



**Pedro Alexandre
Cabral de Carvalho
Winkel Martins**

**Desenvolvimento de um sistema de controlo
inteligente do processo de soldadura por
transmissão de laser em termoplásticos**

Development of an intelligent control system applied to the
process of laser transmission welding for thermoplastics



**Pedro Alexandre
Cabral de Carvalho
Winkel Martins**

**Desenvolvimento de um sistema de controlo
inteligente do processo de soldadura por
transmissão de laser em termoplásticos**

Development of an intelligent control system applied to the process of laser transmission welding for thermoplastics

Relatório de Projeto apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizado sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar, do Departamento de Engenharia Mecânica da Universidade de Aveiro, e de Mihail Fontul, Gestor do Departamento de Inovação, Pesquisa e Desenvolvimento da Iber-Oleff.

Este projeto foi realizado dentro do contexto do projeto S4Plast (Sustainable Plastics Advanced Solutions), financiado através do Sistema de Incentivos à Investigação e Desenvolvimento Tecnológico do Programa Portugal 2020 e teve o apoio dos projetos UIDB/00481/2020 e UIDP/00481/2020 - Fundação para a Ciência e a Tecnologia; e CENTRO-01-0145 FEDER-022083 - Programa Operacional Regional do Centro (Centro2020), através do Portugal 2020 e do Fundo Europeu de Desenvolvimento Regional.

O júri / The jury

Presidente / President

Prof. Doutor António Manuel de Bastos Pereira
Professor Associado com Agregação da Universidade de Aveiro

Vogais / Committee

Prof. Doutor Eugénio Alexandre Miguel Rocha
Professor Associado da Universidade de Aveiro

Prof. Doutor José Paulo Oliveira Santos
Professor Auxiliar da Universidade de Aveiro (orientador)

Agradecimentos / Acknowledgements

Gostaria de deixar o meu agradecimento em primeiro lugar aos meus pais, por todo o apoio e oportunidades que me disponibilizaram durante a minha vida. Espero que este documento e todo o meu percurso académico tenha na vossa mente justificado o esforço. Gostaria de agradecer à minha restante família pelo constante apoio, disponibilidade e carinho.

Deixo também o meu agradecimento às pessoas que caminharam comigo durante estes cinco anos como colegas de curso e amigos. Espero que todos tenham sucesso e encontrem alegria nos próximos passos da vida.

Por fim, agradeço ao professor José Paulo e ao doutor Mihail pelo apoio na realização deste trabalho. Gostaria de agradecer também aos professores Pedro Prates e Eugénio Rocha e ao Pedro Nunes pelas contribuições para este projeto. Fico muito agradecido pelo tempo disponibilizado e pelos conselhos dados.

Dedico este documento aos meus avós.

Keywords

Welding, Thermoplastics, Machine Learning, Parameter Control, Internet of Things

Abstract

The plastic transforming industry is already very well automated in most of its processes, but this potential flexibility does not materialize in the agility that is required by today's new industrial reality that asks for more speed and better quality in the production of new parts. In Iber-Oleff, for instance, when it comes to the laser transmission welding process there are unsolved problems related to the inflexible and lingering nature of the definition of process parameters. Both these problems are responsible for the existence of downtimes in production and also the production of defective parts. The final part defects are perceived through some destructive experiments, using the separation force between the welded parts (rear and front frame) as a metric. These problems contribute to the reduction of the company's income and to the degradation of the environment, when considering the production of defective parts. Therefore it is critical to have an intelligent control system which can recognize the product that is pretended to be produced and quickly communicate the process parameters to the laser welding equipment. In order to achieve this, not only must instant access to the standard production parameters be achieved, once the part to be produced is known, but there is also a need to create a prediction on the separation force between parts united with those parameters, which can help to avoid part defects. The machine operator must be able to interact with the system through a graphical interface in order to critically evaluate the predictions made by the predictive algorithm and alter the key parameters if seen fit. A REST API ought to be built as a way to allow interactions with the database that contains the informations required by the system. During the development of these different modules it is important to guarantee their agility and accessibility, which can be achieved by using open-source programming environments that can be easily interpreted and integrated. In the proposed solution the different modules were developed in C#, Python and Node-Red, while devices like the ESP8266 and Raspberry Pi 4 were used for the reading of the parts' barcodes and for the integration of the system in an industrial environment and communication of the process parameters to the laser equipment, respectively. It is also important to guarantee the accuracy of the predictions made by the predictive algorithm. Two distinct algorithms were tested: artificial neural networks and XGBoost. Both were subject to hyperparameter optimization through the Taguchi method. The best algorithm was an artificial neural network with an accuracy of 93%, relating to the true value of the separation force. This algorithm was developed using the PyTorch framework. This system presents an objective competitive edge in a highly demanding market, with the reduction of downtimes in production and the reduction of material waste.

