da linha que se posiciona entre as classes, ou seja, a que mantém a maior distância possível dos pontos de dados que se encontram mais próximos de cada classe. Esta distância dá pelo nome de Distância de Margem, e o SVM seleciona o hiperplano que permite maximizar esta margem – conhecido como Maximum-margin Hyperplane (Hiperplano de margem máxima). Pela teoria da aprendizagem estatística, a seleção do Hiperplano de margem máxima tende a oferecer maior capacidade de generalização, o que significa um melhor desempenho na classificação de dados novos. [32]

O teorema que fundamente a eficácia das Support Vector Machines é central para o desempenho do algoritmo. A aplicabilidade do SVM envolve, como tal, o comprimento de algumas premissas centrais, nomeadamente a obrigatoriedade dos dados utilizados para treinar o modelo seguirem a mesma distribuição dos dados que são utilizados nas etapas de teste e validação. No entanto, o teorema não obriga que os dados sigam uma distribuição estatística específica como a normal, o que reforça a flexibilidade do SVM quanto à natureza estatística dos dados com os quais irá trabalhar. [32]

Apesar do exposto em cima, as considerações apresentadas partem do princípio de que os dados podem ser perfeitamente separados por um hiperplano. No entanto, essa suposição raramente é sustentada em conjuntos de dados reais e problemas reais, que frequentemente apresentam ruído ou exceções, sendo exemplo a existência de pontos nos dados que não seguem o padrão geral que é maioritariamente seguido pelas classes. Para tratar este tipo de situações, o algoritmo de SVM é adaptado pela introdução da Soft Margin (Margem flexível), que permite marginalizar alguns exemplos para que fiquem situados do "lado incorreto" do hiperplano separador sem comprometer o modelo. Esta abordagem torna o SVM mais consistente na resolução de problemas relacionados com os ruídos nos dados, ao admitir pequenas transgressões à margem de separação. No entanto, essa flexibilidade precisa de ser controlada, com a inclusão de um parâmetro que possibilita ser mais ajustável para definir o equilíbrio entre a largura da margem e a tolerância aos erros de classificação. Este parâmetro regula tanto a quantidade de exemplos aos quais é permitido ultrapassar a margem, como a extensão da violação, o que garante ao modelo melhor capacidade de generalização em contextos imperfeitos. [32]

O SVM apresenta ainda outra limitação. Num cenário unidimensional, o hiperplano de separação de margem máxima é reduzido a um único ponto, no entanto, se os dados que pertencem a classes distintas estiverem organizados de uma forma a não tornar possível a separação com um único ponto – casos em que uma classe está próxima de 0 (zero) e a distinta se distribui por valores mais elevados- o uso de margens flexíveis não resolveria o problema de separação.

Neste contexto, a função kernel surge como uma estratégia para contornar esta limitação. A função transforma os dados para um espaço de dimensão superior, em que a separação linear é possível. No caso exposto em cima, a título de exemplo, elevar os valores ao quadrado originaria uma nova dimensão, onde os dados, originalmente não separáveis, podem ser divididos com uma linha simples no novo espaço bidimensional. A função kernel atua como uma formulação matemática que permite ao SVM realizar classificações lineares em espaços transformados, ainda que os dados sejam linearmente não separáveis na sua forma original. A escolha correta de um kernel permite a projeção dos dados de forma a tornar possível a existência de um hiperplano separador nesse novo espaço, mesmo que a sua visualização não seja possível. [32]

A maior dificuldade prática na aplicabilidade do SVM é a escolha da função kernel que permita a separação correta dos dados, sem introduzir dimensões irrelevantes e/ou excessivas. A escolha do kernel, segue tradicionalmente um processo iterativo de tentativa e erro, em que inicialmente são introduzidos kernels simples e mediante a avaliação dos resultados obtidos, é acrescida ou estabilizada a complexidade dos kernels. É possível e estatisticamente rigorosa a utilização da validação cruzada, dentre um conjunto fixo de kernels, para escolha daquele que apresente melhor desempenho. No entanto, o processo é demoroso e não garante que um kernel fora do conjunto fixo não apresente melhores resultados de desempenho. Os kernels oferecem outra vantagem muito significativa para alguns problemas restritos a alguns setores profissionais, que é a possibilidade de trabalharem com inputs de dados que não sejam vetores. [32]

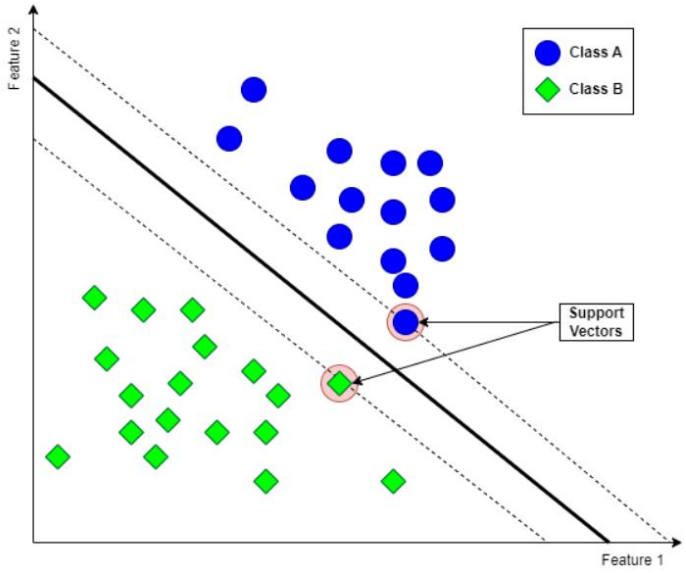


Figura 18 - Representação Gráfica de um SVM em que procura solucionar um problema simples de classificação binária

- k-nearest Neighbors

O algoritmo k-nearest Neighbors pode ser aplicado tanto em problemas de classificação como de regressão. O algoritmo segue o princípio funcional de identificar os k elementos mais próximos, no conjunto de dados de treino, em relação a um novo valor com a finalidade de conseguir estimar o seu valor ou classe. [15]

A classificação pela utilização do algoritmo kNN fundamenta a sua operacionalidade, no princípio de que os exemplos mais próximos de um ponto de interesse num conjunto de dados de treino podem fornecer informações relevantes sobre a classe a que o ponto pertence. Desta forma, o algoritmo categoriza a classe, tendo por base a classe maioritária entre os k vizinhos mais próximos, sendo essencial para o correto funcionamento deste processo utilizar uma métrica de similaridade adequada ao espaço em que os dados estão inseridos, sendo comum a utilização da métrica de Minkowski (ou norma p) em espaços R^q.. [33]

$$|x' - x_j|^p = \left(\sum_{i=1}^{q} \left|(x_i)' - (x_i)_j\right|^p\right)^{1/p} \quad (2.\ 15)$$

Em situações em que o valor de p é igual a 2 (dois) a métrica de Minkowski é equivalente à distância Euclidiana. No entanto, se o espaço amostral se alterar, é necessária a utilização de medidas de distância mais apropriadas, por exemplo, a distância de Hamming em espaços binários B^q. Nos problemas de classificação binária, é usual a utilização de um conjunto de rótulos Y = {1,-1}, e o método kNN é aplicado tendo por base a semelhança da vizinhança em relação à métrica adotada. [33]

$$\begin{aligned} f_{knn}(x') = 1 \text{se} \sum\nolimits_i \in \mathcal{N}_k(x')y_i \geq 0 \\ -1 \text{ se} \sum\nolimits_i \in \mathcal{N}_k(x')y_i < 0 \end{aligned} \quad (2.\ 16)$$

O desempenho e o comportamento do algoritmo kNN são altamente influenciados pelo valor de k, que determina o tamanho da vizinhança considerada para que o modelo efetue uma previsão. Quando k = 1 (um), a decisão é fortemente influenciada e sensível à presença de ruído ou pontos fora de padrão, especialmente nas regiões em que existe sobreposição de classes. Se o valor de k for significativamente maior, como k = 20 (vinte), o modelo ganha maior capacidade de generalização, dado que a influência de pontos minoritários é reduzida e as pequenas aglomerações de dados isoladas são ignoradas. Este comportamento é ilustrado visualmente em conjuntos de dados bidimensionais, em que diferentes regiões do espaço são classificadas tendo por base a predominância de rótulos na vizinhança. Além disso, o kNN, de forma natural, cria um diagrama de Voronoy, que divide o espaço de dados em regiões de influência, associadas aos exemplos que existem no conjunto de treino. Quando o conjunto de dados é consideravelmente extenso, o kNN necessita de identificar o número k de vizinhos mais próximos dentro de todo o espaço dos dados, o que provoca um aumento significativo do esforço computacional. No entanto, mesmo a análise dos subconjuntos já consegue fornecer estimativas satisfatórias. [33]

A decisão da escolha ideal do valor do k, que se traduz na dimensão da vizinhança que proporciona o melhor desempenho em classificação, está diretamente relacionada com o processo de seleção do modelo, em que os métodos como validação cruzada são amplamente utilizados na determinação da combinação mais eficaz dos parâmetros e com isso alcançar um maior valor de precisão. [33]

O algoritmo kNN também é aplicável a problemas de classificação multi-classe. Neste cenário, em cada nova amostra X, o modelo atribui à nova amostra o rótulo da classe mais frequente entre os k vizinhos mais próximos que pertencem ao espaço de dados. Esta abordagem permite ao kNN generalizar soluções em contextos em que existam mais de 2 (duas) categorias de saída, desde que o valor de k e as distribuições dos dados sejam escolhidos de forma adequada. [33]

$$f_{\text{KNN}}(x') = \arg\max_{y \in \mathcal{Y}} \sum_{i \in \mathcal{N}_{\mathcal{K}}(x')} I(y_i = y) \tag{2. 17}$$

onde I(·) é uma função indicadora que assume o valor 1 quando a condição de $y_i = y$ é satisfeita, e 0 caso contrário. [33]

A resolução de um problema de regressão através da utilização do algoritmo kNN é uma tarefa intimamente relacionada com a resolução de problemas de classificação, diferenciando-se principalmente no tipo de saída. Enquanto a classificação associa padrões nos dados a rótulos de classes discretas, a regressão procura mapear os padrões em valores contínuos num subespaço R^d. Apesar das limitações práticas que resultam da representação dos valores previstos em conjuntos discretos, a principal distinção reside no facto de a regressão trabalhar com um conjunto de rótulos significativamente maior. O objetivo é prever um vetor $y \in R^d$ para uma nova entrada nunca antes vista $x \in R^q$, utilizando para esse efeito um conjunto de observações rotuladas do tipo $\{(x_1, y_1), \dots, (x_N, y_N)\}$. O objetivo de uma Regressão é estimar uma função $f: R^q \rightarrow R^d$, que consiga mapear padrões de entrada e converte-los para valores contínuos. Em casos de regressão em que o algoritmo utilizado para dar resposta ao problema seja o kNN, a previsão para um novo ponto x é obtida através do cálculo da média dos valores da função correspondente aos vizinhos k mais próximos de x. [33]

$$fkNN(x') = \frac{1}{K} \sum_{i \in \mathcal{N}_{\mathcal{K}}(x')} y_i \tag{2. 18}$$

Em que conjunto $\mathcal{N}_{\mathcal{K}}(x)$ representa os índices do k vizinhos mais próximos de x. A ideia, na qual o algoritmo kNN se fundamenta para operar na resolução de problemas de regressão, é que em regiões locais do espaço de entrada e saída, os dados apresentem comportamentos semelhantes, ou seja, é esperado que amostras próximas compartilhem rótulos contínuos parecidos. Assim, para a previsão do valor de ouput de uma nova entrada x, o algoritmo utiliza a média dos rótulos das amostras mais próximas, que já existam na vizinhança. O parâmetro k exerce, por isso, uma forte influência nos resultados que o modelo prevê. Em situações em que k=1 (um), o modelo tende a ajustar-se de forma excessiva ao valor do vizinho que se encontre mais próximo (overfitting). Já quando k=N, o resultado é assumido como uma média global do conjunto total dos dados. A visualização do impacto da escolha do valor de k é visualizado através da comparação de diferentes tamanhos de vizinhança. Variações como a regressão kNN ponderada podem contribuir para reproduzir regiões com valores constantes (platôs). [33]

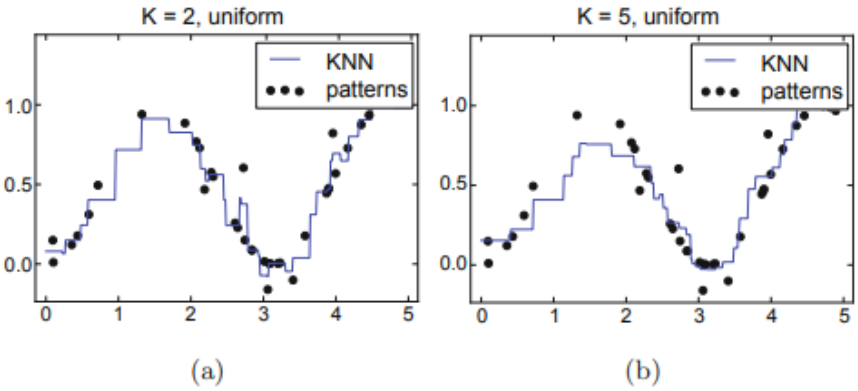


Figura 19 - Ilustração de um algoritmo kNN uniforme para regressão para (a) k=2 e (b) k=5

Como para a resolução do problema proposto no âmbito da presente dissertação, o algoritmo utilizado foi uma rede neuronal, que vai ser explicitada à frente, o próximo subcapítulo é restrito à explicação do papel decisivo que as Funções de Ativação e Otimização desempenham no treino de RNAs.

2.2.1.5. Funções de Ativação e Otimização

As Activation Functions (Funções de Ativação) são essenciais em redes neuronais do tipo feedforward, porque determinam a saída dos neurónios, tendo por base os pesos e os viés atribuídos a cada um deles. [2] Das múltiplas funções de ativação disponíveis, a ReLU destaca-se como a mais utilizada e eficaz, tendo capacidade de superação comparativamente a funções como Sigmoid, Tanh e Softplus, particularmente por apresentar o menor valor do erro quadrático médio após 1000 (mil) épocas de treino. A ReLU também oferece vantagens do ponto de vista computacional em comparação com as funções identificadas anteriormente, dado que evita o uso de operações exponenciais e em redes profundas apresenta melhor performance, justificado pelo facto de ser mais eficiente na convergência dos pesos. [34]

Não obstante o parágrafo anterior, a função Sigmoid possui características vantajosas em determinados contextos, nomeadamente a natureza da função estar limitada entre 0 (zero) e 1 (um), o que previne ativações excessivas. A título de exemplo, para problemas relacionados com a monitorização estrutural, redes simples em que apenas existe uma camada oculta e função de ativação log-sigmoid conseguem dar uma resposta adequada na previsão de falhas de rigidez em edifícios. No entanto, a função Sigmoid sofre com vanishing gradient (gradiente que desaparece), devido ao facto de a derivada máxima da função ser igual a 0.25 (zero ponto vinte-e-cinco), o que, por fim último, leva a que em redes profundas os gradientes sejam cada vez menores, dificultando ou até impossibilitando a eficácia no processo de treino. Em oposição, a ReLU não enfrenta esta limitação devido ao facto de a ativação não ser restringida a um intervalo específico, o que a torna mais capaz para solucionar problemas que obriguem a arquiteturas de rede Neuronal com múltiplas camadas ocultas. [34]

As funções de ativação mais frequentes podem ser agrupadas em duas grandes categorias: Funções Definidas por Partes e Funções Localmente Quadráticas. As funções definidas por partes são compostas por segmentos lineares definidos sobre intervalos específicos e frequentemente com o mesmo tamanho. O exemplo clássico deste tipo de funções é a ReLU, expressa por $\mathrm{ReLU}(x) = \max(0, x)$, e destacam-se pela ausência de curvatura nos intervalos definidos, exibindo derivadas de primeira ordem constantes em cada segmento. [34]

As funções de ativação não lineares, que apresentam suavidade e derivada de segunda ordem diferente de 0 (zero), são consideradas localmente quadráticas. Os exemplos tradicionais incluem a função Swish:

$$\text{Swish}(x) = \frac{x}{1 + e^{-x}} \tag{2. 19}$$

E a função Mish:

$$\text{Mish}(z) = z \tanh(\log(1 + e^{z})) \tag{2. 20}$$

Em que ambas as expressões envolvem componentes suaves e contínuas que contribuem para que sejam consideradas como localmente quadráticas.

Em determinados casos específicos, um número reduzido de funções lineares por partes, como a ELU (Exponential Linear Unit) podem ser classificadas como localmente quadráticas quando apresentam curvatura, ou seja, a segunda derivada da função é não nula em pelo menos um dos seus intervalos. Esta propriedade garante que as funções se aproximem efetivamente de funções quadráticas, contribuindo de forma significativa para a capacidade de aprendizagem em redes neuronais. [34]

No contexto da presente dissertação, vão ser explicitadas as funções mais relevantes no aprimoramento do processo de aprendizagem em redes neuronais. [34]

- Sigmoid

A função de ativação Sigmoid transforma os inputs que recebe, normalmente no intervalo aproximado a (-8, +8), para uma faixa de valores entre 0 (zero) e 1 (um). É uma função linear com derivada contínua e suave. No entanto, o comportamento de compressão dos valores de saída, também denominado por "saturação" pode conduzir a um problema de desvanecimento do gradiente, especialmente em redes profundas, o que dificulta a otimização dos pesos na rede e que conduz por fim último à ineficácia do treino com o aumento da profundidade da rede. [34]

$$\sigma(x) = \frac{1}{1 + e^{-\frac{x}{x_0}}} \tag{2. 21}$$

- Hyperbolic Tangent

A função Hyperbolci Tangent (tanh) é semelhante à função Sigmoid, mas desenvolve o mapeamento dos valores de entrada para um intervalo de saída entre -1 (menos um) e +1 (um). Uma das principais vantagens da tanh em relação à Sigmoid reside no facto da inclinação da derivada ser mais acentuada, o que poderá resultar no aumento da expressividade dos gradientes durante o processo de treino da rede. Não obstante esta característica, o problema exposto no caso anterior do desvanecimento do gradiente permanece, dada a natureza limitada da saída da função. [34]

$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2. 22}$$

- Rectified Linear Unit (ReLU)

A função ReLU é extensivamente adotada para o aprimoramento da aprendizagem tanto em redes neuronais de arquitetura mais simples, como em redes neuronais de arquitetura mais complexa, principalmente por apresentar um desempenho superior em treino quando comparada a funções como a Sigmoid ou a tanh. A definição da operacionalidade é simples, a função retorna o próprio valor de entrada quando positivo e zero caso contrário, e é a simplicidade operacional que permite que a função produza saídas nulas reais, o que contribui para a dispersão representacional de valores nas camadas ocultas das redes neuronais. [34]

Não obstante o exposto em cima, não é recomendável a utilização da função ReLU em redes neuronais recorrentes, ainda que os valores de saída possam variar num intervalo de 0 (zero) até ∞ (infinito), o que favorece a eficiência computacional dos modelos onde a função estiver implantada, a ausência de limites superiores pode causar Gradient explosion (explosão de gradiente).[34]

Um outro ponto negativo relevante prende-se com um fenómeno conhecido como "dying ReLU" e que está associado ao problema do desvanecimento do gradiente. Desenvolve-se em neurónios que recebem continuamente valores negativos e que em decorrência disso começam a produzir saída de valor 0 (zero) em permanência, o que dificulta a recuperação durante a fase de treino. [34]

- Parametric leaky version of a Rectified Linear Unit

A função de ativação PReLU (Parametric Rectified Linear Unit) foi proposta como uma solução para o problema do "dying ReLU. Ao contrário da ReLU tradicional, que anula todos os valores negativos, a PReLU introduz um valor pequeno de declive (α) no intervalo negativo da função. O parâmetro α é ajustado no decorrer do processo de treino, através de back propagation, de forma a conferir à rede maior capacidade de aprendizagem. [34]

Nos casos em que o coeficiente α for fixo em vez de ajustável, a função dá pelo nome de Leaky ReLU.

$$\mathrm{PReLU}(z) = \max(0, z) + a \cdot \min(0, z) \tag{2. 23}$$

- Exponential Linear Unit

A função de ativação ELU (Exponencial Linear Unit) serve como uma alternativa à função ReLU, como forma de tentar superar algumas das suas limitações. Tal como a ReLU, a função ELU retorna o valor de entrada nas situações em que este é não negativo. No entanto, para valores de entrada negativos, a função ELU apresenta um comportamento exponencial que assegura uma transição com maior continuidade. Esta característica permite produzir saídas negativas, que permitem a mitigação da problemática da existência de neurónios "mortos". A expressão da função é dada pela seguinte fórmula: [34]

$$ELU(x) = \max(0, x) + \min\big(0, \alpha(e^{x} - 1)\big) \tag{2. 24}$$

- Scaled Exponential Linear Unit

A função SELU (Scaled Exponential Linear Unit) é de utilização específica em Self-Normalizing Neural Networks (Redes Neuronais Auto-Normalizantes). A função possui características específicas, como um valor fixo para $\alpha \approx 1.67326$ e $\lambda \approx 1.05070$, fundamentais na garantia de que as ativações, ao

atravessarem as diversas estratificações de uma rede, mantenham propriedades estatísticas estáveis, especificamente média próxima de 0 e variância unitária. Isto ajuda na auto-normalização da rede, mesmo com a existência de perturbações e/ou ruídos. [34]