Palavras-chave

Soldadura, Termoplásticos, Machine Learning, Controlo de Parâmetros, Internet das Coisas

Resumo

A indústria transformadora de plásticos é já fortemente automatizada, no entanto, essa flexibilidade potencial não se materializa em agilidade, pelo menos na dimensão exigida pelos desafios da nova realidade industrial que exige maior celeridade na produção das peças e maior qualidade. Na Iber-Oleff, por exemplo, no que diz respeito ao processo de soldadura por transmissão de laser existem problemas relacionados com a parametrização do equipamento pela sua inflexibilidade e demora excessiva. Estes problemas são responsáveis pela criação de tempos mortos na produção de peças poliméricas e pela obtenção de peças defeituosas. Os defeitos nas peças finais são percecionados através de ensaios destrutivos pontuais, sendo a força de separação das duas partes soldadas (rear e front frame) a métrica usada para definir a existência ou não de defeitos de soldadura. Ambos os problemas enunciados contribuem para uma redução das receitas da empresa e no caso específico da obtenção de peças defeituosas, um desperdício de material que também representa um problema para o ambiente. Torna-se, portanto, crítica a existência de um sistema de controlo inteligente que consiga reconhecer o produto que se pretende produzir e rapidamente atuar os parâmetros do equipamento de soldadura. Para isso, não basta conhecer de forma imediata os parâmetros padrão de produção, uma vez conhecida a peça a produzir, mas também realizar uma previsão da força de separação das peças unidas com esses parâmetros de forma a prevenir defeitos no produto final. O operador deve poder conseguir interagir com este sistema através de uma interface para avaliar criticamente as previsões efetuadas e alterar os parâmetros chave, caso seja necessário. Por fim, deve-se garantir a interação com a base de dados que contém as informações necessárias ao desenvolvimento do sistema através de uma REST API. Durante estes desenvolvimentos, é importante garantir a agilidade do sistema de produção e a sua acessibilidade, devendo procurar-se, que todos os programas desenvolvidos usem ambientes de programação open-source de fácil perceção e integração. Na solução proposta os vários módulos foram desenvolvidos em C#, Python e Node-Red, sendo utilizados dispositivos como o ESP8266 para possibilitar a leitura do código de barras de cada peça e o Raspberry Pi 4 para permitir a integração do sistema num ambiente fabril e comunicação dos parâmetros para o autómato associado ao equipamento. Também foi crucial garantir a exatidão das previsões do algoritmo preditivo. Foram testados dois algoritmos distintos: redes neuronais artificiais e XGBoost. Ambos foram alvo de uma otimização de hiperparâmetros realizada através do método de Taguchi, sendo que no final deste processo, o algoritmo mais exato consistia numa rede neuronal artificial com uma exatidão de aproximadamente 93% em relação ao valor verdadeiro da força de separação. Esta foi desenvolvida com recurso à framework PyTorch. Este sistema permite a obtenção de uma vantagem competitiva num mercado altamente exigente, como o da transformação de plásticos, com a redução dos tempos mortos na produção e com a redução do desperdício de matéria-prima.

Índice

1	Introdução	1
1.1	Enquadramento - Projeto S4PLAST	1
1.2	Iber-Oleff	2
1.2.1	Máquinas de injeção	3
1.2.2	Célula integrada de controlo de qualidade	4
1.2.3	Equipamento de soldadura por transmissão de laser	4
1.3	O problema	5
1.4	Objetivos	6
1.5	Organização do documento	6
2	Enquadramento Teórico	7
2.1	Soldadura por transmissão de laser	7
2.1.1	O processo	7
2.1.2	Parâmetros do processo	9
2.2	Conceitos relacionados com o modelo preditivo	11
2.2.1	<i>Machine learning</i>	12
2.2.2	Design experimental	23
2.3	Trabalhos relacionados	26
2.3.1	Otimização da soldadura por transmissão de laser	27
2.3.2	Otimização de outros processos produtivos	30
2.3.3	Síntese dos trabalhos relacionados	36
3	Solução proposta e Implementação	39
3.1	Base de dados com a informação relativa aos <i>front</i> e <i>rear frames</i>	42
3.2	Módulo de leitura dos números série de peças (ESP8266)	44
3.3	Servidor C# e comunicação com os vários módulos	47
3.4	Interface gráfica com o operador da máquina	49
3.5	REST API para interação com a base de dados	51
3.6	Autómato e atuação dos parâmetros do processo	53
3.7	Algoritmo preditivo da força de separação dos <i>front</i> e <i>rear frames</i>	56
3.7.1	Rede neuronal artificial	60
3.7.2	XGBoost	61
3.7.3	Implementação do algoritmo	61
3.8	Containerização dos módulos através do software Docker	61

3.9 Transporte dos módulos para um Raspberry Pi 4 através da plataforma Balena	65
4 Resultados	71
5 Conclusões	75
A Anexo 1 - Criação da base de dados e tabelas	77
B Anexo 2 - Ensaios de otimização de hiperparâmetros para o algoritmo preditivo	83
B.1 Ensaios redes neuronais artificiais	83
B.1.1 Conjunto de dados reais	83
B.1.2 Conjunto de dados mistos	92
B.2 Ensaios XGBoost	100
B.2.1 Conjunto de dados reais	100
B.2.2 Conjunto de dados mistos	108
C Anexo 3 - Tabela de dados mistos	115
Referências	120

Lista de Tabelas

3.1	Conjunto de dados recolhidos associados ao projeto PolyWeld.	56
3.2	Tabela Taguchi L27 que permitiu conduzir a otimização de hiperparâmetros.	58
3.3	Parâmetros utilizados na primeira iteração da otimização de hiperparâmetros da ANN.	60
3.4	Parâmetros utilizados na primeira iteração da otimização de hiperparâmetros do modelo XGBoost.	61
4.1	Valores de exatidão e RMSE para o melhor algoritmo desenvolvido nos 10 conjuntos de dados de treino e teste, com os seus valores médios e desvios padrão também ilustrados.	72
B.1	Ensaio conduzido com o conjunto de dados reais e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso das ANN.	84
B.2	Valores de S/N associados aos ensaios com os conjuntos de dados reais e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para as ANN.	85
B.3	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.	86
B.4	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.	86
B.5	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.	86
B.6	Ensaio conduzido com o conjunto de dados reais e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso das ANN.	88
B.7	Valores de S/N associados aos ensaios com os conjuntos de dados reais e ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para as ANN.	89
B.10	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.	89

B.8	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.	91
B.9	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN. . . .	91
B.11	Ensaio conduzido com o conjunto de dados mistos e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso das ANN.	92
B.12	Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para as ANN.	93
B.13	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.	94
B.14	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.	94
B.15	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.	95
B.16	Ensaio conduzido com o conjunto de dados mistos e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso das ANN.	97
B.17	Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para as ANN.	97
B.18	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.	98
B.19	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN. . . .	99
B.20	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.	99
B.21	Ensaio conduzido com o conjunto de dados reais e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso do XGBoost.	100
B.22	Valores de S/N associados aos ensaios com os conjuntos de dados reais e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para o XGBoost.	101
B.23	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.	102

B.24	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.	102
B.25	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.	102
B.26	Ensaos conduzidos com o conjunto de dados reais e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso do XGBoost.	104
B.27	Valores de S/N associados aos ensaios com os conjuntos de dados reais e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para o XGBoost.	105
B.28	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.	106
B.29	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.	106
B.30	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.	106
B.31	Ensaos conduzidos com o conjunto de dados mistos e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso do XGBoost.	108
B.32	Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para o XGBoost.	108
B.33	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.	109
B.34	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.	110
B.35	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.	110
B.36	Ensaos conduzidos com o conjunto de dados mistos e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso do XGBoost.	111
B.37	Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para o XGBoost.	112

B.38	Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.	112
B.39	Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost. .	113
B.40	Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.	113
C.1	Tabela de dados mistos normalizados.	115