A SELU apresenta sinais positivos de desempenho em redes neuronais totalmente conectadas (feedforward), no entanto para outras arquiteturas ou em resolução de tarefas específicas, os valores de α e λ podem necessitar de ajuste. Existem estudos sobre a aplicabilidade da SELU em redes convulsionais, mas a investigação ainda é pouco aprofundada para se retirarem conclusões sobre a aplicabilidade da função a este tipo de redes neuronais. [34]

$$SELU(x) = \gamma \cdot \big(\max(0, x) + \min\big(0, \alpha(e^x - 1)\big)\big) \tag{2.25}$$

- Swish

A função Swish, é uma função de ativação autorregulável desenvolvida através de técnicas de aprendizagem por reforço. A função não possui limite superior, é contínua e diferenciável, ao contrário da ReLU, no entanto partilha com esta última o facto de ser limitada inferiormente e ilimitada superiormente. Estudos já efetuados, provam a utilidade da função para conseguir resultados mais consistentes em diferentes camadas da rede. [34]

- Mish

A função Mish é uma função de ativação recente, que apresenta formato e características semelhantes à função Swish. A principal distinção entre as duas, existe na região côncava da curva. Apesar da função Mish exigir maior esforço computacional, apresenta melhores resultados de consistência e desempenho em comparação com a ReLU e a Swish. [34]

Name	Equation	Derivative	Range	Order of Continuity	Monotonic derivative
Sigmoid	$\frac{1}{1+e^{-x}}$	$\frac{e^{-x}}{(1+e^{-x})^2}$	$(-\infty, +\infty)$	C^∞	Yes
Hyperbolic Tangent	$\frac{2}{1+e^{-2x}} - 1$	$\frac{1}{\cosh^2 x}$	$(-1, 1)$	C^∞	Yes
ReLU [6] [5]	$\max(0, x)$	$\begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases}$	$[0, \infty)$	C^0	Yes
Leaky ReLU [6]	$\max(0, x) + \alpha \min(0, x)$	$\begin{cases} 1, & \text{if } x \geq 0 \\ \alpha, & \text{otherwise} \end{cases}$	$(-\infty, +\infty)$	C^0	Yes
PReLU [6]	$\max(0, x) + \alpha \times \min(0, x)$	$\begin{cases} 1, & \text{if } x \geq 0 \\ \alpha, & \text{otherwise} \end{cases}$	$(-\infty, +\infty)$	C^0	Yes
ReLU6 [12]	$\min(\max(0, x), 6)$	$\begin{cases} 0, & \text{if } x < 0 \text{ or } x > 6 \\ 1, & \text{if } 6 \geq x \geq 0 \end{cases}$	$[0, 6)$	C^0	Yes
ELU [7]	$\max(0, x) + \min(0, \alpha(e^x - 1))$	$\begin{cases} \alpha e^x, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$	$(-\alpha, +\infty)$	$\begin{cases} C^1, & \text{if } \alpha 1 \\ C^0, & \text{if } \alpha \neq 1 \end{cases}$	Yes if $\alpha \geq 1$
SELU [13] [1]	$\gamma(\max(0, x) + \min(0, \alpha(e^x - 1)))$	$\gamma \begin{cases} \alpha e^x, & \text{if } x < 0 \\ 1, & \text{if } x \geq 0 \end{cases}$	$(-\gamma\alpha, +\infty)$	C^0	Yes
Swish [10]	$\frac{x}{1+e^{-x}}$	$\frac{e^x(x+e^x+1)}{(e^x+1)^2}$	$(\approx -0.2784, +\infty)$	C^∞	No
Mish [11]	$x \tanh(\log(1+e^x))$	$\frac{e^x(4e^x x+4x+6e^x+4e^{2x}+e^{3x}+4)}{(2e^x+e^{2x}+2)^2}$	$(\approx -0.3088, +\infty)$	C^∞	No

Figura 20 - Tabela sumário das Funções de Ativação

Além das funções de ativação, também é possível e recomendável ajustar parâmetros como as funções de otimização, que otimizam o processo de retro propagação, na procura de alcançar melhores resultados no treino, que se reproduzam numa maior capacidade de generalização do modelo. O Stoachastic Gradient Descent (SGD) é um dos optimizadores mais simples e mais utilizados na otimização de redes neuronais, apesar da eficácia ser dependente da correta definição de vários parâmetros. [2]

No entanto, estudos de comparação efetuados entre os optimizadores mais utilizados e conhecidos como o SGD, AdaGrad, Adadelta, RMSProp e Adam indicam que na teoria, o Adam tende a oferecer um desempenho global superior na fase de treino. No entanto, a escolha do optimizador mais eficaz é variável consoante a natureza do problema e as condições de treino definidas. [2]

- Stochastic Gradient Descent

A função de otimização SGD é uma simplificação significativa dos métodos tradicionais. Ao invés de calcular o valor exato do gradiente da função do erro total, o SGD estima o valor do gradiente em cada iteração, utilizando apenas um exemplo aleatório selecionado do conjunto de dados. [36]

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t) \tag{2. 26}$$

O processo estocástico definidor da atualização dos pesos $\{w_t\}$ é influenciado pelos exemplos selecionados de forma aleatória em cada iteração. Embora a aleatoriedade introduzida tenha peso, é esperado que o algoritmo se aproxime do valor esperado. Uma das vantagens deste método é que não existe necessidade de armazenar os exemplos utilizados anteriormente, o que permite o processamento contínuo dos dados em tempo real, como em sistemas de implante. Nestas circunstâncias, o gradiente estocástico realiza a otimização diretamente sobre a média teórica do erro do modelo em toda a distribuição dos dados, que provêem de situações consideradas como verdadeiras, uma vez que a escolha aleatória de um exemplo utilizado no treino é uma aproximação prática do comportamento dos dados em situação real. [36]

No momento atual, em que o volume de dados disponíveis tem aumentado a um ritmo sem precedentes, a gestão do tempo é obrigada a considerar tanto o volume de dados a utilizar como o esforço e tempo computacional. Neste cenário a utilização do SGD apresenta um desempenho particularmente eficaz quando confrontado com a realidade exposta em cima. [36]

- Adaptive Gradient Algorithm

A função de otimização AdaGrad é uma função que funciona na adaptação da taxa de aprendizagem em cada parâmetro de forma individualizada, através do recurso ao acumulado histórico dos gradientes. Especificamente, o optimizador guarda os quadrados dos gradientes numa matriz diagonal $G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^\top$, o que lhe confere capacidade de ajuste em cada componente do vetor dos parâmetros. [37]

A atualização dos parâmetros para o intervalo de tempo t +1 é dada por:

$$x_{t+1} = \Pi_X^{G_t^{1/2}} \left(x_t - \eta G_t^{-1/2} g_t\right) \tag{2. 27}$$

Onde $\Pi_X^A(y) = \arg\min_{x \in X} |x - y|_A$ representa a projeção do ponto y sobre o conjunto X através da norma Mahalanobis associada à matriz A, e η adota o valor da taxa inicial de aprendizagem.[37]

No entanto, derivado ao curso computacional da raiz quadrada e inversa da matriz G_t, quando a dimensão é significativa, o método é simplificado com a utilização única da diagonal de G_t, o que resulta em maior eficiência computacional, principalmente nas situações em que os gradientes são esparsos. [37]

- Adadelta

O Adadelta é um algoritmo de otimização que ajusta de forma automática a taxa de aprendizagem ao longo de um ciclo temporal e para cada parâmetro individualmente, e foi desenvolvida tendo como mote a melhoria da eficiência dos métodos de diminuição do gradiente. A característica mais relevante do optimizador é a capacidade em ajustar automaticamente a taxa de aprendizagem de cada parâmetro com base no histórico dos gradientes calculados, tornando assim dispensável a necessidade de definição manual deste valor. Este tipo de abordagem é eficaz na gestão de gradientes ruidosos, sendo aplicável em diversos tipos de algoritmos e conjuntos de dados, o que torna o optimizador uma solução robusta em múltiplos contextos. [38]

Do ponto de vista das vantagens, o Adadelta acrescenta o facto de ser bom a trabalhar com dados dispersos, ao superar problemas que o AdaGrad enfrenta em gradientes esparsos e a facilidade de utilização por não depender tanto de tuning em Hiper parâmetros. No entanto, apresenta algumas limitações, nomeadamente ser um optimizador de convergência lenta e necessitar de memória e esforço computacional significativos. [38]

- RMSProp

O otimizador RMSProp atua de forma significativa na redução das oscilações verticais durante o processo de ajuste dos pesos, o que permite a utilização de um valor maior para a definição da taxa de aprendizagem. Isto traduz-se num contributo para o avanço mais significativo no eixo horizontal, que por consequência acelera a convergência do algoritmo em questão. O valor do momento, referenciado como β, é ajustado a 0.95, o que condiciona o fator de amortecimento 1 – β a ser igual a 0.005. Já a taxa de aprendizagem α é fixada em 0.001, um valor selecionado tendo por base o facto de melhorar o desempenho do ajuste durante a fase de treino.

Durante a Backpropagation, o optimizador atualiza os parâmetros do modelo, incluindo pesos (W) e o viés (β), através de equações específicas a serem apresentadas no final do presente parágrafo. As variáveis V_{dw} e V_{db}, que correspondem aos valores das médias móveis dos gradientes dos pesos e das tendências, resultam da resolução das equações auxiliares e são utilizadas com o objetivo de minimizar a função de perda. Por sua vez, as derivadas *dw* e *db* representam os gradientes dos pesos e das tendências para a direção de redução da perda, respetivamente. O ε é adicionado aos denominadores para evitar que se efetuem divisões pelo valor de 0 ou que se obtenham resultados numéricos instáveis.

$$W = W_{t-1} - \alpha \cdot \frac{dw}{\sqrt{vdw + \varepsilon}} \tag{2. 28}$$

$$b = b_{t-1} - \alpha \cdot \frac{db}{\sqrt{vdb + \varepsilon}} \tag{2. 29}$$

$$vdw = \beta \cdot vdw + (1 - \beta) \cdot (dw)^2 \tag{2. 30}$$

$$vdb = \beta \cdot vdb + (1 - \beta) \cdot (db)^2 \tag{2. 31}$$

- Adaptive Moment Estimation

O ADAM é um algoritmo eficiente de otimização estocástica, que utiliza apenas gradientes de primeira ordem e é pouco exigente do ponto de vista computacional. Este método calcula taxas de aprendizagem adaptativas para cada parâmetro, tendo por base estimativas dos primeiros e segundos momentos dos gradientes. O nome ADAM resulta precisamente desta abordagem de estimação adaptativa dos momentos. [41]

O optimizador ADAM inclui mais do que um parâmetro configurável. β-s define a quantidade de conhecimento que é transportado de uma época da fase de treino para a seguinte, ou quantifica exponencialmente a quantidade de informação perdida da época anterior para a seguinte. A taxa de aprendizagem (h), é outro Hiper parâmetro que influencia a taxa de convergência. O ADAM é um optimizador de aprendizagem que adapta a taxa de aprendizagem tendo como fundamento a velocidade $\left(\frac{h}{\sqrt{v}}\right)$. Devido à dependência de épocas anteriores, o ADAM é por natureza um sistema dinâmico. [40]

O ADAM pode ser escrito desta forma:

$$g_{n+1} = \nabla_{\theta \mathcal{L}}(t_{n+1}, \theta_n) \tag{2. 32}$$

$$m_{n+1} = (1 - \beta_1) \sum_{i=0}^{n+1} \beta_1^i g_{n+1-i} \tag{2. 33}$$

$$v_{n+1} = (1 - \beta_2) \sum_{i=0}^{n+1} \beta_2^i g_{n+1-i}^2 \tag{2. 34}$$

$$\theta_{n+1} = \theta_n - h \frac{(1 - \beta_1) \sum_{i=0}^{n+1} \beta_1^i g_{n+1-i}}{\sqrt{(1 - \beta_2) \sum_{i=0}^{n+1} \beta_2^i g_{n+1-i}^2}} \tag{2. 35}$$

Onde utilizada de forma abreviada a seguinte notação:

$$\mathcal{L}(t_{n+1}, \theta) \coloneqq \nabla_{\theta \mathcal{L}}\left([X, Y]_{n+1}, \widehat{Y_{n+1}}; \theta\right) \tag{2. 36}$$

Em que $t_{n+1} = t_n + h$. A função de perda é utilizada para ajustar os Hiper parâmetros β, e por consequência os valores de α's utilizados. Estudos científicos já efetuados demonstram que o SGD consegue superar os valores originais de β num problema artificial. [40] Para valores de taxa de aprendizagem significativamente pequenos, ($h \rightarrow 0$) , o algoritmo ADAM pode ser interpretado como uma discretização numérica da equação diferencial ordinária exposta a seguir: [40]

$$\dot{\theta}(t) = -\frac{m(t)}{\sqrt{v(t) + \epsilon}} \tag{2. 37}$$

$$\dot{m}(t) = \dot{a}(t)\nabla_\theta \mathcal{L}\big(t, \theta(t)\big) - r(t)m(t) \tag{2. 38}$$

$$\dot{v}(t) = p(t)\big[\nabla_\theta \mathcal{L}\big(t, \theta(t)\big)\big]^{\alpha 2} - q(t)v(t) \tag{2. 39}$$

Onde $[\cdot]\circ 2$ representa a operação de elevar ao quadrado elemento a elemento, $\circ\sqrt{(3)}\cdot$ indica a raiz quadrada aplicada elemento a elemento, $[\cdot]/[\cdot]$ corresponde à divisão elemento a elemento, e 't' refere-se à componente temporal em contexto contínuo. É possível também utilizar uma notação abreviada: [40]

$$\mathcal{L}(t, \theta) \coloneqq \mathcal{L}([X_t, Y_t], Y_t; \theta) \tag{2. 40}$$

Neste contexto,Y_t, X_t, $\widehat{Y}_t$,correspondem, respetivamente, às entradas e saídas dos dados no instante de tempo t, bem como à saída criada pelo modelo nesse mesmo instante. O termo ϵ do denominador da primeira equação tem como objetivo garantir a estabilidade numérica. [40]

O optimizador ADAM utiliza os termos de correção para compensar o viés decorrente da condição inicial. A seguir, é deduzido o termo que corresponde à estimativa do 2º momento; a dedução do primeiro é análoga. Se considerarmos g o gradiente da função objetivo estocástica f, a estimativa do 2º momento bruto (variância não centrada) é calculada pela média móvel exponencial dos quadrados dos gradientes, em que é imposto o valor da taxa de decaimento β_2. A média móvel exponencial é iniciada como v_0, um vetor de zeros, e a atualização consecutiva em função do instante t é dada pela expressão (2.34). O 2º momento $E[g_t^2]$ é dado pela expressão: [41]

$$E[g_t^2] \cdot (1 - \beta_t^2) + \zeta \tag{2. 41}$$

Onde ζ=0 no caso do segundo momento E[gi2] ser estacionário; para casos distintos, ζ pode assumir um valor reduzido, dado que a taxa de decaimento exponencial β1 deve ser selecionada de forma a que a média móvel exponencial dos quadrados dos gradientes atribua um valor reduzido para os pesos dos gradientes muito distantes no tempo. O termo que permanece é $(1-\beta t2)$, que resulta da inicialização da média com valores zero. Por esta razão, no Algoritmo, divide-se por este fator, com o intuito de corrigir o viés decorrente da condição inicial. No caso de gradientes esparsos, para obter uma estimativa fiável do segundo momento, é necessário calcular a média sobre um número muito significativo de gradientes, o que implica escolher um valor reduzido para β2; contudo, é precisamente neste cenário — com β2 a assumir um valor pequeno — que a ausência de correção do viés decorrente da condição inicial pode originar passos iniciais excessivamente demorados. [41]

A convergência do algoritmo ADAM é analisada pelo quadro de aprendizagem online. Considera-se uma sequência arbitrária e desconhecida de funções de custo convexas, designadas por $f_1(\theta)$, $f_2(\theta)$, ..., $f_T(\theta)$. Em cada instante t, o objetivo é prever o parâmetro θ_t e avaliá-lo através de uma função de custo f_t desconhecida até esse momento. [41]

Dado que a sequência de funções é imprevisível, a avaliação do desempenho do algoritmo é realizada através do conceito de "regret" (arrependimento), que corresponde à soma das diferenças entre o valor obtido pela previsão online $f_t(\theta_t)$ e o valor da melhor escolha fixa possível $f_t(\theta^*)$, considerando todos os passos anteriores e um conjunto viável X. Assim, o regret é formalmente definido como:

A soma, ao longo de todos os passos, da diferença entre a função de custo avaliada na previsão do algoritmo e a função de custo avaliada no melhor parâmetro fixo possível. [41]

$$R(T) = \sum_{t=1}^{T} [f_t(\theta_t) - f_t(\theta^*)] \quad (2.\ 42)$$

O optimizador ADAM apresenta um limite para "regret" da ordem de O(√T). Este resultado é comparável ao melhor limite conhecido para um tipo de problema geral de aprendizagem online convexa.

Para simplificar a notação, são introduzidas algumas definições:

- gt representa o gradiente da função de custo no instante t, isto é, gt=∇ft(θt)$gt=\nabla ft(\theta t)$.
- gt,i corresponde ao i-ésimo elemento do gradiente gt.
- Define-se g1:t,i como o vetor que contém a i-ésima dimensão dos gradientes ao longo de todas as iterações até t, ou seja, g1:t,i=[g1,i,g2,i,…,gt,i].
- Adicionalmente, define-se o parâmetro γ como $\gamma = \frac{\beta_1^2}{\sqrt{\beta_2}}$.

O teorema apresentado é válido quando a taxa de aprendizagem αt decresce a um ritmo de t^{-s} e o coeficiente de média móvel do primeiro momento β1,t decai exponencialmente com um parâmetro λ, que tipicamente se situa próximo de 1, por exemplo, $1-10^{-8}$.