Lista de Figuras

1.1	Chão de fábrica da Iber-Oleff.	2
1.2	Exemplares de um <i>rear frame</i> e de um <i>front frame</i>	3
1.3	Máquinas de injeção responsáveis pela produção do <i>front</i> e <i>rear frame</i>	3
1.4	Célula de controlo de qualidade responsável pela identificação de defeitos nas peças.	4
1.5	Equipamento de soldadura responsável pela união do <i>rear frame</i> e <i>front frame</i>	5
2.1	Ilustração de um processo de soldadura a laser direta [1].	8
2.2	Ilustração do processo de soldadura por transmissão de laser [1].	9
2.3	Feixe de laser Circular/Elíptico [2].	11
2.4	Processo de aprendizagem e teste de um algoritmo de classificação [3].	13
2.5	Representação gráfica de uma ANN <i>feedforward</i> (a)) e de uma ANN <i>feedback</i> (b)) [4].	14
2.6	Representação de um neurónio artificial (4) e das suas interações e comportamento.	14
2.7	Representação gráfica das funções de ativação passo (a)), linear (b)), sigmóide (c)), tangente hiperbólica (d)), ReLU (e)), ELU (f)) e <i>swish</i> (g)) [5].	15
2.8	Exemplo de uma ANN com três camadas escondidas, aplicada na maquiagem a laser de aço [6].	16
2.9	Comparação entre a otimização por <i>gradient descent</i> com o uso de um ritmo de aprendizagem baixo (a) e com o uso de um ritmo de aprendizagem alto (b) [7].	18
2.10	Exemplo de uma DT [8].	20
2.11	Efeito do <i>boosting</i> para dados de treino (em baixo) e dados de teste (em cima) [9].	22
2.12	Resultados do teste dos algoritmos baseados no <i>multi-layer perceptron</i> (à esquerda) e em ANN de funções de base radial (à direita), tanto para a elongação na rutura (em cima), como para a força de separação (em baixo) [10].	27
2.13	ANN utilizada na previsão da força de separação e largura do cordão de soldadura, com recurso à potência de laser, velocidade de laser, distância entre peças e força de compressão [11].	28
2.14	Comparação entre o valor de força de separação previsto pelo algoritmo com os valores reais [12].	29
2.15	Parâmetros relevantes para o processo de união de tubos PVC [13].	31
2.16	Níveis controlados associados a cada parâmetro relevante [13].	31

2.17	Design experimental utilizado [13].	31
2.18	Visualização gráfica do valor médio de S/N (eixo vertical) para cada nível controlado de cada parâmetro considerado (eixo horizontal) [13].	32
2.19	Resultados das experiências realizadas com o conjunto de níveis que em média maximizam o valor de S/N [13].	32
2.20	Ensaio de teste conduzidos [13].	33
2.21	Relação entre a temperatura da bucha e as variáveis de qualidade a controlar [14].	34
2.22	Resultados da aplicação do teste ANOVA [15].	34
2.23	ANN usada na previsão do desgaste da ferramenta [15].	35
2.24	Previsões do desgaste da ferramenta [15].	35
2.25	ANN usada para prever o valor de velocidade de avanço [15].	36
3.1	Esquema da solução proposta, integrada na linha de produção da Iber-Oleff.	40
3.2	Interações e trocas de informação entre os vários módulos da solução proposta.	41
3.3	Diagrama relacional entidade relacionamento da base de dados criada.	43
3.4	Leitor de código de barras utilizado no projeto.	45
3.5	Configuração do leitor de código de barras para comunicar via Rs232.	45
3.6	Esquema da ligação entre o leitor de código de barras, Raspberry Pi 4 e ESP8266.	46
3.7	Ligações físicas associadas ao ESP8266.	46
3.8	Fluxograma do funcionamento do servidor desenvolvido.	48
3.9	Mensagem de alerta da interface.	49
3.10	Página de <i>debug</i>	50
3.11	Página de parametrização.	50
3.12	Página de gráficos.	50
3.13	Fluxograma do funcionamento da interface gráfica.	51
3.14	Criação de um novo “ <i>Scaffolding item</i> ”.	52
3.15	Seleção do item.	52
3.16	Referência ao “ <i>DbContext</i> ”.	53
3.17	Interface gráfico do software Swagger e tipos de mensagens que são admitidas.	53
3.18	Exemplo do resultado de um pedido HTTP GET.	53
3.19	Endereço IP do PLC.	54
3.20	Interface entre CPU e PLC.	54
3.21	Indicação das posições do CPU.	54
3.22	<i>Data block 1</i>	54
3.23	<i>Data block 2</i>	55
3.24	Atualização da posição de memória V.	55
3.25	<i>Reset</i> do indicador de alteração de parâmetros.	55
3.26	Teste de correlação de Pearson aplicado ao conjunto de dados mistos (à esquerda) e ao conjunto de dados reais (à direita).	58
3.27	Fluxograma do funcionamento da aplicação que permite implementar o algoritmo preditivo.	62
3.28	Comparação da estruturação de máquinas virtuais com um conjunto de “ <i>containers</i> ” Docker [16].	62
3.29	Criação da “ <i>fleet</i> ” de dispositivos Raspberry Pi 4.	65