Estas definições e condições são fundamentais para garantir que o ADAM mantém propriedades de convergência robustas e limites teóricos de desempenho equiparáveis aos melhores algoritmos de otimização adaptativa conhecidos. [41]

Podemos, com isto, concluir que o ADAM destaca-se por ajustar automaticamente as taxas de aprendizagem de cada parâmetro, o que acelera a convergência e facilita o treino de redes profundas, sendo eficiente em termos de memória e robusto mesmo com valores padrão dos Hiper parâmetros; no entanto, pode generalizar pior do que métodos como o SGD em tarefas que exigem forte regularização, é mais propenso ao sob reajuste em dados pequenos ou ruidosos e, apesar de eficiente, apresenta um custo computacional superior ao SGD puro, além de o seu desempenho poder variar consoante o ajuste dos Hiper parâmetros. [41]

2.2.1.6. Métricas de Avaliação em algoritmos de ML

As métricas de avaliação em algoritmos de ML são indicadores estatísticos, utilizados na avaliação da eficácia e pertinência dos algoritmos, independentemente de estes serem binários, multiclasse ou multilabel, para a resolução do problema em causa tendo em consideração o conjunto de dados disponível. As métricas permitem quantificar numericamente o desempenho do modelo já treinado, no decorrer da validação com dados que não pertenciam ao conjunto definido para treino, o que contribui para verificar se o algoritmo é eficiente ou se existe necessidade de o ajustar, inclusive poder alterá-lo. [35]

De forma geral, a métrica de avaliação pode ser entendida como um instrumento estatístico que é utilizado para avaliar o desempenho das técnicas de classificação e regressão, tanto em contextos binários como multiclasse. Cada métrica analisa aspetos distintos do previsor criado pelo algoritmo de aprendizagem, e a adequabilidade da utilização de cada uma é dependente do problema em questão. A seguir, é apresentado um resumo das métricas principais na avaliação de classificadores em algoritmos de aprendizagem. [35]

- Métricas de Avaliação para problemas de Classificação

- Confusion Matrix

A matriz de Confusão é uma das métricas mais simples e intuitivas utilizadas para avaliar a precisão em modelos de classificação. Existem 4 categorias e a matriz é do tipo 2x2:[35]

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	***True Positive (TP)***	***False Positive (FP)***
	Negative (0)	***False Negative (FN)***	***True Negative (TN)***

Figura 21 - Exemplo gráfico de uma Confusion Matrix

- Precision

A precisão é uma métrica representativa da proporção de previsões corretas entre todos os casos em que o modelo previu de forma positiva. De forma mais explícita, indica o número de previsões positivas pelo modelo que são de facto verdadeiras. É calculada através da divisão entre o número de casos positivos que são de facto verdadeiros pela soma dos verdadeiros positivos e dos falsos positivos, desta forma: [35]

$$\text{Precisão} = \frac{\text{VP}}{\text{VP} + \text{FP}} \tag{2. 43}$$

- Sensivity

A sensibilidade, ou recall, é uma métrica para avaliar a capacidade do modelo em identificar corretamente os casos positivos dentro de uma característica específica. Em termos formais, a métrica é calculada pela razão entre o número de verdadeiros positivos e a soma entre verdadeiros positivos e falsos. [35]

$$S = \frac{TP}{TP + FN} \tag{2. 44}$$

- Accuracy

A acurácia corresponde à percentagem de previsões corretas efetuadas pelo modelo, tendo por base os dados de teste. A métrica é calculada como a razão entre a soma dos valores verdadeiros e positivos e a dos verdadeiros negativos, dividida pelo total de previsões realizadas. [35]

$$A = \frac{TP + TN}{TP + FP + TN + FN} \tag{2. 45}$$

- F1 – Score

O F1-Score é uma métrica que deriva da precisão e da revocação, sendo particularmente útil na avaliação de modelos aplicados a conjunto de dados não equilibrados entre a distribuição das soluções pelas

variáveis de input. A métrica oferece uma medida equilibrada do desempenho do modelo ao combinar os 2 indicadores numa pontuação única. [35]

$$\text{F1 Score} = \frac{P \cdot R}{P + R} \tag{2. 46}$$

- Specificity (SP)

A Specificity é uma métrica que dota o algoritmo com capacidade de identificar corretamente verdadeiros negativos em relação às diferentes categorias existentes. A equação é dada pela expressão: [35]

$$\text{SP} = \frac{TN}{TN + FP} \tag{2. 47}$$

- Error Rate (ER)

O Error Rate é uma métrica representativa da proporção de previsões incorretas em relação ao total de exemplos utilizados do conjunto de dados de teste. A equação é dada pela expressão: [35]

$$ER = \frac{FP + FN}{TP + FP + TN + FN} \tag{2. 48}$$

- Geometric-Mean (GM)

A GM é uma métrica de avaliação que procura maximizar simultaneamente os percentuais de verdadeiros positivos e verdadeiros negativos, ao mesmo tempo que mantém um equilíbrio entre ambas. A equação é dada pela expressão: [35]

$$GM = \sqrt{TP \times TN} \tag{2. 49}$$

- Kappa Statistic (K)

O coeficiente K é uma métrica que avalia o grau de concordância entre as classificações efetuadas pelo algoritmo de aprendizagem e os rótulos reais dos dados, refletindo a precisão do classificador em relação à precisão que é expectável se fosse ao acaso. A equação é dada pela expressão: [35]

$$K = \frac{P_0 - P_e}{1 - P_e} \tag{2. 50}$$

Onde P_0 representa a proporção de concordância observada entre as classificações do modelo e as classes reais e P_e representa a proporção de concordância que é esperada para o acaso. Esta métrica indica o quão melhor é o desempenho do modelo quando comparado a uma classificação aleatória tendo por base a distribuição das classes no conjunto de dados. [35]

- Receiver Operating Characteristic Curve e Area Under Curve

A Receiver Operating Characteristic Curve é um gráfico que representa o desempenho de um algoritmo de classificação em mais do que um limiar. Na curva, são representadas a taxa de verdadeiros positivos (TPR) e a taxa de falsos positivos (FPR) para mais do que valor de corte. Com o aumento do

limiar, existe um incremento no número de dados de validação que são classificados como positivos, o que aumenta tanto o número de verdadeiros positivos como de falsos positivos. [35]

A Area Under Curve corresponde à área abaixo da curva, que oferece uma visualização resumida da capacidade do classificador em distinguir as classes. Um valor próximo de 1 indica um desempenho excelente no fracionamento entre as classes positiva e negativa, enquanto um valor próximo de 0.5 sugere que o classificador terá um desempenho equivalente a uma decisão aleatória ou ao acaso. Os valores intermédios identificam que o modelo consegue identificar corretamente algumas diferenças entre as classes. Esta análise é extremamente útil para problemas de comparação, avaliação, seleção e combinação de mais do que um modelo, e apresenta vantagens como o facto de ser independente do limiar escolhido, sensibilidade a variações e capacidade de indicar o fracionamento entre classes consideradas positivas e negativas. [35]

Existem duas variáveis representadas ao longo da curva, expostas pelas equações seguintes:

True positive rate:

$$\text{TPR} = \frac{TP}{TP + FN} \tag{2. 51}$$

False positive rate:

$$\text{FPR} = \frac{FP}{FP + TN} \tag{2. 52}$$

- Métricas de Avaliação para problemas de Regressão

  - Mean Square Error (MSE)

O erro quadrático médio é uma métrica resultante do cálculo dos quadrados das diferenças entre os valores reais e os valores previstos de um conjunto de dados. A equação é dada pela expressão:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widetilde{Y}_i\right)^2 \tag{2. 53}$$

Onde n assume o valor do número total de registos e o Σ o somatório da diferença entre os valores observados e os previstos em cada instância i de 1 até n. [35]

  - Coefficient of Determination (R^2)

Esta métrica é amplamente utilizada para avaliação de algoritmos de regressão, dado que consegue expressar a proporção da variabilidade da variável dependente a qual se pretende que no pós-treino o modelo tenha capacidade de generalizar. De forma genérica, pode ser interpretada como um valor percentual, que reflete o grau de dependência da variável dependente em relação às variáveis proditoras. [15]

A fórmula é expressa desta forma:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (2.\ 54)$$

- Mean Absolute Percentage Error (MAPE)

Esta métrica avalia o erro médio absoluto entre os valores reais e os valores previstos, expresso em termos percentuais. Por se tratar de uma medida relativa, facilita a comparação do desempenho do modelo nos diferentes conjuntos de dados, independentemente da escala. [15]

A fórmula é expressa desta forma:

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100 \quad (2.\ 55)$$

- Root Mean Squared Error (RMSE)

Esta métrica corresponde à raiz quadrada da média dos erros quadráticos entre os valores previstos pelo modelo e os valores observados. Ao elevar os erros ao quadrado antes de calcular a média, isto penaliza mais os desvios maiores, o que oferece uma perspetiva diferente daquela fornecida pelo erro absoluto médio (MAE).

A fórmula é expressa desta forma:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \quad (2.\ 56)$$

3
DESENVOLVIMENTO DO MODELO DE MACHINE LEARNING

3.1. OBJETIVO DO MODELO DE MACHINE LEARNIG

Como exposto no capítulo 1º da presente Dissertação, o objetivo do trabalho é a construção de um modelo de ML funcional, que permita definir o ponto de arranque de um algoritmo de otimização estrutural. O algoritmo permite o desenvolvimento de soluções para o dimensionamento das coberturas e das fachadas de uma estrutura metálica, com recurso a perfis metálicos normalizados, e está subdivido em 2 algoritmos distintos e interligados.

Existem 2 possibilidades na escolha dos perfis a aplicar que diferem entre si em parâmetros específicos, a serem apresentados no seguimento deste capítulo.

Um dos objetivos primordiais do simulador de cálculo é o dimensionamento e execução de Coberturas e Fachadas (Horizontais ou Verticais), sendo que as soluções que oferece garantem aplicabilidade em qualquer tipo de estrutura (Aço, Madeira ou Betão), e consistem na colocação do sistema como apoio do revestimento final da cobertura ou da fachada. A grande vantagem da utilização dos perfis é a possibilidade de obter grandes vãos e maiores afastamentos entre madres, reduzindo assim o seu número. O modelo de cálculo adotado para o desenvolvimento das soluções baseia-se num modelo de viga contínua, que é assegurada através da sua capacidade de sobreposição para reforço dos picos de momentos negativos e positivos, fator que permite aumentar a sua resistência global. A utilização dos perfis verticais permite, automaticamente e sem custos acessórios, a execução de platibandas.

O modelo, construído para apresentação como conclusão da presente Dissertação, servirá como ponto de partida, que permitirá acelerar o processo de obtenção da estrutura (s0) e que entrará no simulador posteriormente para obtenção da estrutura otimizada para o dimensionamento das coberturas e fachadas. A Rede Neuronal, depois de construída será treinada seguindo os princípios da aprendizagem supervisionada, tendo por base os dados obtidos do simulador, estratificados entre valores de entrada e soluções calculadas a partir desses dados. Posteriormente, o objetivo será alcançar um modelo funcional capaz de generalizar soluções com dados de entrada novos, desconhecidos na fase de treino e validação, que garanta uma solução inicial muito próxima da solução que o algoritmo de otimização obterá, com uma diminuição drástica do tempo de cálculo da solução que posteriormente irá ser otimizada pelo algoritmo de otimização desenvolvido.

A utilização do modelo de ML em detrimento da utilização repetitiva do simulador cálculo garantirá, entre outras coisas, maior velocidade no processo de cálculo e menor exigência do ponto de vista computacional. Neste tópico, se o modelo de ML for funcional e garantir resultados praticamente similares aos do algoritmo com erros insignificantes, a sua integração no simulador permitirá poupanças

do ponto de vista financeiro muito significativas, porque permitirá reduzir de forma substancial a utilização de CPU (Unidade Central de Processamento), comumente paga por utilização.

3.1.1. Apresentação e Explicitação das Características do Perfil Tipo Madre

Os perfis que existem dentro da Base de Dados foram concebidos com o propósito de criar uma solução construtiva que consumisse menos materiais e reduzisse a necessidade de mão de obra, ao mesmo tempo que torna o processo de montagem mais ágil e prático, superando os obstáculos típicos das obras convencionais.

As implementações de perfis permitem do ponto de vista do dimensionamento estrutural:

1) Versatilidade ampliada nas soluções estruturais possíveis, permitindo adequar o projeto a diferentes necessidades.
2) Redução do peso das estruturas projetadas, facilitando o manuseio, a execução e diminuindo cargas.
3) Facilidade de conexão direta a pilares e vigas treliçadas, graças a furações apropriadas para fixações, tornando as ligações mais simples e eficientes.
4) Diminuição dos custos relacionados a materiais, mão de obra, processos fabris e montagem, tornando o projeto mais económico.
5) Maior agilidade no processo de instalação, otimizando o tempo de execução da obra.
6) Eliminação da necessidade de componentes estruturais adicionais, como tirantes, patilhas, vigas treliçadas em fachadas ou chapas de cumeeira interna.
7) Aplicação do cálculo estrutural baseado em viga contínua, o que distribui melhor os esforços na estrutura e pode otimizar o dimensionamento.
8) Possibilidade de vencer vãos maiores e permitir maiores espaçamentos entre as madres, resultando em estruturas mais enxutas e económicas.

Do ponto de vista estrutural, as descrições relevantes para a correta compreensão das características estruturais do perfil Madre são as seguintes:

- Trata-se de um perfil fabricado em aço S350GD + Z275, conforme a norma EN 10346.
- Desenvolvido para uso em sistemas construtivos de aço, betão ou madeira.
- Possui como destaque a capacidade de sobreposição por encaixe sequencial, facilitando a montagem modular das peças.
- A estabilidade e a rigidez estrutural são obtidas através de múltiplos pontos de furação distribuídos ao longo do perfil, possibilitando ajustes de posição conforme a necessidade do projeto.
- Permite o cruzamento entre perfis, sejam eles de secção idêntica ou diferentes.
- Disponível em nove opções de secções: 50x1.5, 100x1.5, 150x1.5, 200x1.5, 200x2.0, 250x2.0, 250x2.5, 300x2.5 e 300x3.0.
- Os perfis são providos de orifícios tanto nas almas quanto nos banzos inferiores, compatíveis com parafusos M10.

- Estes orifícios estão distribuídos ao longo de toda a extensão do perfil, com espaçamento regular de 50 mm entre eles.

É importante salientar que, contudo, é aconselhável a utilização de parafusos para aparafusar nos perfis da Classe de Resistência 8.8.

A identificação do tipo de secções disponíveis para cada perfil é descrita pelo valor de altura da secção e da espessura da chapa. A título de exemplo, a secção 50x1.5 é uma secção com 50 mm de altura de secção e 1.5 mm de espessura da chapa.

São ainda oferecidas 4 (quatro) possibilidades de variações geométricas de aplicabilidade do perfil selecionado em estruturas reais.

Geometria Perfil 1

- Forma geral: Perfil trapezoidal com base menor no topo.
- Dimensões marcadas:
    - b1: largura da base superior (no topo, parte mais estreita).
    - y1: altura vertical do segmento central.
    - t: espessura da parede do perfil.
    - b3: largura total da base inferior (parte mais larga).

Geometria Perfil 2

- Forma geral: Perfil similar ao primeiro, com o topo aberto e abas laterais inclinadas.
- Dimensões marcadas:
    - H: altura total do perfil.
    - t: espessura da parede.
    - y1: altura parcial do segmento central.
    - b2: largura entre as extremidades externas das abas.
    - r1: raio das curvaturas laterais.

Geometria Perfil 3

- Forma geral: Perfil simétrico com projeções laterais, topo estreito (b1), corpo central, e extremidades largas (b3).
- Dimensões marcadas:
    - b1: largura do topo.
    - y1: altura central.
    - t: espessura da parede.
    - b3: largura total da base.

Geometria Perfil 4

- Forma geral: Perfil simétrico com cintura estreita e abas laterais, topo e base arredondados.
- Dimensões marcadas:
    - H: altura total.
    - b2: largura máxima do perfil.
    - y1: altura parcial do segmento central.
    - t: espessura da parede.
    - b1: largura do topo.

Onde b1 assume o valor da largura da parte superior da secção, b2 a largura intermédia da secção, b3 a largura total da base do perfil, h a altura total do perfil e t a espessura da chapa de aço utilizada para a constituição do perfil.

A tabela que apresenta as características relevantes para o dimensionamento estrutural de cada secção disponível do perfil em causa não pode estar publicada, mas é relevante a sua leitura para percecionar as características que definem o comportamento estrutural dos perfis em causa.

No catálogo, são apresentadas as disposições possíveis para cada secção, as dimensões afetas a cada disposição, a massa, a área de desenvolvimento (comprimento total da chapa de aço em mm de todas as faces e dobras da secção do perfil), as características mecânicas e a fixação (subdivida em 2 colunas, indica o diâmetro do furo existente no perfil com a respetiva distância entre a face exterior e o centro do furo, e o tipo de parafuso utilizado para a fixação dos perfis à estrutura principal).

É de grande relevância salientar que as propriedades apresentadas no catálogo dizem respeito à secção bruta e que não são considerados efeitos de não linearidade geométrica do material, bem como imperfeições de execução.

Os perfis do tipo Madre são perfis metálicos produzidos por enformação a frio (cold-formed), ou seja, são obtidos por conformação mecânica a temperatura ambiente, são compostos por aço estrutural S350GD com galvanização Z275 segundo EN 10346 e estão disponíveis em diversas seções, comercializadas em várias larguras e espessuras padronizadas, otimizadas para utilização como madres, vigas secundárias e elementos estruturais leves em coberturas e fachadas. São sistemas modulares, facilmente aplicáveis e aparafusados diretamente a pilares ou vigas.

Perfis enformados a frio, como os do tipo Madre, devido à sua esbelteza (paredes relativamente finas face às dimensões globais), são classificados estruturalmente como seções de Classe 4 segundo o EC-3.

Secções de classe 4 são seções "finas", suscetíveis a fenómenos de encurvadura local e perdas de eficácia resistentes por instabilidade local. Devem ser tidas em conta reduções de capacidade resistente nas verificações estruturais. Este tipo de secções requer uma análise rigorosa dos riscos de encurvadura e cálculo de resistência eficaz (effective width method), conforme a norma.

A utilização mais indicada é para cargas de magnitude intermédia, vãos entre 15 e 30 m, e situações em que o peso reduzido e a facilidade de construção sejam fatores diferenciadores. Perfis enformados a frio, especialmente de Classe 4, não são recomendados como elementos principais em estruturas sujeitas a grandes esforços concentrados, sem prévio dimensionamento e validação rigorosa do projeto.

3.1.2. Apresentação e Explicitação das Características do Perfil tipo Omega

O perfil do tipo Omega constitui uma opção eficiente de perfis metálicos para sistemas construtivos, destacando-se pelo seu custo acessível e versatilidade, o que proporciona ganhos de tempo e redução de recursos nas obras. Diferente do perfil do tipo Madre, o Omega apresenta almas lisas, sem perfurações, encontra-se disponível, no simulador, exclusivamente em quatro configurações distintas, todas com espessura de 1,5 mm, e é fabricado em aço S220GD.

No contexto técnico, são agora apresentadas informações essenciais para a compreensão detalhada das propriedades estruturais do perfil tipo Omega.

- Construção em aço S220GD com galvanização Z200, proporciona robustez estrutural e proteção anticorrosiva.
- Montagem direta em pilares e vigas através de aparafusamento, permitindo uma fixação prática e eficiente na estrutura.
- Orifícios uniformemente posicionados ao longo do comprimento, com espaçamento de 50 mm, facilitando várias opções de ligação.
- Permite a sobreposição entre perfis, possibilitando soluções construtivas ajustáveis conforme a necessidade do projeto.
- Aberturas de 12x14 mm, concebidas para receber parafusos M10 de classe 8.8, garantindo uniões seguras e resistentes.

A identificação do tipo de secções disponíveis para cada perfil é descrita pelo valor de altura da secção e da espessura da chapa. A título de exemplo, a secção 50x1.5 é uma secção com 50 mm de altura de secção e 1.5 mm de espessura da chapa.

De presente, a solução com perfil do tipo Omega não oferece variabilidade geométrica, tornando-se, por isso, não possível a sua apresentação. A aplicabilidade de cada secção do Perfil é realizada de forma única, ou seja, não existe no processo de dimensionamento a possibilidade da junção de 2 secções como uma solução estrutural em coberturas e fachadas.

A tabela que apresenta as características relevantes para o dimensionamento estrutural de cada secção disponível do perfil em causa não pode estar publicada, mas é relevante a sua leitura para percecionar as características que definem o comportamento estrutural dos perfis em causa.

No catálogo são apresentadas as disposições possíveis para cada secção, as dimensões afetas a cada disposição, a massa, a área de desenvolvimento (comprimento total da chapa de aço em mm de todas as faces e dobras da secção do perfil), as características mecânicas e a fixação (subdivida em 2 colunas, indica o diâmetro do furo existente no perfil com a respetiva distancia entre a face exterior e o centro do furo, e o tipo de parafuso utilizado para a fixação dos perfis à estrutura principal).

É de grande relevância salientar que as propriedades apresentadas no catálogo dizem respeito à secção bruta e que não são considerados efeitos de não linearidade geométrica do material, bem como imperfeições de execução.

Os perfis do tipo Omega são elementos metálicos obtidos por enformação a frio (cold-formed), isto é, fabricados através de conformação mecânica a temperatura ambiente. São constituídos em aço estrutural S220GD com galvanização Z200, cumprindo os requisitos da norma EN 10346. Estes perfis estão

disponíveis em quatro secções específicas, todas com espessura padrão de 1,5 mm, adequadas para aplicações recorrentes como elementos leves de suporte em coberturas e fachadas.

A geometria dos perfis Omega visa soluções construtivas funcionais, e permite rapidez na montagem, já que estes podem ser fixados diretamente a vigas ou pilares. O sistema modular simplifica os processos em obra e permite integrar estes perfis em diversos esquemas estruturais de forma eficiente, sobretudo quando se pretende racionalizar o consumo de materiais e o tempo de construção.

Do ponto de vista normativo, os perfis Omega, assim como outros perfis enformados a frio, apresentam paredes delgadas em relação às suas dimensões globais. Por esta razão, enquadram-se estruturalmente como seções de Classe 4 segundo o EC-3. Este facto implica que estas secções sejam mais propicias a sofrer fenómenos de encurvadura local e perda de resistência por fenómenos de instabilidade na parede, obrigando à aplicação do método da largura efetiva (effective width method) durante o dimensionamento, para considerar a real capacidade resistente da secção.

A utilização mais indicada é para cargas de magnitude intermédia, vãos entre 15 e 30 m, e situações em que o peso reduzido e a facilidade de construção sejam fatores diferenciadores. Perfis enformados a frio, especialmente de Classe 4, não são recomendados como elementos principais em estruturas sujeitas a grandes esforços concentrados, sem prévio dimensionamento e validação rigorosa do projeto.

A nível construtivo, os perfis Omega não possuem furações nas almas e distinguem-se, portanto, pela ausência de furos para ligação, por oposição aos perfis do tipo Madre. Isto condiciona a configuração da ligação e pode limitar opções de montagem, mas favorece a integridade estrutural da alma. São, por isso, amplamente utilizados como elementos secundários em estruturas de aço, madeira ou betão, especialmente em suportes para painéis e revestimentos de coberturas e fachadas, onde o peso da estrutura e a rapidez de construção desempenham papel relevante.

3.1.3. Explicação do Método de Operacionalidade do Simulador de Cálculo Estrutural

O simulador de cálculo estrutural é uma plataforma de cálculo estrutural, que permite dimensionar e otimizar elementos estruturais para coberturas e fachadas de uma estrutura metálica a ser projetada e edificada.

Considerando o algoritmo por inteiro, este permite ao utilizador, perante o input de uma série de variáveis a serem apresentadas no seguimento deste subcapítulo, definir o tipo de cargas a que a estrutura irá estar sujeita, bem como a definição geométrica da estrutura a ser projetada. Posteriormente, oferece ao utilizador do simulador uma variedade de soluções já otimizadas para todos as secções existentes, dentro dos 2 perfis possíveis, que cumpram os regulamentos inscritos no EC-3, indicando aquela que mais próxima se encontra da solução ideal.

O utilizador terá com isto a possibilidade de escolher, de entre as várias soluções possíveis, aquela que prefere para solucionar o problema caracterizado de forma completa pelo conjunto estruturado dos dados de entrada. A identificação da solução que melhor se aproxima da solução ideal não é vinculativa, o que significa que o utilizador tem a opção de selecionar uma outra solução também possível que melhor se enquadre nos princípios prioritários que define, por exemplo, a facilidade de aplicabilidade da solução em contexto real de obra que não obrigue a contratar uma equipa maior ou mais especializada.

Os dados de entrada são processados diretamente pelo simulador de cálculo, que calcula diretamente as soluções finais possíveis já otimizadas. O simulador integra todos os cálculos necessários num único processo computacional, que gera automaticamente as soluções possíveis já otimizadas, o que significa que, além de cumprir os regulamentos normativos, não apresenta um custo encarecido para o seu desenvolvimento em fábrica. Para a definição geométrica e o tipo de cargas a que a estrutura irá estar sujeita, o utilizador deverá indicar o valor das seguintes variáveis de input:

3.1.3.1. Variáveis de Input necessárias para o desenvolvimento do cálculo dentro do Simulador

- Type

Tipo de Perfil ou solução estrutural a considerar (ex: continuidade natural, reforço duplo, etc.), permite que o utilizador selecione inicialmente o tipo de perfil que prefere utilizar. Como descrito no subcapítulo (3.1.1), o perfil do tipo Madre permite que o desenvolvimento do cálculo estrutural siga o princípio do modelo de viga contínua. Na consideração inicial das variáveis de entrada, a seleção de "type" 1 assegura a escolha de um perfil com continuidade natural tipo Madre. Quando selecionado o "type" 1 como variável de input, o utilizador está a dar preferência a soluções com este tipo de perfil que sigam o desenvolvimento de cálculo pelo princípio do modelo de viga contínua.

A seleção de "type" 2 como variável de input significa que o utilizador está a dar preferência a soluções do tipo perfil com reforço duplo nos apoios/vãos, perfil do tipo Omega.

A seleção correta do tipo de perfil atua como ponto de partida parametrizado de preferência do utilizador. No entanto, para a solução fornecida pelo algoritmo 1, esta seleção não é decisória. O simulador utiliza uma matriz decisória multicritério (a ser apresentada posteriormente) que se sobrepõe à seleção inicial se identificar ganhos significativos de desempenho.

A seleção do "Type" serve essencialmente para operar na definição da geometria inicial e nas restrições indiretas a serem priorizadas no cálculo (type:1 prioriza verificações de flexão 6.3.2.d do EC-3, enquanto que o type:2 ativa checks de estabilidade local 6.3.3.a do EC-3).

A Matriz Decisória para a variável type segue um algoritmo de seleção deste tipo:

```
graph TD
    A[Input type] --> B{Geração de Soluções}
    B --> C1[Tipo 1: Madre]
    B --> C2[Tipo 2: Omega]
    B --> C3[Outros Tipos]
    C1 --> D[Análise de utilizationRatio]
    C2 --> D
    C3 --> D
    D --> E{utilizationRatio < 0.95?}
    E -->|Sim| F[Comparação de Peso Total]
    E -->|Não| G[Descarta Solução]
```

F --> H[Seleciona Perfil com Menor Peso]

Em que os parâmetros de decisão seguem os critérios de utilizationRatio (Rácio de utilização, é a divisão entre o esforço atuante pelo esforço resistente) que detêm um peso de 60% na decisão final; o Peso Total da solução de perfil selecionado, em kg/m, que influencia de forma direta o valor do preço de execução dos perfis, detêm um peso de 25%, e a Conformidade normativa, que rejeita soluções em que a estabilidade local em elementos submetidos a fenómenos de flexão esteja comprometida pelo não cumprimento da norma estabelecida no EC-3, detêm o peso restante.

Para cenários de vãos curtos (< 4 m) e de cargas uniformes com cargas distribuídas inferiores a 1 kN/m^2 o algoritmo favorece a continuidade estrutural, e por isso cenários do Type 1.

Por oposição, em cenários com vãos significativos (> 6 m) e com cargas dinâmicas superiores a 1.4 kN/m^2, é mandatária a existência de reforços locais e, por consequência, a escolha de soluções de Type 2 são preferidas.

Por último, o algoritmo dispõe de um limiar de eficiência, sendo que, cajo os pressupostos anteriores sejam todos cumpridos, se a diferença no utilizationRatio for superior a 5%, ativa a troca automática para fazer sair a solução mais próxima da otimização.

$$\Delta_{UR} = \frac{UR_{tipo1} - UR_{tipo2}}{UR_{tipo1}} \geq 0{,}05 \quad (3.\ 1)$$

- Country

A variável "country" do simulador tem impacto crítico no dimensionamento estrutural, ao ativar uma sequência de parâmetros técnicos e normativos específicos em cada país, presentes nos anexos nacionais anexados ao EC-3.

A Matriz Decisória para a variável "Country" segue um algoritmo de seleção deste tipo:

country: Portugal

├→ Normas: NP EN + RSA

├→ Materiais: S350GD/S220GD

├→ γ_M0=1.0, γ_M1=1.10

└→ Ações: Vento RSA/Zonas, Ψ_2=0.0 para neve<1000m

- Location

A variável "location" de input do simulador desempenha um papel crítico na definição das condições ambientais e normativas de projeto, influenciando de forma direta o dimensionamento estrutural. A seleção adequada é essencial para garantir conformidade com regulamentos locais e precisão nos cálculos.

O valor da "location" influencia de forma direta, os valores da carga do vento a considerar no projeto, a intensidade e distribuição do valor da carga da neve, e os coeficientes Ψ de combinação em ações variáveis. A variável "location" está ainda vinculada a requisitos específicos normativos e de especificações de materiais (Aço S350GD é preferido em zonas costeiras enquanto o S220GD é preferível em áreas urbanas), para a seleção da solução que melhor responde ao problema constituído pelo somatório das influências que resultam da seleção das variáveis de entrada.

Pode assumir 3 valores em Portugal (depende das restrições enunciadas nos anexos do EC-3 em cada país):

Valor 1 - Zona Costeira (Rugosidade III) em que a velocidade do vento a considerar em Projeto é de 27 a 30 (m/s).

Valor 2 - Urbana (Rugosidade IV) em que a velocidade do vento a considerar em Projeto é de 24 a 26 (m/s).

Valor 3- Rural (Rugosidade II) em que a velocidade do vento a considerar em Projeto é de 30 a 33 (m/s)

A "location" é também decisiva ao, por exemplo, exigir perfis do tipo Omega em projetos com comprimentos de vãos muito significativos, em que a "location" assume valor 1, devido a existência de pressões dinâmicas maiores, ou ao permitir perfis do tipo Madre com utilizationRatio ≤ 0.6 se "location" assumir o valor de 2.

Se "location" assumir valor de 1 (Zona Costeira) exige tratamentos anticorrosivos adicionais.

A Matriz Decisória para a variável "location" segue um algoritmo de seleção deste tipo:

graph TD

A[Identificar coordenadas geográficas] --> B{É zona costeira?}

B -->|Sim| C[location=1]

B -->|Não| D{Área urbana densa?}

D -->|Sim| E[location=2]

D -->|Não| F[location=3]

C --> G[Verificar rugosidade do terreno]

E --> H[Ajustar combinationFactors]

F --> I[Considerar efeitos topográficos]

- Lang

"lang" é a variável que define a linguagem com que será apresentada as soluções. Não apresenta relevância para o cálculo estrutural, trata-se apenas do idioma com que a solução será apresentada e descrita.

- Direction

A variável "direction" desempenha um papel crítico na definição da orientação das cargas e na configuração geométrica dos perfis metálicos. A variável "direction" atua como um parâmetro de controle espacial que influencia a orientação das cargas (vento, permanente, neve) a disposição dos perfis em relação aos eixos estruturais e a aplicação das verificações normativas. A direction pode assumir 2 valores:

Valor 1: Análise longitudinal.

Valor 2: Análise transversal.

- Format

A variável "format" configura-se como a variável que garante a adequação técnico-normativa em projetos. A definição como 2 garante a conformidade com as práticas europeias da engenharia estrutural, desde a correta apresentação de unidades até a hierarquização das verificações de segurança segundo o EC-3. No fundo assegura que tanto as variáveis de input como os valores calculados que decorrem da execução do algoritmo estão no formato correto em que são apresentadas no EC-3, e hierarquiza as verificações necessárias a realizar.

- Inclination

A variável "inclination" configura-se como parâmetro crítico para a modelação geométrica de coberturas inclinadas, influenciando diretamente o comportamento estrutural dos perfis metálicos normalizados. A variável representa o ângulo de inclinação da cobertura em graus, com implicações diretas na distribuição de cargas (vento, neve, permanente), na geometria da aplicação dos esforços e ainda na configuração dos apoios e das ligações estruturais. A título exemplificativo, a inclinação é um dos parâmetros para a definição da esbelteza reduzida, λLT, calculada com fatores de correção específicos do EC-3.

A inclinação condiciona ainda de forma normativa o valor a considerar para a acumulação de neve (coeficiente de forma μ1), a pressão dinâmica do vento (coeficiente de exposição ce) e as cargas acidentais de manutenção, essenciais para o correto dimensionamento estrutural dos perfis metálicos a utilizar nas Coberturas e Fachadas de uma estrutura metálica.

- ModelSpans

A variável "modelSpans" configura-se como um dos parâmetros fundamentais na definição geométrica dos vãos onde vão ser aplicados os perfis normalizados disponíveis. A sua configuração influencia diretamente a distribuição de esforços, seleção de perfis e verificações normativas.

A variável representa a distribuição de comprimentos de vãos em metros, organizados sequencialmente para a modelação precisa da estrutura. A combinação com "transversalDistance" (distância entre perfis) e "supportWidth" (largura de apoios) a serem explicitas posteriormente, define a malha estrutural completa. Esta relação determina a eficiência do sistema através do índice:

$$\text{Eficiência} = \frac{\sum \text{modelSpans}}{\text{transversalDistance} \times \text{supportWidth}} \tag{3. 2}$$

A configuração dos vãos afeta ainda, diretamente, os momentos fletores máximos, permite ao algoritmo verificar a existência de vãos com comprimentos significativos, que exigem perfis de maior inércia (série 300x3 ou 200x2) para manter a utilizationRatio abaixo de 1.0, e limitam o valor dos vãos máximos para estruturas não contraventadas, limitados a 6 m, e o valor do comprimento da consola, dado que é exposto no EC-3, sendo que o máximo comprimento possível será ¼ do valor do vão adjacente.

- hasCantilever

A variável "hasCantilever", que pode assumir valor verdadeiro ou falso, define a existência ou não de uma consola dentro da variável "modelSpans". É relevante na definição estrutural que implicará todo o processo de cálculo desenvolvido pelo algoritmo.

- transversalDistance

A variável "transversalDistance" representa a distância entre eixos dos perfis paralelos na direção transversal, em metros. Influencia de forma decisiva a distribuição de cargas entre os elementos estruturais, a seleção económica dos perfis, e o cumprimento das normas expostas no EC-3.

A distância transversal, afeta de forma direta o cálculo da carga por metro linear, e o valor deve estar concordante com o limite máximo de 3.0 m para estruturas não contraventadas, se for esse o caso. Em distâncias transversais inferiores a 2.5 m, é aplicada uma taxa de redução de 15% na carga do vento a considerar para projeto.

- supportWidth

A variável "supportWidth" representa a largura em metros dos apoios aos quais os perfis metálicos, que irão constituir as coberturas e as fachadas, estarão conectados. A largura do apoio afeta de forma direta a distribuição dos momentos fletores, e condiciona as verificações do Esmagamento da alma (6.2.6.3), da Encurvadura por esforço transverso (6.2.6.6) e da interação momento-corte (6.2.8) presentes no EC-3.

O valor desta variável irá ainda ser definidora na avaliação dos princípios normativos que exigem que a largura do apoio terá de assumir o valor de 1/20 do vão adjacente, ao qual se acrescenta um acréscimo de 15% se o apoio for em alvenaria.

Para valores de "supportWidth" = 1 a ligação é considerada como ligação rotulada. Por oposição, para valores de supportWidth ≥ 2, a ligação é considerada como semi-rígida. Para valores intermédios entre 1 e 2, o algoritmo detêm um mecanismo de interpolação normativa, que aplica a verificação combinada dos artigos 6.3.2 e 6.4.2 presentes no EC-3 , garante a utilização de um fator de segurança adicional de 1.1 para momentos negativos e limita a rotação máxima a 25 mrad.

- Loads

A variável "loads" assume os valores de todas as cargas que vão atuar sobre a estrutura durante o tempo de vida útil projetado. No algoritmo, está definido como um array que contém os objetos definidores das cargas de projeto.

"loads": [

{"case":"permanent","type":"distributed","value":[0.1,…,0.1]},

{"case":"wind","type":"pressure","value":[[0.96,12],[0.96,0]]},

{"case":"wind","type":"suction","value":[[0,12],[0,0]]}

]

"Case" assume um identificador que caracteriza a componente de carga,como permanente, resultante dos efeitos do vento ou da neve; "Type" assume a natureza física da carga, se é distribuída, se é de pressão, de sucção, concentrada etc.., e "value" a magnitude em kN/m ou kN/m^2.

- Deformation

A variável "deformation" assume o valor da flecha máxima admissível em milímetros descrita no Anexo Nacional Português do EC -3. É essencial a sua correta definição, para verificar se uma secção de um perfil pode ou não, ser utilizada como solução numa cobertura e/ou fachada.

- spanRestraints

A variável de "spanRestrains" define o tipo de restrições no apoio a que viga estará sujeita. Pode assumir 3 valores, -1 se o Engaste for completo, e com isto bloquear translações e rotações, 0 se a restrição for livre, e por isso permitir rotações e translações, e 1 se o tipo de restrição for um apoio simples parcial que restringe translações, mas permite rotações.

É essencial, por exemplo, para o cálculo dos momentos fletores e da flecha máxima (neste caso o fator k de restrição é diretamente dependente do tipo de restrições no apoio).

- combinationFactors

A variável "combinationFactors" assume os valores dos coeficientes de redução probabilística a serem utilizados para cálculo das combinações das cargas que irão atuar sobre a estrutura. São normativos e estão definidos na NP EN 1990 (Eurocódigo 0)

- internal

A variável "internal" é uma variável de entrada complexa, que pode ser considerada como um modelo avançado de parametrização que controla aspetos específicos da análise estrutural, e que funciona como configurações internas do sistema de cálculo. Esta variável engloba diferentes categorias de informações, que não são carregamentos externos diretos, mas sim condições de análise e carregamentos adicionais processados internamente pelo simulador.

A estrutura típica da variável "internal" inclui várias secções especializadas. A secção "loads" dentro de "internal" contém carregamentos distribuídos permanentes, com controles como "isMinMaxModeOn", carregamentos pontuais com valores em vãos específicos, e carregamentos axiais de direção definida. Adicionalmente, a variável inclui opções de verdadeiro ou falso para diferentes condições como "windLocal", "snowAccumulation", "snowExceptional" e "isWindLoadsByBand".

A principal diferença entre as variáveis "internal" e "loads" reside na natureza e origem dos carregamentos. A variável "loads" define carregamentos externos aplicados diretamente à estrutura, incluindo cargas permanentes distribuídas, pressões e sucções de vento com valores específicos para cada caso. Por contraste, a variável "internal" trabalha com carregamentos que são processados ou calculados internamente pelo algoritmo, como carregamentos pontuais adicionais com valores nulos por padrão, carregamentos axiais controlados e configurações especiais para análises do vento local ou acumulação de neve nas estruturas.

A título exemplificativo: enquanto a variável "loads" pode definir um carregamento do vento de valor 0.96 kN/m² aplicado sobre um comprimento de 12 metros, a variável "internal" pode ter "windLocal" definido como verdadeiro, indicando que análises locais de vento devem ser conside-

radas adicionalmente. Esta configuração interna não altera o carregamento base, mas instrui o sistema a realizar verificações adicionais.

Podemos, com isto, então definir, de forma geral, que a variável "internal" se organiza em subparâmetros que gerenciam desde carregamentos não convencionais até flags de ativação de análises especializadas. A arquitetura da variável no simulador inclui:

- "loads": Configurações para carregamentos distribuídos, pontuais e axiais com modos específicos (ex: "isMinMaxModeOn").
- Flags de ativação: "windLocal", "snowAccumulation", "snowExceptional", e "isWindLoadsByBand", que habilitam verificações adicionais conforme as exigências normativas.
- Parâmetros de restrição: Como "spanRestraints", que interage com definições de fibras superiores/inferiores para a análise de estabilidade da estrutura, com foco especial nas ligações dos perfis aos apoios.

Para a correta perceção da pertinência da existência desta variável dentro do simulador, enquanto variável de input, vão agora ser apresentados alguns casos tipo, em que a variável "internal" assume papel decisivo no correto dimensionamento estrutural dos perfis a utilizar em Coberturas e Fachadas

- Caso 1 – Situações críticas originadas por Carregamentos do tipo Cargas Distribuídas

No exemplo apresentado abaixo, a subvariável "distributed" dentro de "internal.loads" contém o parâmetro isMinMaxModeOn, que determina se o sistema considera valores mínimos e máximos para cargas permanentes:

```
"internal": {
  "loads": {
    "distributed": {
      "permanent": {"isMinMaxModeOn": false}
    }
  }
}
```

Quando ativado (true), o simulador calcula envolventes de esforços para cenários extremos, o que é essencial em projetos que exigem valores mais conservadores, que lhe permitam estar do lado da segurança. Em contraste, a variável independente "loads", referida anteriormente, define apenas magnitudes fixas, sem flexibilidade analítica.

- Caso 2 – Análise em situações específicas de Vento localizado

A subvariável "windLocal" é também um exemplo da forma como a variável "internal" introduz condições específicas. No algoritmo, a ativação da subvariável (true) instrui o sistema a aplicar pressões de vento localizadas conforme a NP EN 1991-1-4, complementares aos carregamentos globais definidos em "loads":

```
"internal": {
  "windLocal": true
```

}

Esta ativação é implicada na geração automática de coeficientes de forma para elementos secundários (ex: telhas), aprimorando a distribuição de esforços em áreas críticas. Sem esta ativação, o cálculo global iria ignorar os micro-efeitos aerodinâmicos, subestimando solicitações em componentes localizados.

- Caso 3 – Projetos em altitude que exigem a consideração de valores excecionais para acumulação de neve

Projetos em regiões montanhosas, ou seja, em altitude, requerem frequentemente a consideração de "snowAccumulation" e "snowExceptional", sub-variávies que modificam fatores de combinação e durações de carga. Num projeto de altitude elevada, a ativação simultânea desses flags tende a aumentar significativamente os momentos fletores nos apoios devido à redistribuição não linear dos esforços. A "internal" aqui atua como um interruptor normativo, integrando disposições da NP EN 1991-1-3 (EC-3 com o anexo nacional Português) que não são consideradas por carregamentos estáticos convencionais.

- Caso 4 – Cenário de não linearidade geométrica

Em cenários de não-linearidade geométrica, a "internal" ativa as sub-variáveis "axial value" e "axial direction", que controlam cargas axiais incrementais durante o processo iterativo. Num arco com 24 m de vão, a definição de "axial value" como 0.3 direciona o algoritmo Newton-Raphson a considerar os efeitos P-Δ progressivos, que evita erros de convergência comuns em modelos simplificados.

É assim possível perceber, tanto pela definição como pelos casos demonstrativos, como a variável "internal" opera dentro do simulador.

No final, a configuração numérica ou de formato de todas as variáveis de input permitem ao simulador processar os dados de entrada e desenvolver o processo de cálculo, através da combinação dos parâmetros de entrada, resultando com isso em mais do que uma solução para o dimensionamento das coberturas e/ou fachadas da estrutura metálica a ser projetada.

A título exemplificativo, as variáveis finais formam um conjunto estruturado dos dados de entrada que caracteriza de modo completo o cenário a ser analisado pelo simulador de cálculo estrutural.