3.30	Adição de um novo dispositivo à <i>"fleet"</i>	66
3.31	Informações relativas ao dispositivo disponibilizadas através da balenaCloud.	67
3.32	<i>Login</i> na plataforma balenaCloud através da linha de comandos.	67
3.33	Comando balena <i>"push"</i> executado com sucesso.	68
3.34	Carregamento dos ficheiros no dispositivo efetuado com sucesso.	68
3.35	Mensagens recebidas no terminal que permite avaliar o funcionamento dos diferentes programas.	69
4.1	Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.	73
4.2	Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.	74
4.3	Sistema de controlo inteligente.	74
A.1	Criação de uma nova base de dados.	77
A.2	Página de configuração da nova base de dados.	78
A.3	Definição do utilizador principal da base de dados.	78
A.4	Criação de uma nova tabela na base de dados.	79
A.5	Criação de um novo <i>"Login"</i>	80
A.6	Configuração do novo <i>"Login"</i>	81
B.1	Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.	87
B.2	Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.	88
B.3	Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.	91
B.4	Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.	92
B.5	Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.	96
B.6	Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.	96
B.7	Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.	99
B.8	Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.	100
B.9	Comparação das previsões e valores reais para os conjuntos de dados de teste.	103
B.10	Comparação das previsões e valores reais para os conjuntos de dados de treino.	103
B.11	Comparação das previsões e valores reais para os conjuntos de dados de teste.	107
B.12	Comparação das previsões e valores reais para os conjuntos de dados de treino.	107
B.13	Comparação das previsões e valores reais para os conjuntos de dados de teste.	110

B.14	Comparação das previsões e valores reais para os conjuntos de dados de treino.	111
B.15	Comparação das previsões e valores reais para os conjuntos de dados de teste.	113
B.16	Comparação das previsões e valores reais para os conjuntos de dados de treino.	114

Capítulo 1

Introdução

Neste capítulo inicial será explicitado o enquadramento do projeto exposto bem como a motivação que levou ao seu desenvolvimento e os seus objetivos. Será também dada a conhecer a empresa que colaborou com o autor para concretizar o projeto, a Iber-Oleff, e o seu chão de fábrica. Será também, neste contexto, fornecida informação sobre como o processo de soldadura por transmissão de laser se encaixa na produção de peças plásticas na Iber-Oleff e serão explicitados quais os maiores problemas associados a este processo atualmente.

1.1 Enquadramento - Projeto S4PLAST

A indústria de moldes é um dos campos de indústria onde a nação à beira-mar plantada consegue uma maior expressão. O zelo pela qualidade e satisfação do cliente diferencia o produto luso daqueles que são produzidos um pouco por todo o mundo. Sabe-se, no entanto, que este é um mercado muito competitivo e que necessita de respostas rápidas e eficazes para problemas reais. É pretensão do S4Plast abordar alguns destes problemas de frente. Este é um projeto mobilizador que visa auxiliar as empresas do ramo da produção de peças plásticas/poliméricas a criar riqueza, garantindo efeitos positivos para a sociedade e para o meio ambiente. Estes são garantidos pelos vetores de utilização de sistemas de fabrico mais otimizados, de automação, de manufatura de precisão e garantia de qualidade. O objetivo passa por criar produtos de valor acrescentado, com maior nível de reciclabilidade (permitindo o seu reaproveitamento, revalorizando os seus resíduos), mais leves e resistentes mecanicamente. O projeto encontra-se estruturado em vários eixos de ação. São estes:

- PPS1: Design para a circularidade, sustentabilidade e valorização;
- PPS2: Novos materiais poliméricos avançados e multifuncionais;
- PPS3: Processos avançados de fabrico;
- PPS4: Integração inteligente de processos e produtos;
- PPS5: Gestão e disseminação.

Este projeto mobilizador conta com vários parceiros, não só empresas de referência no mercado nacional como a Iber-Oleff e OLI, mas também instituições de ensino universitário, como o Instituto Superior Técnico, Universidade de Aveiro e Universidade do Minho. O presente projeto encontra-se incluído no tema PPS4, e resulta de uma parceria entre a Iber-Oleff e Universidade de Aveiro. Esta empresa pretende conceber um método que permita o desenvolvimento e industrialização de produtos de aparência para o habitáculo automóvel, que incorporem peças com aspeto superficial metálico, customizáveis, sem defeitos e produzidos em massa através de um processo de injeção *one-shot* com pigmentos metálicos incorporados na peça.

1.2 Iber-Oleff

Nesta secção será retratada a empresa que propôs a realização do projeto que é descrito neste relatório, a Iber-Oleff. Esta empresa foi criada em 1993, pertence ao grupo IberoMoldes e emprega mais de 400 trabalhadores. A sua maior área de atividade consiste na produção de peças plásticas para a indústria automóvel. A empresa tem conseguido destaque, adquirindo a confiança de clientes como Volkswagen, Ford, BMW, entre outros. De acordo com a empresa isto deve-se ao seu plano estratégico no que diz respeito à ênfase dada à flexibilidade e agilidade da organização fabril e formação dos trabalhadores. De seguida serão apresentados os vários equipamentos e secções do chão de fábrica, que permitem o funcionamento da empresa. A seguinte figura 1.1 apresenta um esquema representativo dos processos estabelecidos no chão de fábrica, que permitem transformar a matéria-prima inicial num produto final de qualidade.

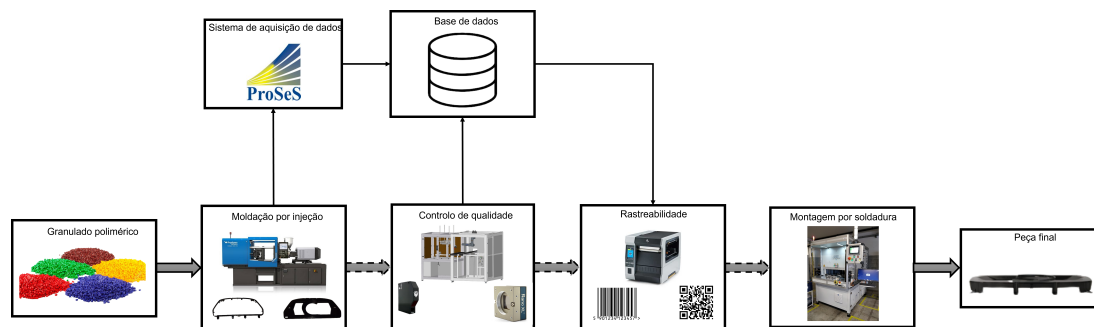


Figura 1.1: Chão de fábrica da Iber-Oleff.

Nesta figura as setas maiores cinzentas representam uma troca de material entre os diferentes módulos no chão de fábrica, enquanto que as setas menores representam trocas de informação. De notar que algumas setas maiores apresentam um contorno a traço interrompido. Isto representa o facto de as trocas de material apresentadas ainda não ocorrerem na atualidade, devido ao desenvolvimento dos módulos em questão, que nesta altura se encontra em decurso. Por isso, na realidade, depois da injeção das peças estas são encaminhadas para o equipamento de soldadura.

Como se pode observar pela figura 1.1, para produzir um mesmo produto final é necessário injetar duas peças distintas. Por vezes as empresas responsáveis pela injeção de peças plásticas encontram vários desafios relacionados com a geometria das peças

que são requisitadas pelos seus clientes. De facto estas podem apresentar contrasaídas e outras características geométricas que dificultam ou impossibilitam a sua produção através do processo de moldação por injeção. Como forma de contornar este problema a Iber-Oleff introduziu como solução a subdivisão das peças em questão em duas partes distintas (um *rear-frame* e um *front-frame*). Podemos encontrar um exemplo destas partes desenvolvido num software CAD (*computer-aided design*) na figura 1.2:

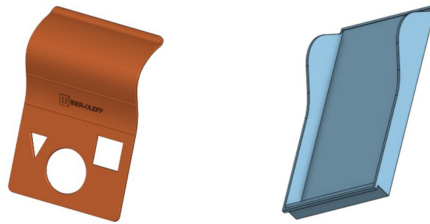


Figura 1.2: Exemplos de um *rear frame* e de um *front frame*.

Uma vez feita a injeção destas partes é necessário fazer a sua união. A Iber-Oleff, perante os vários possíveis processos que permitiriam realizar este processo optou pela soldadura por transmissão de laser. Este é um processo que apresenta uma estanqueidade e qualidade na união das peças muito superior àquela que é oferecida por processos de colagem. A nível estético também apresenta as suas vantagens já que o cordão de soldadura é invisível ao utilizador.

1.2.1 Máquinas de injeção



Figura 1.3: Máquinas de injeção responsáveis pela produção do *front* e *rear frame*.

Na Iber-Oleff existem variados equipamentos responsáveis pela injeção de polímeros. Há vários equipamentos *babyplast* e outros equipamentos que apresentam forças de fecho de muitas toneladas. O uso destas máquinas está diretamente relacionado com o tamanho do molde a ser usado e consequentemente com o tamanho da peça a ser produzida. Para estas poderem trabalhar, necessitam que o polímero seja introduzido na tremonha. O sistema de alimentação existente das máquinas de injeção na Iber-Oleff é totalmente automatizado, conseguindo-se a transferência da matéria-prima desde o armazém até à sala das máquinas através de tubagens percorridas por ar, com forte sucção. Após a abertura do molde, a extração da peça e remoção dos canais de alimentação é feita com braços robóticos EPSON. Estes braços, podem colocar a peça numa estação de montagem, para que um trabalhador possa assemblar alguns conjuntos se assim for exigido pelo projeto do produto. Devido a esta grande automatização que existe dos processos,

apenas é necessário um trabalhador para vigiar três ou quatro máquinas. Estes devem, no fim do ciclo de injeção analisar as peças e verificar a existência de defeitos. Caso após inspeção se verifique a existência de defeitos na peça, o operador pode classificar essa peça específica como “*scrap*”, através de um HMI (*human machine interface*), sendo esta descartada, enquanto que as peças sem defeitos seguem para inspeção posterior na célula integrada de controlo de qualidade.