Type: 1

location: 1

country: Portugal

lang: pt

direction: 2

format: 2

inclination: 5

modelSpans: [0.3, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 0.3]

hasCantilever: True

transversalDistance: 2

supportWidth: 1

loads: [{‘case’: ‘permanent’, ‘type’: ‘distributed’, ‘value’: [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]}, {‘case’: ‘wind’, ‘type’: ‘pressure’, ‘value’: [[0.96, 12], [0.96, 0], [0.96, 0]]}, {‘case’: ‘wind’, ‘type’: ‘suction’, ‘value’: [[0, 12], [0, 0], [0, 0]]}]

deformation: {‘limit’: 200, ‘combination’: 1}

spanRestraints: {‘upperFibre’: [-1, -1, -1, -1, -1, -1], ‘lowerFibre’: [0, 0, 0, 0, 0, 0]}

combinationFactors: {‘live’: {‘id’: 1, ‘value’: [0, 0, 0]}, ‘wind’: {‘id’: 1, ‘value’: [0.6, 0.2, 0]}, ‘snow’: {‘id’: 1, ‘value’: [0.5, 0.2, 0]}}

internal: {‘loads’: {‘distributed’: {‘permanent’: {‘isMinMaxModeOn’: False}}, ‘points’: {‘permanent’: [{‘value’: [0, 0, 0, 0], ‘span’: [1, 2, 3, 4, 5, 6, 7, 8]}], ‘live’: [{‘value’: [0, 0, 0, 0], ‘span’: [1, 2, 3, 4, 5, 6, 7, 8]}]}, ‘axial’: {‘value’: 0, ‘direction’: 1}, ‘windLocal’: False, ‘snowAccumulation’: False, ‘snowExceptional’: False, ‘isWindLoadsByBand’: False}, ‘axial’: False, ‘spanRestraints’: True}

A escolha final do perfil depende da otimização entre peso, custo e desempenho estrutural e cada solução será de um perfil do tipo Madre e/ou Omega, com secção diferente desde que cumpra os regulamentos necessários impostos ao dimensionamento estrutural do elemento construtivo. A escolha do perfil a ser otimizado segue a lógica enunciada no subcapítulo anterior (3.1.3).

3.1.3.2. Variáveis de output que constituem a solução do perfil calculado pelo Simulador

Como explicitado nos subcapítulos anteriores, o algoritmo faz sair uma série de soluções que respondem com sucesso ao problema constituído pela combinação das variáveis de entrada. As variáveis de saída são as seguintes:

- Type

A variável de saída “type” identifica o tipo de perfil selecionado para a solução em causa. Como explicado anteriormente, se “type” = 1 o perfil é do tipo Madre, se “type” = 2 o perfil é do tipo Omega.

- Tubular

A variável “tubular” é um campo de ativação presente no output da solução, que identifica se o perfil estrutural proposto apresenta uma secção transversal fechada (tubular) ou aberta, sendo isto referente às variações geométricas que o perfil do tipo Madre e Omega podem assumir. Como explicitado nos subcapítulos que servem de introdução e explicitação das características de cada um dos 2 tipos de perfis, apenas o perfil do tipo Madre admite soluções com perfis de secção transversal fechada, logo se “tubular” tiver valor verdadeiro é necessariamente um perfil “type”=1, se “tubular” for de valor falso “type” poderá ser qualquer um dos 2 perfis.

- Type_tag

A variável type_tag é um campo textual presente nas soluções geradas pelo simulador de cálculo estrutural, e que descreve a estratégia de cálculo adotada.

A “type_tag” pode assumir 3 descrições:

“Perfil com continuidade natural”: Indica perfis sem reforços adicionais, otimizados para vãos contínuos.

“Perfil com reforço duplo nos apoios e/ou nos vãos”: Refere-se a perfis que incluem reforços estruturais em áreas críticas para aumentar a resistência.

“Perfil com reforço duplo e/ou triplo nos apoios e tubular nos vãos”: Combina reforços múltiplos com secções tubulares em partes específicas da estrutura.

- Name

“name” é um campo textual que existe nas soluções calculadas pelo simulador e serve como identificador único e descritivo do perfil estrutural proposto para as coberturas e fachadas. O valor de “name” é composto por três partes, separadas por espaços:

Tipo de perfil: Indica a família do perfil (ex: Madre, Omega).

Dimensões da secção: Representadas como “largura x espessura” (ex: 50x1.5).

Tipo de aço: Especificação do material conforme as normas em vigor (ex: S350GD, S220GD).

- Section

A variável “section” é parte integrante da descrição técnica do perfil estrutural, e identifica a dimensão vertical do perfil (altura nominal), que influencia de forma direta o desempenho mecânico do mesmo.

- sectionName

A variável “sectionName” é um campo textual integrante das soluções calculadas pelo simulador, e que representa a designação completa das dimensões da secção transversal do perfil estrutural no formato padronizado. Esta variável sintetiza informações geométricas do perfil pela combinação da altura e da espessura, numa notação técnica universalmente reconhecida.

- Weight_M

A variável “weight_M” é uma variável numérica que representa o peso por metro linear do perfil estrutural expresso em quilogramas por metro (kg/m). A variável constitui um parâmetro de caracterização das propriedades físicas do perfil selecionado e influencia diretamente os custos de material, transporte e montagem da estrutura metálica.

O “weight_M” indica a massa linear do perfil estrutural, calculada tendo por base as dimensões geométricas da secção transversal (altura, largura, espessura) e a densidade do material especificado. O valor é expresso em kg/m e representa o peso do perfil por cada metro de comprimento instalado na estrutura.

- totalWeightAlignmen

A "totalWeightAlignment" corresponde ao peso total calculado para todo o alinhamento estrutural, considerando as características específicas dos perfis utilizados e as suas configurações geométricas. Esta variável é expressa em quilogramas (kg) e representa a soma do peso de todos os elementos estruturais que compõem o alinhamento, incluindo perfis principais, sobreposições e elementos de reforço quando aplicáveis.

- lengthAlignment

A variável "lengthAlignment" representa o comprimento total efetivo do alinhamento estrutural em sistemas de coberturas e fachadas em perfis metálicos. Este valor é calculado considerando a geometria do sistema, as condições de apoio e as especificidades dos perfis utilizados.

O "lengthAlignment" corresponde ao comprimento total da estrutura contínua, incluindo vãos principais ("modelSpans"), extensões da consola, sobreposições necessárias para as conexões e margens técnicas para a montagem dos perfis.

Para os perfis em causa, Madre e Omega, este valor é expresso em metros (m) e varia conforme a configuração geométrica e o tipo de perfil selecionado.

- totalWeightEstimated

A variável "totalWeightEstimated" representa uma estimativa preliminar do peso total dos perfis existentes nos elementos estruturais de cobertura e/ou fachada, calculada antes da análise detalhada das interações estruturais e das configurações específicas. Este valor serve como referência inicial para avaliações técnicas e económicas, e permite executar comparações rápidas entre as diferentes soluções projetadas.

O "totalWeightEstimated" corresponde a uma aproximação simplificada do peso total do sistema estrutural, geralmente associada ao comprimento total do alinhamento ("lengthAlignment"), ao peso linear teórico do perfil ("weight_M") e aos fatores de segurança genéricos. É expressa em kg.

- Height

A variável "height" representa a altura nominal do perfil estrutural, expressa em milímetros (mm), correspondendo à dimensão vertical da secção transversal do elemento. Este parâmetro é fundamental na determinação das características mecânicas, capacidade portante e compatibilidade geométrica em sistemas de cobertura e/ou fachadas metálicas. A "height" define a distância entre as fibras extremas superior e inferior do perfil.

- self_weigh

A variável "self_weight" representa o peso próprio linear do perfil, expresso em quilogramas por metro (kg/m), constituindo um parâmetro fundamental para a definição das cargas permanentes em projetos de estruturas. Este valor é calculado com base nas características geométricas do perfil e na densidade do material utilizado.

O "self_weight" corresponde ao peso linear teórico do perfil, calculado pela fórmula:

$$\text{self_weight} = A_{\text{secção}} \times \rho_{\text{aço}} \times 10^{-3} \tag{3. 3}$$

Onde:

$A_{\text{secção}}$: Área da secção transversal (mm²)

$\rho_{\text{aço}}$: Densidade do aço (7850 kg/m³)

- Width

A variável "width" representa a largura da aba do perfil metálico, expressa em milímetros. Nos dados analisados, observam-se valores consistentes. Os valores de "width" são consistentes para os diferentes tipos de perfis estruturais, incluindo as séries Madre e Omega. Esta dimensão constitui uma das principais características geométricas que definem a capacidade resistente e o comportamento estrutural do perfil.

Os dados estruturais apresentam valores padronizados da variável "width", que variam conforme o tipo e tamanho do perfil. A título exemplificativo, para os perfis Madre, as larguras variam entre 100 mm e 250 mm, sendo os valores mais comuns 100 mm, 150 mm, 200 mm e 250 mm.

- Thickness

A "thickness" representa a espessura da chapa de aço utilizada para o fabrico do perfil, expressa em milímetros (mm). Nos perfis normalizados, que servem como soluções para projeto de coberturas e fachadas, observa-se que a variável varia entre 1.5 mm e 3 mm para a maioria dos perfis das séries Madre e Omega.

A "thickness" integra a nomenclatura técnica dos perfis, aparecendo imediatamente após a largura ("width") na designação comercial. Por exemplo, no perfil Madre 200x2 S350GD, o valor "2" indica uma espessura de 2 mm.

- Steel

A variável "steel" representa a classificação do aço utilizado no fabrico de perfis metálicos, e que determina as suas propriedades mecânicas e de resistência. Esta especificação é crítica para garantir conformidade normativa, durabilidade e desempenho estrutural adequado às condições de projeto.

A variável "steel" segue o sistema de classificação da norma EN 10346, onde:

S: Indica "Structural steel" (aço estrutural)

XXX: Valor numérico correspondente ao limite de resistência mínima em Mpa (ex: 350 = 350 Mpa)

GD: Sufixo que identifica o revestimento a quente contínuo com zinco (Galvanized by Hot-Dip Process)

- Segments

A variável "segments" representa a distribuição física real do perfil metálico ao longo do eixo longitudinal da estrutura, detalhando as medidas específicas de cada tramo entre conexões ou reforços. Esta informação é crítica para o fabrico e montagem precisos dos elementos estruturais.

Os "segments" correspondem a um array numérico que define o comprimento individual de cada secção do perfil, por cada vão, expresso em metros. Por exemplo, no perfil do tipo Madre 200x1.5 S350GD2, se os segments forem especificados como [0.3, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 0.3], totalizam 12.8 metros de comprimento, com 1.9 m de perfil a colocar nos vãos principais, e 0.3 m nas consolas.

- Overlappings

A variável "overlappings" (sobreposições) representa a configuração de conexão entre segmentos adjacentes dos perfis metálicos, definindo a quantidade de camadas do material que são utilizadas nas junções para conferir resistência e estabilidade estrutural.

Os "overlappings" são definidos como um array numérico, onde cada valor corresponde a um segmento do perfil, indicando:

1: Conexão simples, sem sobreposição adicional.

2: Sobreposição dupla, com uma camada de reforço na junção

≥3: Reforços múltiplos (menos comum, presente em casos especiais)

- governingCheck

A variável "governingCheck" representa a verificação crítica que determina a adequação de um perfil metálico para uma aplicação específica, identificando o mecanismo de falha dominante e quantificando a margem de segurança.

O governingCheck é um objeto de estrutura completa com três elementos-chave:

1) x: Posição crítica ao longo do perfil (em metros) onde ocorre a solicitação máxima.
2) utilizationRatio: Razão entre esforços atuantes e a capacidade resistente ($0 \leq$ valor ≤ 1)
3) check: Código normativo da verificação determinante, ou seja, a que apresenta (ex: "6.3.2. d" do EC-3)

- minimumSpacing

A variável "minimumSpacing" representa o espaçamento mínimo requerido entre elementos estruturais em projetos, especificamente no contexto de dimensionamento de perfis metálicos para utilizar em coberturas e fachadas.

A "minimumSpacing" define a distância mínima entre os eixos dos perfis estruturais paralelos, com valores que variam entre 1.25 m e 2.5 m conforme o projeto. Este parâmetro é determinado por:

1) Características dos perfis (ex.: Madre 50x1.5 vs. Omega 200x1.5)
2) Magnitude das cargas atuantes (permanentes, vento, neve)
3) Limites de deformação (ex.: 1/200 do vão)

- Id

A variável "id" constitui um identificador único numérico utilizado para referenciar elementos específicos em sistemas de gestão de projetos estruturais. A pertinência de utilização na solução serve na identificação de configurações globais até à especificação de componentes individuais.

O "id" atua como código único de projeto e permite:

1) Rastreabilidade completa de diferentes versões de projeto
2) Associação automática com sistemas de gestão documental
3) Vinculação com bases de dados externas (ex: CAD/BIM)

O valor da variável "id" identifica de forma única cada solução dentro do array solutions, permitindo distinguir e rastrear cada alternativa de perfil que dá resposta às condições analisadas. Por exemplo, se "id" for 0 em "solutions" isto significa que é a 1º solução calculada e listada pelo simulador. "id" é uma variável identificativa, não tem relevância do ponto de vista do cálculo estrutural.

Por fim, como já foi exposto em cima, o algoritmo faz sair uma série de soluções, priorizando a solução que considera como ótima, permitindo na mesma ao utilizador selecionar outra solução, se entender que está mais de acordo com o que pretende ou com as condições de construção, como por exemplo, a facilidade de aplicabilidade da solução em contexto real de obra que não obrigue a contratar uma equipa maior ou mais especializada.

(utilizador tem a opção de selecionar uma outra solução, também possível, que melhor se enquadre nos princípios prioritários que definir, por exemplo, a facilidade de aplicabilidade da solução em contexto real de obra que não obrigue a contratar uma equipa maior ou mais especializada.)

A título explicativo, cada uma das soluções finais calculadas pelo simulador é apresentada da seguinte forma:

solutions: [{'type': 1,

'tubular': False,

'type_tag': 'Perfil com continuidade natural',

'name': 'Madre',

'section': 50,

'sectionName': '50x1.5',

'weight_M': 2.4,

'totalWeightAlignment': 29.2,

'lengthAlignment': 12,

'totalWeightEstimated': 1.3,

'height': 54,

'self_weight': 2.43,

'width': 100,

'thickness': 1.5,

'steel': 'S350GD',

'segments': [0.3, 1.9, 1.9, 1.9, 1.9, 1.9, 1.9, 0.3],

‘overlappings’: [1, 1, 1, 1, 1, 1, 1, 1],

‘governingCheck’: {‘x’: 2.1999999999999997, ‘utilizationRatio’: 0.898164052197341, ‘check’: ‘6.3.2.d’},

‘minimumSpacing’: 2,

‘id’: 0}

3.2. EXPLICITAÇÃO DA CONSTRUÇÃO DO MODELO DE MACHINE LEARNING

O objetivo do modelo de ML passa essencialmente por 2 objetivos principais. O 1º será a capacidade de, perante um conjunto de dados, ser capaz de os importar e, posteriormente, através de treino supervisionado, identificar as relações numéricas entre as variáveis de entrada e as variáveis de saída. No momento em que os resultados no pós treino sejam satisfatórios (análise dos valores das métricas inseridas dentro do modelo para avaliação dos resultados), terá que ser avaliada a capacidade de generalização do modelo, ou seja, perante um conjunto de dados novos que não tenham sido utilizados durante a fase de treino, avaliar os resultados que o modelo dá para as soluções e compará-las com as que são consideradas como verdadeiras, que são as soluções obtidas pela execução do simulador.

O 2º passa pela capacidade de o modelo generalizar soluções para dados novos não antes vistos e não incluídos dentro da base de dados, através da aprendizagem que se espera que tenha adquirido quando concluído o “1º” objetivo.

Por estes motivos, e pela explicação exposta no subcapítulo 2.2.1.4, o problema a tratar é um problema de regressão.

O modelo de ML será constituído por 2 programas, o P1 e P2, que serão explicitados de forma pormenorizada na sequência deste subcapítulo. A linguagem escolhida para executar a construção do modelo foi a linguagem Python, por se tratar de uma linguagem eficiente, simples de entender e programar, com uma vasta biblioteca de algoritmos de implementação simplificada em modelos de ML.

A Base de Dados utilizada no desenvolvimento do modelo é constituída por todos os casos já observados e soluções calculadas pelo simulador. Totaliza um número impressionante de 60 000 casos distintos e engloba tanto as variáveis de entrada, que caracterizam de forma completa o problema em causa, como as variáveis de saída, que definem todas as soluções possíveis, dentro dos perfis existentes do tipo Madre e Omega, que cumprem as normas em vigor.

No entanto, foi necessário estratificar os dados, separando as soluções do tipo Madre e Omega, uma vez que não era expectável que um modelo de regressão conseguisse processar adequadamente as diferenças estruturais e técnicas entre os dois tipos de perfis num mesmo processo de aprendizagem (classificar os perfis). Esta heterogeneidade dos dados resultaria em dificuldades acrescidas ou mesmo impossibilitaria o processo de treino eficaz do modelo. Após a estratificação, apenas os dados dos perfis Madre foram utilizados para treinar o modelo.

Além da estratificação acima, foram incluídas ainda três restrições relevantes, sendo elas: a obrigatoriedade da estrutura de cobertura ou fachada ser composta por 8 vãos, os vãos finais serem em consola, ou seja, com um apoio fixo numa das extremidades e livre na outra, e só considerar soluções com perfis de secção 100 mm.

Este procedimento otimiza e acelera a aprendizagem do modelo, limitando os casos de estudo a sistemas estruturais complexos, mas parametrizados por um número fixo de vãos (n=8), com condições de contorno bem definidas, especificamente com a presença de consolas nos vãos extremos.



Figura 22 – Esquema estrutural da cobertura ou fachada, composto por 8 vãos e 2 consolas, conforme definido pelas restrições aplicadas aos dados

Este tipo de abordagem favorece a capacidade de aprendizagem do modelo e assegura que as relações entre variáveis de entrada e de saída mantenham a coerência do ponto de vista da análise estrutural, e que sejam similares em sistemas estruturais distintos, com ou sem consolas, e com diferente número total de vãos.

As restrições impostas permitem a criação de um conjunto de dados para treino coeso e que não contenha um número ainda mais elevado de variáveis de entrada e saída, o que, por consequência, acelera o processo de aprendizagem do modelo sem que, com isto, o comprometa para situações em que necessite de generalizar soluções para casos com geometria estrutural diferente (Exemplo: 10 vãos totais em que os vãos extremos são consolas).

No contexto em que esta Dissertação foi desenvolvida, em que essencialmente se procura averiguar a possibilidade da aplicabilidade de um modelo de ML para acelerar o processo de cálculo do simulador, restrições que acelerem o processo de aprendizagem são positivas, na medida em que é avaliada, em conjuntos de dados mais pequenos, a fiabilidade da capacidade de aprendizagem do modelo. Se o modelo não apresentar capacidade para generalizar soluções em situações circunscritas da base de dados, também não irá ser capaz de o fazer para outros casos presentes nessa mesma base. A juntar a isto, o facto de não ser viável a criação de um modelo de ML que consiga, além de identificar as relações entre as variáveis, produzir uma saída que clarifique o tipo de perfil a utilizar, o método processual correto será criar um modelo para cada tipo de perfil, que seja capaz de identificar as relações entre as variáveis, generalizar as solução com valores análogos aos calculados pelo simulador, mas que seja de fácil perceção o perfil que corresponde à solução generalizada; ao contrário de um modelo que generalize igualmente de forma correta, mas que não seja percetível a que perfil correspondem os valores obtidos.

A título de exemplo, imaginemos que, para um conjunto de dados de entrada, o modelo generaliza 4 soluções diferentes para cada tipo de perfil Madre. As soluções que resultam da execução da rede neuronal até podem estar certas, mas, se não for possível perceber a qual dos perfis corresponde cada uma das soluções, a escolha do perfil torna-se impossível, o que impossibilita a utilização do algoritmo.

A opção preferível será a constituição de um número de modelos igual ao número total de perfis que existem, e fazer sair as soluções com a identificação do perfil em causa, para ser percetível ao utilizador e à empresa como irá ser constituída a solução final.

No final, o simulador deverá correr o programa P2 de cada modelo criado, para cada perfil existente, e fazer sair as soluções otimizadas que cumpram as normas em vigor e que dão resposta ao problema constituído pelo conjunto de variáveis de entrada, identificando os valores de solução requeridos e os perfis a que corresponde cada uma dessas soluções.

O algoritmo de aprendizagem utilizado foi uma Rede Neuronal, por serem altamente potentes e eficazes na resolução de problemas de regressão, e por apresentarem simplicidade de programação em linguagem Python, o que facilita a correção de erros que possam surgir durante o processo de construção

do modelo e, ao mesmo tempo, confere maior capacidade de interpretação e leitura do processo operacional do modelo.

3.2.1. Explicitação da Operacionalidade do programa P1

3.2.1.1. Construção e Import da Base de Dados

A existência da base de dados é imprescindível no desenvolvimento de um projeto em ML. A base de dados é o sistema organizado que armazena toda a informação relevante para o processo de criação e de treino de um modelo de ML, e é sobre ela que todo o restante processo de conceção se desenvolve.