1.2.2 Célula integrada de controlo de qualidade



Figura 1.4: Célula de controlo de qualidade responsável pela identificação de defeitos nas peças.

As operações que são levadas a cabo na célula integrada de controlo de qualidade são o controlo de qualidade superficial e o controlo dimensional das peças. O controlo de qualidade superficial é garantido através de um sistema de controlo por visão artificial, que deve identificar defeitos superficiais na peça, como vazios, chupados ou linhas de soldadura, através do processo de defletometria. O equipamento usado consiste num conjunto de luzes e uma câmara Dalsa Genie NanoXL M5100, que deteta os defeitos superficiais. O controlo dimensional da peça deve ser precedido de um *scanning* da mesma que resulta na sua recriação digital. Uma vez realizada esta etapa, o desenho gerado é utilizado para fazer uma comparação ponto-a-ponto das coordenadas relativas de vários locais da peça. O equipamento utilizado para fazer o *scanning* da peça é um GOCATOR 2530 *high speed 3D laser profiler*. O processamento da informação contida no desenho obtido por *scanning* é feito pelo software HDevelop da HALCON. As peças que passam a inspeção feita pela célula recebem uma etiqueta RFID ou código de barras, que permite realizar a rastreabilidade da peça e de todo o seu processo de produção. É a partir desta célula que se retira informação sobre a qualidade das peças. São identificados defeitos visuais, como riscos superficiais mas também são identificados outros defeitos estruturais, como empenos, por exemplo. Esta informação será reencaminhada para uma base de dados, e é crucial na determinação da força de separação entre duas peças soldadas.

1.2.3 Equipamento de soldadura por transmissão de laser

O interesse da empresa no processo de soldadura por transmissão de laser, surge da filosofia de inovação constante e de criar novas tecnologias que permitam a construção de objetos de qualidade superior para, neste caso, a indústria automóvel. Este interesse é reforçado pela presença da empresa no projeto PolyWeld, onde em conjugação com a Universidade do Minho e PIEP, está a estudar a conceção de uma metodologia que

permita o desenvolvimento de peças plásticas produzidas por moldação por injeção e unidas por um processo de soldadura por transmissão de laser. A Iber-Oleff que tem como desafio tornar o processo de soldadura por transmissão de laser mais otimizado, conseguiu identificar vários parâmetros a considerar para a soldadura de um produto polimérico:

- Velocidade do feixe de laser;
- Intensidade do feixe de laser;
- Força de compactação das duas peças a soldar.

Todas estas variáveis são monitorizadas no equipamento presente no chão de fábrica da empresa. O seu controlo, por sua vez, é garantido de forma manual pelos operadores, através de um HMI e de um computador local que fazem a comunicação dos parâmetros do processo para um PLC Siemens S7/300 através do protocolo Profibus. Este por sua vez atua os parâmetros no equipamento de soldadura, uma máquina do modelo LPKF PowerWeld 4600, que pode ser visualizado na figura 1.5:



Figura 1.5: Equipamento de soldadura responsável pela união do *rear frame* e *front frame*.

No entanto, a informação sobre estas variáveis não é comunicada para uma base de dados. Para se iniciar a tomada de decisões de forma inteligente, no processo de soldadura, terá que se efetuar essa ligação à base de dados.

1.3 O problema

Como se pode perceber a configuração atual do processo de soldadura por transmissão de laser na Iber-Oleff não se encontra otimizado para os objetivos da empresa. Como já foi referido anteriormente a empresa visa estabelecer uma cadência alta de produção. Deste modo a alteração manual de parâmetros, via HMI, pelo operador do equipamento não é de todo favorável, já que se iriam criar grandes tempos mortos entre espaços de produção. De momento a Iber-Oleff define um conjunto único de parâmetros para um dia inteiro de produção, minimizando-se este problema. No entanto, esta configuração não permite o ajuste de parâmetros para peças para as quais os parâmetros padrão não são adequados. Verifica-se, por isso, que muitas peças são produzidas com defeitos associados. Como nos seus objetivos para o S4Plast prevê-se a produção de peças sem defeitos, esta solução não é viável. Devido a este desafio, a empresa, identificou a necessidade da integração da informação proveniente dos vários sistemas que compõem a linha de produção, para

que possam ser tomadas decisões em tempo real que corrijam os erros, eliminando o desperdício e procurando a automatização do processo. É neste sentido que é importante a existência de um sistema de controlo inteligente associado ao processo de soldadura por transmissão de laser.