A base de dados foi fornecida em formato JSON, que posteriormente foi restruturado para acomodar as restrições em cima identificadas.

Com as restruturações, existem 88 linhas no JSON, que identificam que vão existir pelo menos 87 casos distintos em que, pelo menos, uma das soluções é um perfil do tipo Madre com secção 100 mm.

3.2.1.2. Estruturação dos Dados

O import da Base de Dados não é, por si só, suficiente para garantir que o modelo de ML será funcional e que apresentará resultados condizentes com os desejados. É necessário proceder de início à definição da variável alvo, que neste caso em específico são os comprimentos dos segmentos dos perfis metálicos a utilizar nas coberturas e/ou fachadas, incluindo o comprimento dos casos em que exista sobreposição.

A variável alvo fica assim definida como uma variável multi-target, em que o objetivo do modelo será prever os valores da variável “segments”, “overllapings” e “l0verlapMinEqualSpansEqualLoads” (comprimento de sobreposição).

Posteriormente, é definido o numero de “features” (características ou atributos que servem como variáveis de entrada em modelos de ML) extraídas das variáveis de entrada presentes na base de dados, e que, em conjunto com a ou as variáveis-alvo, formam o conjunto de dados que o modelo irá utilizar para aprender relações e padrões entre os dados e, com isso, ser capaz de generalizar soluções.

No entanto, a correta definição das “features” envolve a organização e seleção das variáveis de interesse que estão dentro da base de dados. Em problemas de regressão, e em especial no problema tratado nesta Dissertação, é fundamental a identificação, dentro das variáveis de input, daquelas que, de facto, são relevantes para a aprendizagem do modelo.

A seleção de todas as variáveis pode ser excessiva, e inclusive, dificultar ou impossibilitar a aprendizagem do modelo. Nas situações onde muitas variáveis irrelevantes ou redundantes são incluídas, o modelo pode sofrer de sobreajuste (overfitting), tornando-se incapaz de generalizar de forma assertiva para novos dados. Além disso, o aumento do número de variáveis aumenta a complexidade computacional e pode levar a problemas de instabilidade numérica e dificuldade de convergência dos algoritmos de aprendizagem.

No entanto, a escolha de um número muito pequeno de variáveis pode provocar perda de informação relevante e o modelo pode não ser capaz de captar relações fundamentais entre os inputs e os outputs, o que resulta em previsões imprecisas ou enviesadas. Quando o modelo se torna demasiado simples, é

incapaz de representar a complexidade dos dados, o que, por consequência, se reproduz no baixo desempenho, tanto na aprendizagem como na validação dos dados.

Seguindo o critério exposto nos parágrafos anteriores, as variáveis selecionadas como "features" para serem incluídas no Dataframe foram as seguintes:

- 'hasCantilever'
- 'transversalDistance'
- 'supportWidth'
- 'direction'
- 'inclination'
- 'modelSpans'
- "loads"
- "spanRestraints"

A variável "internal" é extremamente relevante no processo de cálculo e determinação dos valores dos comprimentos dos perfis, do número de sobreposições estruturais necessárias e do comprimento dessas sobreposições. No entanto, a restrição da base de dados só para incorporar casos com um número fixo de 8 (oito) vãos e em que os vãos extremos são 2 (duas) consolas resultou na não existência de nenhum caso em que a variável "internal" tivesse sido ativada. Por esta razão, foi excluída do processo de treino.

No entanto, foi necessário proceder a uma reestruturação de algumas destas variáveis e à criação de outras, para permitir a correta representação dos dados no modelo, assegurando a compatibilidade com o formato de entrada exigido pelo algoritmo de aprendizagem e otimizar o processo de treino.

No final, o modelo faz sair a composição do Dataframe, uma matriz com 81 linhas e 50 colunas, e o número de amostras válidas, que totaliza 81.

Quando a variável for um dicionário, como o caso da variável "loads", a lógica é a mesma mas necessita de um passo transato à desagregação das colunas complexas, que consiste na separação prévia de cada chave do dicionário em colunas individuais. Assim, cada campo do dicionário (por exemplo, "case", "type" e "value" em "loads") é extraído e transformado em colunas próprias no DataFrame. Posteriormente, estas colunas são também elas desagregadas para que as variáveis de entrada sejam compatíveis com a arquitetura da rede neuronal.

De forma a otimizar o processo de aprendizagem do modelo, procedeu-se a outra reestruturação, desta vez na variável "modelSpans", que continha o valor do comprimento de cada um dos vãos que componham a estrutura.

Neste processo, os valores numéricos brutos (A título de exemplo, 1.9 m) dos vãos foram convertidos para relações percentuais em função do comprimento total da estrutura, o que permitiu normalizar a informação e eliminar a dependência da escala absoluta. Esta abordagem facilita a aprendizagem de padrões estruturais pela rede neuronal, tendo em conta que o cálculo de momentos fletores e a distribuição de esforços dependem fundamentalmente das proporções relativas entre os vãos. Além disso, a utilização das relações percentuais contribui para a estabilidade numérica e acelera a convergência do modelo, o que irá, com certeza, melhorar a sua capacidade de aprendizagem.

Nas situações em que a desagregação das variáveis origina, em casos específicos, a inexistência de valor numa determinada coluna no DataFrame, o campo correspondente é preenchido com o valor NaN (Not a Number), no entanto, este valor é substituído por 0 através do comando "fillna" que substitui todos os valores ausentes do DataFrame por 0. No contexto das variáveis em causa, a substi-

tuição por 0 é correta, porque a não existência de valor significa para efeitos práticos do cálculo estrutural adotar o valor de 0.

Esta substituição é mandatária para poder utilizar o DataFrame na arquitetura das redes neuronais, uma vez que estes algoritmos exigem que todas as variáveis de input estejam representadas por valores numéricos definidos e de dimensão fixa. Assim, ao garantir que não existem valores ausentes (NaN) e que todos os campos possuem um valor válido — neste caso 0, quando a ausência de valor corresponde à inexistência da variável — assegura-se a integridade dos dados e a compatibilidade com os requisitos de entrada dos modelos de aprendizagem.

Para a manipulação, organização e preparação dos dados foi utilizada a biblioteca Pandas, uma ferramenta fundamental para análise de dados em Python.

O Pandas é uma biblioteca Python de código aberto projetada para análise de dados e fornece recursos de alto desempenho para a manipulação de dados. A biblioteca permite que os utilizadores trabalhem com estruturas de dados semelhantes a folhas de cálculo, facilitando tarefas como carregamento, limpeza e análise de dados. O Pandas é particularmente eficaz a lidar com dados tabulares, onde oferece várias funções e métodos para otimizar o processamento dos dados. A biblioteca é amplamente utilizada em vários campos científicos, muitas vezes em conjunto com outras bibliotecas como NumPy e Matplotlib. [42]

No final, o modelo faz sair a composição do Dataframe: uma matriz com 81 linhas e 50 colunas, e o número de amostras válidas, que totaliza 81.

3.2.1.3. Normalização e Seleção dos dados de treino e validação

A seleção dos dados para treino e validação constitui uma etapa fundamental no desenvolvimento de modelos de ML, uma vez que determinam a forma como o modelo irá aprender os padrões que existem na relação entre as variáveis de input e de output. Nesta etapa, o conjunto de dados é segmentado em dois subconjuntos principais: um utilizado para treinar o modelo, onde são ajustados os parâmetros a partir dos dados disponíveis, e outro reservado para a validação, que permite aferir o desempenho do modelo em dados que não foram utilizados no processo de aprendizagem.

Esta divisão é crucial para avaliar a capacidade de generalização do modelo e prevenir fenómenos de sobreajuste, assegurando resultados corretos nas situações em que o modelo é submetido a casos que desconhece e que não se encontram sequer presentes no Dataframe.

As divisões tradicionais entre dados de treino, validação e teste são da seguinte magnitude: 70% (setenta por cento) – 10% (dez por cento) – 20% (vinte por cento), 60% (sessenta por cento) – 30% (trinta por cento) – 10% (dez por cento), 50% (cinquenta por cento) – 25% (vinte e cinco por cento) – 25% (vinte-cinco por cento), respetivamente.

No entanto, a existência de um número relativamente pequeno (em contexto de ML) de amostras válidas impõe limitações à aplicação direta destas proporções clássicas, tornando necessário ajustar a divisão dos dados, de forma a garantir que ambos os subconjuntos — treino e validação — disponham de uma quantidade suficiente de exemplos que permitam ao modelo aprender as relações entre as variáveis e, posteriormente, validar estes resultados com dados que não existiam no conjunto de dados para treino.

Considerando estas limitações, os dados do Dataframe foram distribuídos aleatoriamente entre os conjuntos de treino e validação. Adotou-se a proporção de 95% dos dados totais para o conjunto de treino e os restantes 5% para o conjunto de validação. Como o conjunto de dados para treino é limitado, a

preferência é atribuir uma grande quantidade de dados para treino para o modelo conseguir compreender e reter a relação entre as variáveis, e selecionar um pequeno conjunto de dados para validação, para aí avaliar a capacidade de generalização do mesmo.

A distribuição aleatória dos dados foi efetuada através da utilização do "train_test_split" do módulo "sklearn.model_selection".

A função "train_test_split" do módulo "sklearn.model_selection" é essencial na divisão de conjuntos de dados em subconjuntos de treino e validação, o que constitui uma prática fundamental em modelos de ML. Esta função facilita a avaliação do desempenho do modelo ao garantir que o modelo é treinado num subconjunto de dados e validado noutro, o que proporciona uma melhor avaliação da capacidade real de generalização do modelo. [43]

Daqui resultou que as dimensões do conjunto de treino para a variável X fosse (76, 57) ou seja, 76 amostras e 57 "features" e por sua vez o conjunto de treino Y (76, 17) ou seja 76 amostras e 17 variáveis de output ou variáveis "alvo".

Posteriormente, procedeu-se à normalização dos dados, que constitui uma etapa preparatória que transforma atributos quantitativos antes da aplicação de algoritmos de ML. A relevância deste procedimento reside na prevenção da dominância de certas variáveis sobre outras, especialmente quando existem discrepâncias significativas nas amplitudes de variação.[44] A título exemplificativo para o caso tratado nesta dissertação, o valor das cargas em alguns casos, serem significativamente maiores que os valores dos vãos, o que poderia resultar em que na fase de treino o modelo desse maior preponderância ao valor das cargas do que ao valor dos vãos o que não é correto, já que a preponderância dos 2 no contexto do problema aqui tratado é a mesma.

A normalização dos dados de treino foi efetuada através da utilização do "sklearn.preprocessing" do módulo "StandardScaler", e para os conjuntos de dados X de treino e validação. Só faz sentido proceder à normalização dos dados nas variáveis independentes, já que nas dependentes não existe o risco de dominância de certas variáveis sobre outras.

O StandardScaler constitui uma metodologia de pré-processamento que aplica a normalização Z-score através da centralização dos dados através da remoção da média e do ajuste para variância unitária. Este processo de padronização transforma cada valor, subtraindo a média correspondente e procedendo à divisão pelo desvio padrão, o que origina uma distribuição com média zero e variância igual a 1. [45]

3.2.1.4. Arquitetura da Rede Neuronal

A arquitetura da Rede Neuronal implementada para o projeto P1 foi concebida especificamente para resolver o problema de regressão multi-alvo em questão, e já explicado no decorrer do presente capítulo. O problema envolve a predição simultânea de comprimentos de segmentos, número e comprimento de sobreposição nos perfis estruturais. Esta abordagem foi selecionada após a análise criteriosa das características do conjunto de dados e dos requisitos técnicos do problema, considerando a natureza numérica e multidimensional dos dados de entrada e saída.

A estrutura da rede segue um padrão "Sequential", composta por três camadas, densamente conectadas, que formam uma topologia progressivamente redutora. A primeira camada recebe os dados de entrada pré-processados, com dimensionalidade correspondente às 57 características identificadas no conjunto de treino. Esta camada inicial emprega 64 unidades neuronais com função de ativação ReLU, escolhida pela sua eficiência computacional, capacidade de mitigar o problema de gradientes desapa-

recidos durante o treino e capacidade em percecionar relações não lineares entre os dados utilizados para treino.

A segunda camada, com 32 neurónios e função de ativação igual à anterior ReLU, opera como camada oculta intermediária, que permite a extração de características hierárquicas e não-lineares dos dados. A redução dimensional de 64 para 32 unidades foi estrategicamente definida para equilibrar a capacidade de aprendizagem e prevenir o sobreajuste do modelo, tendo por consideração o tamanho limitado do conjunto de dados (81 amostras válidas).

A camada de saída, com dimensionalidade dinâmica determinada pelo cálculo do número de variáveis a serem preditas, utiliza ativação linear para desenvolver previsões contínuas. Este valor é calculado como a soma das colunas de segmentos (8), sobreposições (8) e a variável adicional "OverLapSegment", totalizando 17 neurónios de saída. A linearidade na camada final é essencial para valores numéricos contínuos, típicos de problemas de regressão.

A configuração de treino emprega o optimizador ADAM com função de perda MSE, combinados para minimizar erros quadráticos nas previsões dos comprimentos. Complementarmente, a métrica MAE é monitorizada durante o treino e validação para fornecer uma avaliação complementar crítica:

- Interpretação física direta:

O MAE expressa os erros nas unidades das variáveis de saída (para este problema em específico, em metros), o que permite correlacionar diretamente desvios de previsão com requisitos de tolerância estrutural. Um MAE de 0.05 m indica erro médio de 5 centímetros; é, por isso, uma métrica tangível para problemas de análise e dimensionamento estrutural.

- Complementaridade à perda MSE:

Enquanto o MSE penaliza erros maiores (útil para evitar outliers perigosos), o MAE fornece uma medida robusta do erro médio absoluto, menos sensível a valores extremos. Esta dualidade permite equilibrar a otimização entre sensibilidade a grandes erros e precisão média geral.

- Validação do desempenho prático:

Dada a natureza física do problema (comprimentos de segmentos em metros), o MAE serve como ponte entre a avaliação estatística e requisitos práticos de projeto. Durante o treino, o "val_mae" atuou como referência chave para decisões de "early stopping", garantindo que o modelo se mantinha adequado em dados não vistos.

Para a implementação dos modelos de ML neste projeto, foram utilizadas as seguintes bibliotecas e técnicas essenciais:

- TensorFlow/Keras: Para a construção e treino das redes neuronais, utilizando as classes "Sequential" (modelo sequencial) e "Dense" (camadas totalmente conectadas).
- Scikit-learn: Para pré-processamento de dados com "StandardScaler" (padronização Z-score) e "train_test_split" (validação cruzada).
- Pandas/NumPy: Para manipulação estruturada dos dados extraídos do JSON de entrada, transformação de variáveis e normalização da variável "spans".

- Adam Optimizer: Algoritmo de otimização adaptativo para eficiência em gradientes esparsos, combinado com função de perda MSE.

Esta stack tecnológica permitiu desenvolver um modelo de regressão multi-alvo capaz de prever comprimentos de segmentos e configurações de sobreposição em perfis estruturais, com métricas de avaliação, como o MAE, para interpretação física direta dos erros.

O "TensorFlow" é um framework open-source para computação numérica de alto desempenho, especializado em gradientes automáticos e execução distribuída de grafos computacionais. A sua arquitetura permite a implementação eficiente de modelos de ML em CPUs, GPUs e TPUs. O "TensorFlow" foi projetado para simplificar a expressão de algoritmos de aprendizagem em modelos e a sua execução eficiente em sistemas heterogêneos, o que contribui para a resolução de desafios de escalabilidade em cenários de produção. [46]

O "Keras" está integrado nativamente no "TensorFlow" e oferece uma API de alto nível que abstrai complexidades de baixo nível. O "Keras" acelera o desenvolvimento de protótipos sem sacrificar desempenho, sendo adotado em casos práticos de ML devido à sua intuição. [47]

O modelo "Sequential" implementa um paradigma de construção de redes neuronais por composição linear de camadas. É ideal para arquiteturas "feedforward", organiza camadas em sequência unidirecional, onde a saída de uma camada é a entrada da próxima. Esta abordagem reduz a complexidade de código para problemas de regressão e classificação com dados tabulares, sendo particularmente eficaz para fluxos de dados sem ramificações. [47]

Camadas Dense são camadas totalmente conectadas que realizam a operação output = activation (dot(input, kernel) + bias), onde "kernel" é a matriz de pesos que resulta do treino e "bias" o vetor de viés. Cada neurónio conecta-se a todos os neurónios da camada anterior, o que lhes permite aprender relações complexas entre características. Camadas densamente conectadas são essenciais para extrair representações hierárquicas em dados estruturados, transformando combinações não-lineares de atributos em padrões. [48]

Esta arquitetura demonstra o equilíbrio ideal entre complexidade e eficiência para o problema específico, permitindo capturar relações não-lineares nos dados, enquanto mantém requisitos computacionais adequados ao tamanho do Dataset.

3.2.1.5. Treino da Rede Neuronal

O processo de treino da rede neuronal do Programa P1 foi implementado com medidas rigorosas de reprodutibilidade e controlo de sobreajuste, que garantem a estabilidade dos resultados do modelo final. A configuração adotada seguiu os seguintes princípios fundamentais:

- Estabilidade e Reprodutibilidade

Foram implementadas sementes (seeds) determinísticas em todas as stack computacionais:

1) random.seed (42) para operações Python puras.
2) np.random.seed (42) para gerar números aleatórios no "NumPy".
3) tf_random:set_seed (42) para operações estocásticas no "TensroFlow".

Esta abordagem garante que os resultados sejam totalmente replicáveis em diferentes execuções, requisito essencial na análise científica dos resultados.

- Estratégia de Early Stopping

O "Early Stopping", com valor de "paciência" de 500 épocas, foi implementado para interromper o treino automaticamente, quando a diminuição do erro na validação estagna, prevenindo o sobreajuste e otimizando a capacidade preditiva do modelo. Além disso, a introdução da "restore_best_weights" como verdadeira permite a recuperação automática dos "melhores pesos" no final do treino (os pesos de cada neurónio, com os quais o modelo conseguiu obter o menor erro no processo de validação). Este mecanismo previne o sobreajuste do modelo, enquanto permite que este explore plenamente o espaço de parâmetros.

- Processo de Treino do modelo

Para treinar o modelo, utilizou-se "batch size" igual a 1, que corresponde a treino estocástico, com o objetivo de maximizar a precisão em conjuntos de dados pequenos. (Quando se escolhe batch size = 1, cada amostra individual faz um passo de atualização separado e os pesos do modelo são ajustados de forma imediata). Estabeleceu-se um limite de 5000 épocas, embora este teto nunca tenha sido alcançado graças ao mecanismo de "early stopping", que interrompe o treino assim que o valor do erro na validação estagna e não diminui nas 500 épocas consecutivas seguintes.

Os 5% do conjunto de dados reservados para a validação, permitiu monitorizar em tempo real o desempenho do modelo em dados que este não conhecia da fase de treino.

No final deste processo são obtidos os resultados das métricas MSE e MAE, que serão analisados no capítulo seguinte.

No processo de treino do modelo recorreu-se às bibliotecas Python, de referência em cálculo numérico, e ML, nomeadamente NumPy, TensorFlow, e bibliotecas auxiliares para a preparação dos dados e otimização do processo de treino. A combinação destas ferramentas permite uma implementação eficiente e reprodutível do pipeline de treino, desde o carregamento e normalização dos dados até à otimização dos parâmetros do modelo.

O NumPy é a biblioteca fundamental da linguagem Python para programação. Com arrays multidimensionais, a biblioteca fornece estruturas de dados e operações vetoriais de alto desempenho. Os arrays NumPy suportam dados homogéneos e permitem o acesso eficiente à memória do computador, o que é crucial para operações matemáticas em larga escala. [49]

3.2.1.6. Armazenamento da Matriz de Pesos

A matriz de pesos é a matriz onde estão alocados os valores numéricos atribuídos às conexões entre neurónios de camadas adjacentes. Esses valores representam o conhecimento das relações entre as variáveis de input e as variáveis alvo, que o modelo adquiriu no final da fase de aprendizagem. Cada elemento desta matriz indica a intensidade da contribuição de um neurónio de uma camada anterior para um neurónio da camada imediatamente a seguir, enquanto os vetores de viés controlam o limiar de ativação.

No fim da fase de treino do Programa P1, a matriz de pesos é armazenada para garantir a reprodutibilidade da aprendizagem da Rede Neuronal. O armazenamento da matriz de pesos é efetuado através da serialização, em formato HDF5 ou formatos nativos do "framework", que garantem fidelidade e integridade dos dados numéricos. Além dos pesos, são guardados também os vieses e os metadados da

arquitetura já que só assim é possível replicar o modelo com a configuração original num Programa P2, em que o objetivo será agora generalizar soluções para dados de entrada que não existam no Dataframe.

O processo de treino e armazenamento de modelos de redes neuronais envolve uma sequência estruturada de operações, desde o pré-processamento dos dados até ao armazenamento das características de aprendizagem dos modelos. Para garantir eficiência, reprodutibilidade e interoperabilidade, recorreu-se a bibliotecas Python especializadas em manipulação numérica, aprendizagem profunda e serialização de objetos. A combinação de "TensorFlow/Keras" para construção e treino da rede neuronal, "pandas" para gestão estruturada de dados, e "joblib" para a serialização de objetos de pré-processamento constitui um fluxo consolidado na literatura de engenharia de ML.

O módulo "Keras", integrado nativamente ao "TensorFlow", fornece APIs de alto nível para a construção e treino de redes neuronais. No contexto deste subcapítulo, utilizou-se o "model.save_weights()", que é um método que armazena os parâmetros treinados do modelo (pesos e viés) no formato binário HDF51. Este formato permite a reconstituição exata da arquitetura sem dependências externas, crítico para implantação em ambientes heterogéneos.

A biblioteca panda oferece estruturas de dados e operações para manipulação eficiente de tabelas numéricas e séries temporais. Destacou-se no projeto através da utilização do "DataFrame.to_pickle()"que serializou o DataFrame em formato binário Python (pickle), e preservou os metadados, tipos de dados e índices. Esta abordagem é vantajosa para fluxos de trabalho que exijam reconstituição precisa de estruturas complexas entre programas.

Joblib é uma biblioteca especializada em serialização eficiente de objetos Python, particularmente otimizada para arrays NumPy. A função dump(), utilizada para persistir o objeto StandardScaler oferece compactação integrada (via zlib) para redução de espaço no disco e a recuperação consistente do estado interno do pré-processador, que permite que novas amostras sejam normalizadas com os mesmos parâmetros (média e desvio padrão) do conjunto de treino.

3.2.2. Explicitação da Operacionalidade do Programa P2

3.2.2.1. Construção e Import da Base de Dados

Como expresso anteriormente, ao longo desta Dissertação, o objetivo do Programa P2 é funcionar como um modelo preditivo de soluções, que seja capaz de, perante um conjunto de dados novos que não constem do Dataframe, generalizar soluções para os comprimentos dos perfis metálicos, número e comprimento das sobreposições a aplicar nas coberturas e/ou fachadas de uma estrutura a ser projetada.

O Programa P2 inicia-se com a reconstrução da base de dados já estruturada em P1. Para tal, importa-se o ficheiro "X_features.pkl" que contém o DataFrame das características normalizadas e serializadas, através do recurso à biblioteca pandas e joblib, e o ficheiro y_shape.txt, onde está guardada a dimensão do vetor de saídas (número de colunas de y), para configurar a camada de output da rede neuronal.

Por fim, utiliza-se a função load() do "joblib" para reconstituir o objeto "StandardScaler" já treinado em P1, e que permite aplicar exatamente a mesma transformação de normalização aos novos dados de entrada.

Este procedimento garante que as novas variáveis de input sejam submetidas ao mesmo pré-processamento replicável que foi utilizado no treino original, evitando discrepâncias que possam introduzir data leakage ou enviesar as previsões

3.2.2.2. Definição da Arquitetura do modelo

Com os dados de entrada já estruturados, o Programa P2 reconstrói a arquitetura da Rede Neuronal Sequencial, idêntica à utilizada no treino do Programa P1. A rede consiste numa camada de entrada de dimensão igual a X_novos_norm.shape[1]; duas camadas ocultas com 64 e 32 neurónios, respetivamente, ambas com ativação ReLU, e uma camada de saída Y_novos, cuja ativação é linear.

A compilação utiliza o optimizador ADAM, o erro quadrático médio (MSE) e métrica MAE, estando, por isso, alinhada com os parâmetros de treino originais de forma a garantir consistência nos resultados preditivos.

3.2.2.3. Import da Matriz de Pesos e Previsão de Soluções

Depois de definir o modelo, o Programa P2 importa a Matriz de Pesos do programa P1, através do método "load_weights()", que recupera integralmente o modelo treinado em P1.

Com o modelo carregado e o vetor de entrada normalizado, procede-se à predição das soluções. O resultado" y_pred" é um array de dimensão (n_amostras, Y_novos), onde cada linha contém os comprimentos previstos de cada segmento do perfil do tipo Madre de secção 100 mm a utilizar no recobrimento das coberturas e/ou fachadas; os valores previstos para o número de sobreposições nos apoios entre os segmentos e o valor previsto para o comprimento do perfil de sobreposição.

3.2.2.4. Pós Processamento dos Dados e Outputs de Resultados

O Programa P2 converte o array das soluções preditas num DataFrame dos pandas, tornando a informação mais acessível e organizada para o utilizador. Em seguida, gera uma tabela em Excel, estruturada, onde cada segmento é associado ao respetivo comprimento previsto do perfil do tipo Madre de secção 100 mm, ao número de sobreposições a aplicar em cada nó de ligação (apoio) da estrutura, bem como ao valor do comprimento de sobreposição correspondente.

4
ANÁLISE DOS RESULTADOS

4.1. A ANÁLISE DE RESULTADOS EM ML

A análise de resultados constitui uma etapa central em qualquer projeto de ML, assumindo particular relevância no contexto da presente Dissertação. É neste momento que se verifica, de forma rigorosa e quantitativa, a eficácia dos modelos desenvolvidos, recorrendo a métricas objetivas que permitem aferir o grau de aproximação entre as previsões do modelo e os valores reais do problema em estudo. Para além de validar o desempenho do algoritmo, a análise dos resultados possibilita a identificação de padrões de erro, a deteção de eventuais outliers do conjunto de dados e a compreensão das limitações do modelo. Na estrutura da Dissertação, a análise de resultados é determinante para fundamentar as conclusões do trabalho, justificar as opções metodológicas tomadas e sustentar a aplicabilidade prática das soluções propostas. Desta forma, a análise de resultados não só encerra o ciclo de desenvolvimento do modelo, como também assegura a credibilidade e o rigor científico da abordagem seguida.

4.1.1. ANÁLISE DOS VALORES OBTIDOS PARA AS MÉTRICAS MSE E MAE NO P1

Os resultados obtidos na fase de treino do modelo de ML, na ordem de 0.006 para MSE (erro quadrático médio) e de 0.0150 para MAE (erro médio absoluto) , evidenciam um desempenho muito satisfatório na previsão das variáveis de interesse do problema em questão. O valor reduzido do MSE indica que, em média, as previsões do modelo encontram-se muito próximas dos valores calculados pelo simulador, sendo penalizados de forma mais acentuada os desvios de maior magnitude. Por outro lado, o valor que o modelo calculou para o MAE, que em termos práticos traduz o desvio médio absoluto entre os valores previstos e os calculados pelo simulador, verifica a precisão do modelo, ao apresentar um valor baixo e de grandeza similar ao MSE (o que é relevante para perceber se o modelo, de facto, consegue aprender e se está apto a generalizar soluções para dados desconhecidos).

Estes indicadores, analisados em conjunto, demonstram que a abordagem adotada foi consideravelmente eficaz na validação dos dados e que o modelo treinado é promissor quanto à capacidade de prever soluções (s0) praticamente idênticas às soluções calculadas pelo simulador. No entanto, apesar dos resultados globais serem positivos, a análise detalhada dos erros individuais poderá evidenciar situações específicas em que o modelo apresenta maiores dificuldades, sendo fundamental identificar e compreender essas limitações para orientar futuros desenvolvimentos e otimizações do sistema.

4.1.2. Análise comparativa entre as soluções calculadas pelo Simulador de Cálculo e as Soluções Previstas pelo modelo de ML

Depois de armazenada a matriz de pesos, o programa P1 é encerrado, iniciando em sequência o programa P2, um modelo de previsão em ML que, pelos motivos expressos anteriormente, consegue, perante o input de dados novos, prever soluções. Reconhece-se, no entanto, que a base de dados é limitada, o que levou à utilização dos mesmos dados para treino, validação e também para a averiguação da capacidade preditiva do modelo. Esta situação, não cria nenhuma dificuldade especial porque, mesmo para situações das quais já conheceu tanto as variáveis de entrada como de saída, o P2 infere soluções com base nas variáveis de entrada e nos parâmetros aprendidos durante o treino em P1.

É com este pressuposto que será feita agora uma análise comparativa entre as soluções que existiam de início na Base de Dados- e que são as calculadas pelo simulador de cálculo - e as soluções que o modelo de ML previu. A comparação é realizável a partir de 5 tabelas em EXCEL, extraídas diretamente do programa P2, sendo elas as seguintes:

- comparacao_solucoes_previsto_real_
- distribuicao_erro_percentual_intervalos
- analise_erro_percentual_comprimentos
- scatter_plot_reais_vs_previstos
- distribuicao_espacial_erros_percentuais

A seleção destas 5 tabelas permite conceber uma perceção geral sobre a capacidade do problema em generalizar soluções em contextos diferentes. De forma geral, os resultados apresentados comprovam que o modelo consegue generalizar soluções similares às soluções calculadas pelo simulador.

4.1.2.1. Comparação de Soluções Previstas e Reais

Este arquivo Excel contém uma análise abrangente das 81 soluções diferentes e compara os valores previstos com os valores reais. O documento representa um estudo comparativo de desempenho do modelo.

O arquivo está organizado em 82 folhas: a primeira, informativa, e as 81 restantes contendo os dados das soluções. Cada folha referente a uma solução contém exatamente 17 linhas de dados, estruturados em 5 colunas: identificador do segmento, valor previsto, valor real, erro absoluto e erro percentual. A estrutura é consistente em todas as soluções, o que permite comparações diretas entre elas.

Cada uma das 81 soluções (Solucao_1 até Solucao_81) representam uma solução individual com a mesma estrutura dos dados. Os dados são categorizados em três tipos principais de segmentos:

- 8 Segmentos principais (segment_1 a segment_8): Representam as predições principais do modelo.
- 8 Segmentos sobrepostos (overlapping_1 a overlapping_8): Representam áreas de sobreposição.
- 1 Segmento de sobreposição geral (OverLapSegment): Comprimento de sobreposição.

O desempenho das soluções varia significativamente entre elas. Para os segmentos principais, o erro percentual médio varia de 0,84% (melhor solução) até 18,81% (pior solução), com uma média geral de 2,99%.

Dado o volume de dados utilizados para proceder ao treino da Rede Neuronal, os resultados são claramente positivos, existindo, no entanto, espaço para melhorias.

4.1.2.2. Distribuição do Erro percentual por Intervalos percentuais do Erro

O ficheiro Excel anexo reúne a base de dados detalhada utilizada para avaliar a capacidade de predição do modelo. Estão registados 648 dados individuais, que refletem a relação entre o comprimento de segmentos e o respetivo erro percentual, quando comparados à solução proveniente do simulador. A análise destes dados permite identificar tendências relevantes na distribuição dos erros, essenciais para avaliar a precisão e robustez do modelo.

A estrutura do ficheiro organiza-se em três folhas principais:

Folha "Distribuição" que exibe um resumo estatístico, agrupando os registos em intervalos de erro percentual. Ao todo são apresentados 11 intervalos, que vão desde os valores mais baixos (0-10%) até intervalos superiores a 60%. Esta folha fornece uma visão clara sobre a frequência relativa de cada intervalo do erro percentual e destaca a predominância de erros baixos na maioria dos casos avaliados.

Folha "Dados_Completos" que compila os dados brutos referentes aos 648 segmentos, com registo de três parâmetros: comprimento, erro percentual e categorização do erro no intervalo correspondente. Esta folha oferece acesso pormenorizado para análise aprofundada das variações segmentares no desempenho do modelo.

Folha "Grafico" que apresenta um histograma que permite a visualização da distribuição dos erros percentuais dos valores previstos. O eixo horizontal representa os intervalos de erro percentual, enquanto o eixo vertical indica a quantidade de valores registados em cada intervalo. A configuração gráfica permite verificar a forte concentração de registos nos intervalos de menor erro (sobretudo 0-10%), evidenciando o grau de confiança e a eficácia do sistema de predição desenvolvido.

A conjugação do resumo descrito nas tabelas com a representação gráfica facilita a identificação rápida de padrões, proporcionando uma avaliação objetiva sobre a precisão do modelo e permitindo detetar eventuais exceções ou desvios relevantes na base analisada.

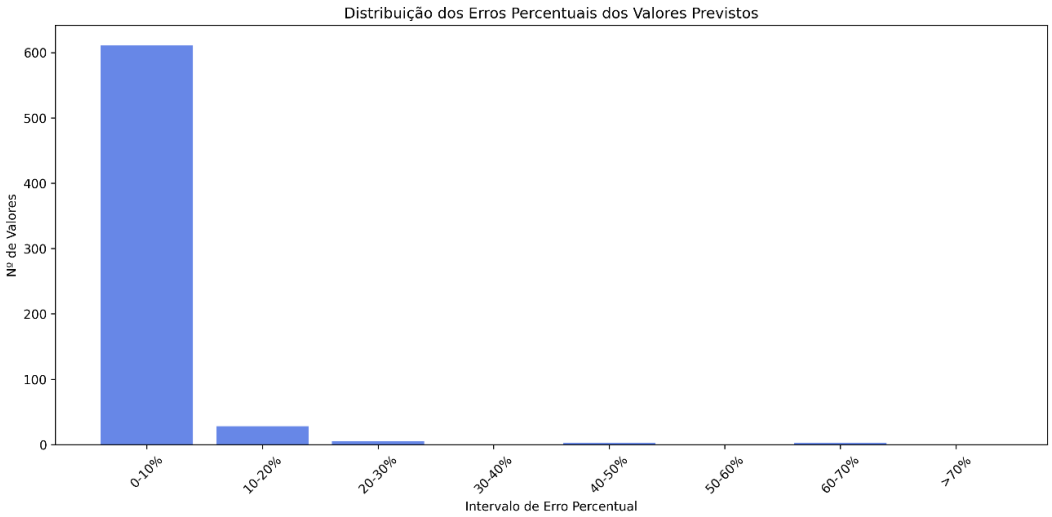


Figura 23 - Distribuição dos Erros Percentuais dos Valores Previstos

A distribuição dos erros percentuais apresenta uma acentuada assimetria positiva, característica típica de modelos com desempenho globalmente satisfatório. De acordo com os dados, cerca de 94,3% dos segmentos previstos apresentam erros inferiores a 10%, enquanto apenas 5,7% dos casos excedem esse patamar. Essa elevada concentração de valores com erro reduzido indica que o modelo de ML possui alta precisão para a esmagadora maioria das situações analisadas.

A análise detalhada, baseada nos intervalos do ficheiro extraído para Excel, revela que:

- O intervalo com maior frequência é o de 0-10% de erro percentual, que representa 611 registos de 648 (94,3% do total).
- O segundo maior grupo corresponde ao intervalo de 10-20%, com 28 de 648 registos (4,3% do total).
- Os intervalos superiores concentram proporções residuais (inferiores a 1% em todos os casos).

Esta distribuição demonstra, assim, que praticamente todos os resultados previstos pelo modelo apresentam erro percentual inferior a 10%, comprovando a eficácia do sistema preditivo.

É de salientar que a existência de alguns valores acima de 10% demonstra que, em situações específicas, o modelo pode não generalizar com total exatidão. No entanto, tendo em conta a amostra utilizada para o treino (81 casos, dos quais 76 para treino e 5 para validação), os resultados obtidos são manifestamente positivos para a base de dados disponível.

4.1.2.3. Análise do Erro percentual em função do Comprimento do Vão

O ficheiro Excel "analise_erro_percentual_comprimentos.xlsx" está organizado em quatro folhas distintas, cada uma com propósitos específicos na análise metrológica. A primeira folha, denominada "Dados", além de um resumo dos erros médios percentuais organizados por intervalos de comprimento. Esta folha apresenta seis intervalos principais: 0.1-1.7m (6.53%), 1.7-3.4m (1.22%), 3.4-5.0m (1.12%), 5.0-6.7m (1.57%), 6.7-8.3m (1.78%), e 8.3-10.0m (1.71%), o que evidencia uma clara tendência de redução do erro com o aumento do comprimento.

É ainda exposto um gráfico de dispersão clássico, onde é possível visualizar, de forma clara, a relação inversa entre comprimento dos segmentos e erro percentual.

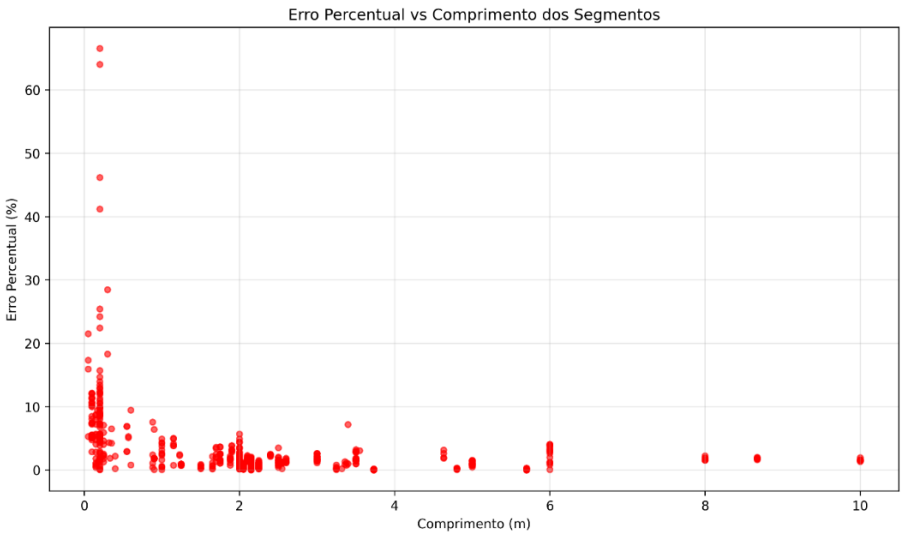


Figura 24 - Gráfico de Dispersão Erro Percentual em função do Comprimento dos Segmentos

O gráfico mostra claramente a concentração de pontos com erro elevado (superior a 15%) na região de comprimentos inferiores a 1 m, enquanto, para segmentos que cobrem vãos maiores, o erro é consistentemente mais baixo. A dispersão dos pontos diminui significativamente à medida que o comprimento aumenta, indicando maior precisão e repetibilidade nas medições de objetos maiores.

Para comprimentos inferiores a 0.5 m, o erro médio de 8.646%, com desvio-padrão de 9.478%, indica deficiências significativas tanto na veracidade como na precisão. Em contraste, para segmentos que cobrem vãos de comprimento superior a 2.0 m , o erro médio é inferior a 1.6%, com o desvio-padrão controlado, o que demonstra grande capacidade de generalização por parte do modelo para segmentos superiores a 2 m.

O erro percentual varia drasticamente de 0,003% a 66,567%, com média global de 2,993% e desvio-padrão de 5,479%. O valor da correlação negativa ser moderado (-0,332) entre o comprimento e o erro percentual confirma estatisticamente a tendência observacional de que vãos maiores, com maiores comprimentos de segmentos de perfil do tipo Madre 100x1,5, apresentam menor erro relativo.

4.1.2.4. Precisão das Estimativas: Alinhamento entre Medidas Reais e Previstas

O arquivo Excel "scatter_plot_reais_vs_previstos.xlsx" apresenta uma análise abrangente da validação do modelo P2 de generalização. O coeficiente de correlação de Pearson obtido (r = 0,9997) indica uma relação linear quase perfeita entre os valores observados e os previstos, classificando-se como correlação muito forte segundo critérios já estabelecidos. O coeficiente de determinação (R^2 = 0,9994) demonstra que o modelo explica 99,94% da variabilidade dos dados, valor que supera significativamente os padrões de excelência típicos em modelos estatísticos.

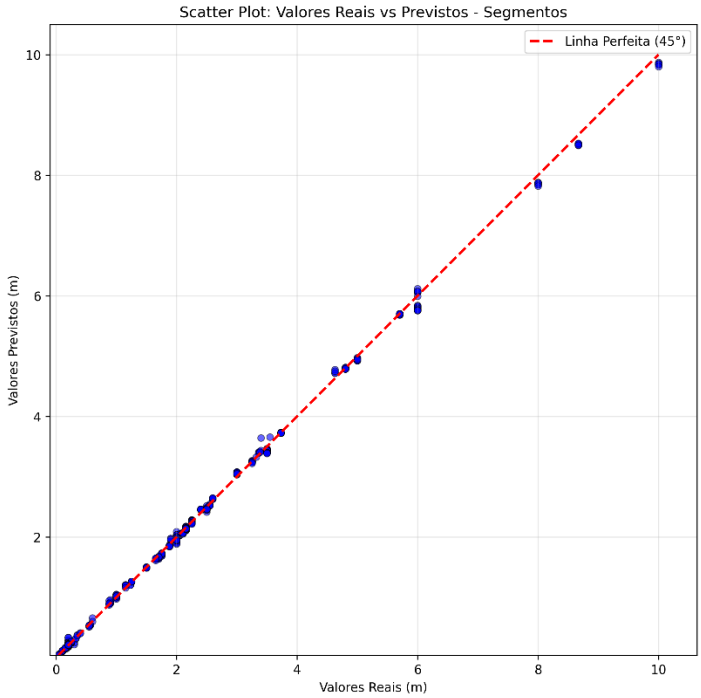


Figura 25 - Gráfico Scatter Plot de comparação entre os valores Reais e Previstos

O gráfico de dispersão apresentado é de visualização fundamental para avaliar a validação de modelos preditivos. Oferece insights visuais imediatos sobre a qualidade e confiabilidade do ajuste estatístico. Esta representação gráfica dos valores reais versus previstos permite uma avaliação direta da capacidade preditiva do modelo através da proximidade dos pontos à linha de referência diagonal.

A imagem apresenta um scatter plot clássico, com valores reais no eixo horizontal (0 a 10 metros) e valores previstos no eixo vertical (0 a 10 metros). A linha diagonal vermelha tracejada representa a "linha perfeita" de 45 graus, onde valores reais e previstos seriam idênticos, constituindo o referencial ideal para avaliação da precisão preditiva. Os pontos azuis distribuem-se de forma notavelmente próxima a esta linha de referência, indicando alto grau de concordância entre observações e predições. A observação mais notável é a extraordinária proximidade dos pontos à linha diagonal, característica que indica correlação quase perfeita entre valores observados e previstos.

Esta avaliação, a par da avaliação do erro percentual, é fundamental para perceber de forma clara como o erro percentual se traduz na proximidade ou não aos valores reais. O erro percentual, apesar de significativo, é recorrente em segmentos diminutos, o que se traduz em diferenças relativamente pequenas no erro absoluto; daí, e não obstante a existência de erros percentuais significativos, praticamente todas as soluções previstas são idênticas às soluções reais obtidas pela execução do simulador de cálculo.

4.1.2.5. Distribuição Espacial dos Erros Percentuais

O arquivo Excel apresenta uma análise abrangente da distribuição espacial de erros percentuais em oito segmentos distintos. O conjunto de dados analisado demonstra um erro percentual médio geral de 2,99%, com desvio padrão entre posições de 3,06%. Esta métrica aparentemente moderada esconde uma distribuição crítica, onde dois segmentos apresentam performance drasticamente inferior aos demais no que toca ao erro percentual.

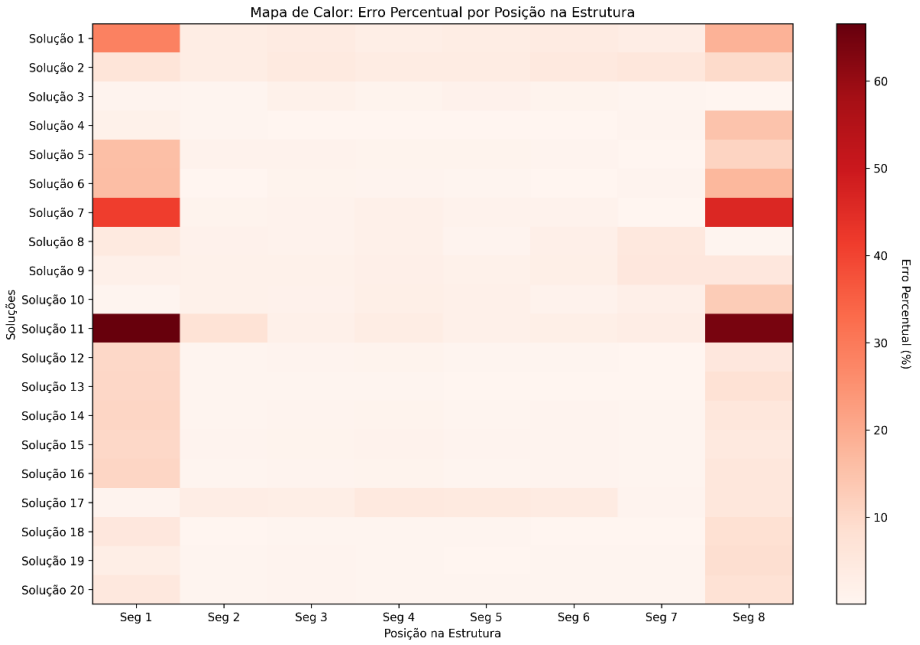


Figura 26 - Mapa de Calor: Erro Percentual por Posição na Estrutura

Como é possível observar pelo mapa exposto imediatamente acima, para as primeiras 20 soluções, o erro percentual é crítico nos segmentos 1 e 8, ou seja, nos segmentos de menor comprimento e de extremidade.

Isto adiciona informação relevante ao que já está exposto nos subcapítulos anteriores, que é o facto de o modelo apresentar dificuldade para representar adequadamente as condições físicas e numéricas nas extremidades estruturais, manifestando problemas de convergência e discretização inadequada que comprometem a precisão em segmentos de menor comprimento.

A análise do mapa de calor confirma que os segmentos 1 e 8 apresentam erros percentuais sistematicamente elevados em comparação com os segmentos intermediários. Esta distribuição não é aleatória, mas reflete limitações estruturais do método na predição de comprimentos de segmentos nos vãos mais pequenos das extremidades da estrutura.

No entanto, os erros percentuais extremamente diminutos nos comprimentos dos segmentos a utilizar para cobrir os vãos não extremos são um facto extremamente positivo e importante na avaliação do modelo de generalização.

4.1.2.6. Avaliação da Capacidade do modelo na Previsão dos valores das restantes variáveis-alvo

No contexto da presente discussão, a avaliação da capacidade do modelo na previsão dos valores das restantes variáveis-alvo não é relevante pelo facto de, em perfis do tipo Madre 100x1,5, o valor da sobreposição ser sempre igual a 1, o que significa que o respetivo comprimento de sobreposição será sempre 0. O modelo está capacitado a prever tanto o número de sobreposições em cada nó, como o respetivo comprimento de sobreposição; no entanto, para os dados utilizados na fase de treino, a não existência de outro tipo de casos, em que existem sobreposições nos apoios, limita a análise dos valores previstos pelo modelo construído.

Esta ausência de diversidade nos dados de treino compromete a avaliação robusta da capacidade do modelo em prever situações fora do padrão dominante, uma vez que não existem exemplos suficientes de cenários alternativos para validar a capacidade de generalização do modelo. O desempenho do modelo é satisfatório nos casos predominantes, com erros absolutos médios baixos, mas a falta de casos com sobreposições variadas ou comprimentos de sobreposição não nulos impede uma análise abrangente da robustez do modelo em situações heterogéneas. Assim, a limitação do conjunto de treino inviabiliza uma avaliação crítica da capacidade preditiva do modelo para as variáveis do número de sobreposições em cada nó da estrutura e do comprimento dessa sobreposição.

De qualquer forma, é de notar que os resultados para as medições com valor real igual a 1,0, o modelo demonstra alta precisão com erro absoluto médio de 0,006736. O desvio padrão de 0,006338 sugere consistência na performance preditiva. O valor baixo para o erro percentual confirma a alta qualidade das predições do modelo nos casos predominantes.

A variável “OverLapSegment” apresenta um comportamento similar à variável overlapping. Todos os 81 valores reais são iguais a 0, confirmando que o comprimento de sobreposição é sempre nulo nos perfis analisados. Esta uniformidade impossibilita o cálculo do erro percentual, uma vez que envolveria divisão por zero.

O erro absoluto médio para OverLapSegment é de 0,005813, ligeiramente inferior ao erro das variáveis overlapping. O desvio padrão (0,004714) indica boa consistência preditiva. O erro máximo observado foi de 0,028893, significativamente menor que os outliers das variáveis overlapping.

Os valores previstos pelo modelo para OverLapSegment variam entre -0,020969 e 0,028893, com média de -0,004249. Esta ligeira tendência negativa sugere que o modelo tende a prever pequenos valores negativos quando o valor real é zero, o que não deveria acontecer, dado que não é possível, do ponto de vista físico, existir comprimentos negativos. Mas, mais uma vez, a não existência na base de dados de outro tipo de situações onde é necessária a existência de sobreposições nos apoios não permite julgar de forma assertiva a capacidade preditiva do modelo.

Todos os ficheiros utilizados na análise de resultados encontram-se disponibilizados numa unidade de armazenamento online, acessível através do seguinte link: https://drive.google.com/drive/folders/1l-m3hKHBxXwh3WWXSzAh4AQ4gaKn9BMo?usp=drive_link

5
CONCLUSÃO E TRABALHO FUTURO

A presente Dissertação teve como objetivo o desenvolvimento de um modelo em ML que fosse capaz de acelerar o processo de dimensionamento otimizado de perfis metálicos em coberturas e fachadas.

Através do acesso a uma base de dados, foi possível desenvolver um modelo de ML funcional, em que os valores que prevê se aproximam, de forma muito significativa, dos valores calculados por um simulador de cálculo. Apesar da existência de outliers, e de os fenómenos de relação entre as variáveis serem complexos e algo difíceis de serem compreendidos, os resultados obtidos provam que a utilização da AI é altamente eficaz para acelerar processos de dimensionamento estrutural.

A velocidade no processo de cálculo, a capacidade de lidar com grandes volumes de dados, e a capacidade de prever relações entre variáveis fazem da AI uma das áreas em expansão que vai alterar o futuro da Engenharia Civil.

Perspetivas Futuras:

Tendo em vista a evolução e o aprimoramento da ferramenta proposta, identificam-se os seguintes vetores de desenvolvimento:

1) Aprofundamento do modelo de redes neuronais, nomeadamente, a construção de arquiteturas distintas, e averiguar as diferenças nos resultados obtidos, para estudar a possibilidade de alteração da arquitetura no modelo de ML original e que resulta da conclusão da presente dissertação. Em especial, percecionar se os resultados do comprimento dos segmentos do perfil em vãos extremos de comprimento reduzido, se apresentam com erros relativos menores.

2) Testar a possibilidade de utilização de outros algoritmos (Ex: kNN, Random Forest, Boosting Algorithms etc.), e comparar os resultados para que seja possível percecionar qual dos modelos conduz a soluções mais próximas da solução ótima.

3) Aumentar a base de dados e criar, no modelo, um sistema que, a cada introdução de novas variáveis e soluções resultantes do cálculo pelo simulador, atualize automaticamente a matriz de pesos.

4) Tentar reaplicar o modelo para o perfil do tipo Omega e para as restantes secções do perfil do tipo Madre e Omega.

5) Analisar os resultados generalizados por parte do modelo para o número de sobreposições por nó e o comprimento de sobreposições, através do recurso a outros perfis e secções onde a sobreposição nos apoios é relevante, porque não foi possível fazê-lo.

6) Tentar aprimorar o modelo, com vista a conseguir conjugar mais do que um modelo de regressão, ou seja, um modelo que englobe e que calcule soluções para os diferentes perfis e secções (o treino para cada tipo de perfil e secção continua a ser autónomo).

7) Aumentar o número de variáveis de input a utilizar para a fase de treino, em especial a variável "internal", nas situações em que esta seja relevante.

8) Procurar resolver os outliers que surgem nas generalizações do programa P2.

9) Averiguar a possibilidade de inclusão num servidor onde esteja inserido o simulador de cálculo, da rede neuronal construída e produzir, na sequência, um estudo sobre os ganhos temporais e financeiros, derivados à potencialidade e velocidade dos processos de cálculo em modelos de AI e à economização da utilização do CPU, respetivamente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Zhang, S. *Artificial Intelligence and Applications in Structural and Material Engineering.* Higthligths in Science, Engineering and Technolgy, British Columbia, 2023.

[2] Pinhão, N. *Artificial Intelligence for the Analysis of Structures in Civil Engineering*, Dissertação de Mestrado, Instituto Superior Técnico de Lisboa, 2021.

[3] Soares, F. *O Estado da Arte da Aplicação da Inteligência Artificial (IA) e Aprendizagem Automática (AA) ao Cálculo de Estruturas*. Dissertação de Mestrado, Faculdade de Engenharia da Universidade do Porto, 2024.

[4] Marques, L., Rebelo, C. Silva, L. *Métodos Avançados de Análise por Elementos Finitos para Verificação da Estabilidade de Estruturas Metálicas.* VII Congresso de Construção Metálica, 19 e 20 / 11 /2009, ISISE, Departamento de Engenharia Civil, Universidade de Coimbra, 6 a 8, Research Gate, Coimbra.

[5] Martinez, R. *Artificial Intelligence: Distinguish Between Types&Definitions* .2019. https://heinonline.org/HOL/Page?handle=hein.journals/nevlj19&id=1015&collection=journals&index=.21/03/2025.

[6] Gomes, D. *Inteligência Artificial: Conceitos e Aplicações*. Revista Olhar Científico – Faculdades Associadas de Ariquemes, Ariquemes, 2010.

[7] Jaboob, A., Durrah, O., Chakir, A. traditional goals of AI research. In *Artificial Intelligence: An Overview*, 115, Springer Nature, 2024.

[8] Abubakar, S., Karimi., M., M., S, A., B. *Structural Engineering Applications Using Artificial Intelligence and Machine Learning: A Review*, International Journal of Advanced Natural Sciences and Engineering Researches, 27/06/2024, Istambul.

[9] Marrone, R., Cropley, D., Medeiros, K.. *How Does Narrow AI Impact Human Creativity?*, Revista Creativity Research Journal, Philadelphia, Pennsylvania, 15/07/2024.

[10] Jaboob, A., Durrah, O., Chakir, A. AI categories. In *Artificial Intelligence: An Overview*, 209/215, Springer Nature, 2024.

[11] Jaboob, A., Durrah, O., Chakir, A. categories based on functionality. In *Artificial Intelligence: An Overview*, pp.223-234, Springer Nature, 2024.

[12] Mahesh, B. *Machine Learning Algorithms – A Review*, International Journal of Science and Research, Bangalore, 2019.

[13] Thatavarthi, N. *Developing AI and ML Solutions with ML.Net*, Journal of Artificial Intelligence Cloud Computing, Wichita, 2022.

[14] Gollapudi, S., introduction to machine learning. In *Practical Machine Learning*, 3, PACKT Publishing, 2016.

[15] Magalhães, C. *Dimensionamento de ligações metálicas suportado por técnicas de Machine Learning*. Dissertação de Mestrado, Faculdade de Engenharia da Universidade do Porto, 2023.

[16] Gollapudi, S., introduction to machine learning. In *Practical Machine Learning*, pp. 4-9, PACKT Publishing, 2016.

[17] Brunel, A., Mazza, D., Pagani, D. In *Backpropagation in the Simply Typed Lambda-Calculus with Linear Negation*, 64:1, ACM Digital Library, 2020.

[18] Yao, H. *Discussion on Classification problems in machine learning*. 2nd International Workshop on Materials Engineering and Computer Sciences (IWMECS 2015), 10 e 11 / 10 /2015, Jinai, Shandong, Atlatins Press, Zhengzhou.

[19] Kumar, S. Bhatnagar, V. *A Review of Regression Models in Machnine Learning*. Journal of Intelligent Systems and Computing, Ariquemes, Jaipur, 31/12/2022.

[20] Thai, H. *Machine learning for structural engineering: A state-of-the-art review*. ELSEVIER, Parkville, 2022.

[21] Oliveira, A. *Learning the Optimal Representation Dimension for Restricted Boltzmann Machines*. Association for Computing Machinery, New York, 2023.

[22] Ramtin, S., how to choose appropriate bivariate test. In *Translational Sports Medicine*, 145, Academic Press, 2023.

[23] Islam, K. Shapla, T. *On Invariance of Chi-squared Tests Under Different Probability Models*. Internationl Journal of Statistics and Probability, 4/12/2024, 8, Canadian Center of Science and Education, Ontario.

[24] Harrell, F. E., Lee, K. L., Mark, D. B. *Multivariable Prognostic Models: Issues in Developing Models, Evaluating Assumptions and Adequacy, and Measuring and Reducing Errors*. Statistics in Medicine, 29/02/1996, pp. 361–387, Wiley, Chichester.

[25] Alexopoulos, E. C. *Introduction to Multivariate Regression Analysis*. Hippokratia, 01/12/2010, pp. 23–28, Hippokratia, Tessalónica.

[26] Mijwel, M. M., Esen, A., Shamil, A. *Overview of Neural Networks*. Journal of Artificial Intelligence Research, 15/03/2021, pp. 45–56, Elsevier, Amsterdão.

[27] Liu, Y., Wang, Y., Zhang, J. *New Machine Learning Algorithm: Random Forest*. Proceedings of the International Conference on Information Computing and Applications (B. Liu, M. Ma, J. Chang, eds.), 14–16 Setembro 2012, Tangshan, China, pp. 246–252, Springer-Verlag, Berlin Heidelberg.

[28] Breiman, L. *Bagging predictors*. Machine Learning, 1996, pp. 123–140, Kluwer Academic Publishers, Dordrecht.

[29] Matzavela, V., Alepis, E*. Decision tree learning through a Predictive Model for Student Academic Performance in Intelligent M-Learning environments*. Computers and Education: Artificial Intelligence, 5/10/2021, pp. 100035, Elsevier, Piraeus.

[30] Raschka, S., Mirjalili, V. a tour of machine learning classifiers using scikit-learn. In *Python Machine Learning*, pp. 51-105, Packt Publishing, Birmingham, 2017.

[31] Ferreira, A. J., Figueiredo, M. A. T. boosting algorithms: a review of methods, theory, and applications. In *Ensemble Machine Learning*, pp. 35–85, Springer, New York, 2012.

[32] Noble, William S. *What is a support vector machine?* Nature Biotechnology, 2006, v. 24, n. 12, p. 1565-1567. Nature Publishing Group, New York.

[33] Kramer, O. k-nearest neighbors. In *Dimensionality Reduction with Unsupervised Nearest Neighbors*, p. 27 -37, Springer, Berlim, 2013.

[34] Rasamoelina, A. D., Adjailia, F., Sincak, P. *A Review of Activation Function for Artificial Neural Network*. In: 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informat-

ics (SAMI) (Sincak, P., ed.), 23–25 January 2020, Herl'any, Slovakia, pp. 1-6, IEEE, Piscataway, NJ.

[35] Naidu, G., Zuva, T., Sibanda, E. M. *a review of evaluation metrics in machine learning algorithms*. In Lecture Notes in Networks and Systems, pp. 15–25, Springer Nature, Vanderbijlpark, 2023.

[36] Bottou, L. stochastic gradient descent tricks. In *Lecture Notes in Computer Science*, vol. 7700, pp. 421–436, Springer, Berlin, 2012.

[37] Duchi, J., Hazan, E., Singer, Y. *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*. Journal of Machine Learning Research, 12 (2011), pp. 2121–2159, Microtome Publishing, Boston.

[38] Zeiler, M. D. *Adadelta: an adaptive learning rate method*. arXiv preprint, 13/12/2012, pp. 1–8, arXiv.org e-print archive, Ithaca, NY.

[39] Saqib, M., Iqbal, M., Othman, M. T. B., Shahazad, T., Ghadi, Y. Y., Al-Amro, S., et al. (2024). *Lumpy skin disease diagnosis in cattle: A deep learning approach optimized with RMSProp and MobileNetV2*. PLoS ONE, 19/08/2024, e0302862.

[40] Bhattacharjee, A., Popov, A. A., Sarshar, A., Sandu, A. *Improving the Adaptive Moment Estimation (ADAM) stochastic optimizer through an Implicit-Explicit (IMEX) time-integration approach,* CSL-TR-2024-2, 14/09/2024, pp. 1-23 Cornell University, Ithaca.

[41] Kingma, D. P., Ba, J. *ADAM: A Method for Stochastic Optimization*, 30/01/2017, Cornell University, Ithaca, NY.

[42] Khan, M., Arora, A. introduction to pandas. In *Essentials of Python for Artificial Intelligence and Machine Learning*, pp. 161–196, Springer, Singapura, 2024.

[43] Koch, P., Bosch, J. *A Diagnostic Approach to Assess the Quality of Data Splitting in Machine Learning*. Journal of Systems and Software, 06/2022, pp. 110944, Elsevier, Amsterdam.

[44] Starovoitov, V., Golub, M. *Data Normalization in Machine Learning*. International Journal of Computer Science and Network Security, 01/10/2021, pp. 30–35, IJCSNS, Seoul.

[45] Amorim, L. B. V., Cavalcanti, G. D. C., Cruz, R. M. O. *The choice of scaling technique matters for classification performance*. Applied Soft Computing Journal, 15/12/2022, p. 109924, Elsevier, Amsterdam.

[46] Abadi, M., Barham, P., Chen, J., et al. *TensorFlow: A System for Large-Scale Machine Learning*. In: Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (Kim, M., ed.), 2–4 de Novembro de 2016, Savannah, GA, pp. 265–283, USENIX Association, Berkeley, CA.

[47] Chollet, F. deep learning for computer vision. In *Deep Learning with Python*, pp. 135–182, Manning Publications, Shelter Island, NY, 2017.

[48] Goodfellow, I., Bengio, Y., courville, A. deep feedforward networks. In *Deep Learning*, pp. 168–226, MIT Press, Cambridge, MA, 2016.

[49] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. *Array programming with NumPy.* Nature, 28/07/2020, pp. 357–362, Nature Publishing Group, London.