1.4 Objetivos

O objetivo principal do projeto apresentado está patente no título deste relatório: “Desenvolvimento de um sistema de controlo inteligente para a soldadura por transmissão de laser de termoplásticos”. Este objetivo geral, materializa-se nas seguintes etapas:

- Estabelecer uma comunicação entre um equipamento de soldadura e a base de dados da empresa (Microsoft SQL) e integrar a rastreabilidade das peças na produção;
- Parametrização automática do equipamento de soldadura a partir do reconhecimento de um conjunto de peças;
- Criação de um modelo preditivo (*machine learning*) que permita perceber se é possível existirem defeitos na produção de um produto final específico;
- Implementação da solução proposta.

Este projeto não se foca na instrumentação dos equipamentos e como esta deve ser feita para se conseguirem obter as informações mais relevantes do processo. Foca-se, sim, em extrair a informação dessa instrumentação e conseguir tomar decisões em tempo real, que permitem tornar a produção mais eficiente.

1.5 Organização do documento

O documento atual, representativo do projeto desenvolvido, encontra-se subdividido em 4 secções. Primeiramente, será realizado um enquadramento teórico onde serão apresentados os conceitos teóricos que regem o processo de soldadura por transmissão de laser e outros que servem de base para as ferramentas utilizadas no projeto. Neste enquadramento, será feita também uma análise ao estado da arte no que toca ao controlo inteligente, tanto em processos de soldadura por laser, como em outros processos produtivos, permitindo perceber-se que soluções são encontradas atualmente e quais as ferramentas que são implementadas. De seguida é apresentada a solução proposta para este projeto, em conjunto com todos os seus módulos e como estes foram definidos. Por fim, serão apresentadas os resultados e as conclusões que podem ser extraídas deste trabalho.

Capítulo 2

Enquadramento Teórico

Neste capítulo será abordado e explicitado a nível teórico o processo de transformação de produto relevante para este projeto que ocorre na linha de produção da Iber-Oleff. É este o processo de soldadura por transmissão a laser, responsável pela união de um *rear frame* e um *front frame*. Também serão abordados conceitos relevantes para a execução das funcionalidades da solução proposta, tais como design experimental e *machine learning*. Por fim serão revistos projetos que serviram de inspiração para obter a solução apresentada no capítulo 3.

2.1 Soldadura por transmissão de laser

Neste subcapítulo será apresentado e explicitado o processo de transformação associado à união de peças poliméricas praticado na Iber-Oleff. Trata-se da soldadura por transmissão de laser. Serão explicados o processo e as várias variáveis que de alguma forma o influenciam. Com a leitura deste subcapítulo conseguir-se-á obter uma compreensão aprofundada sobre o processo que este projeto tenta automatizar e otimizar.

2.1.1 O processo

Os polímeros nos dias que correm são usados nas mais variadas áreas científicas. Deste modo, são criadas inúmeras exigências de comportamento para produtos deste tipo de material. Cria-se então a necessidade da existência de um processo que permita ligar as peças que seja flexível, rápido e ecológico [1]. No entanto, a obtenção de um processo que garanta também a melhor qualidade possível da peça, tem sido ao longo dos anos um processo moroso e com os seus desafios [17]. A soldadura é vista como processo atrativo já que permite, quando feita corretamente, a ligação permanente entre dois componentes, através da fusão e junção dos materiais numa interface que com o auxílio da compressão se irá solidificar [18], sem a necessidade de se usarem agentes que influenciem a estética de uma peça, como parafusos ou rebites. Tem ainda a vantagem de geralmente conseguir aumentar a resistência mecânica dos materiais envolvidos no ponto de ligação (junta). Existem, portanto, vários processos de soldadura industriais que podem ser utilizados na produção de ligações entre peças poliméricas. Dentro destes processos e como solução para o problema enunciado surgiu a soldadura por feixe laser [1, 17, 19]. A soldadura por feixe de laser é um processo que tem como vantagens necessitar de pouca energia para unir as peças, conseguir ligações mais resistentes, cordões de soldadura de maior qualidade

e poder ser usado na junção de geometrias complexas. Por outro lado, este processo está mais dependente dos materiais envolvidos, do historial do produto, pigmentos e outros aditivos [18]. A soldadura por feixe de laser pode ser conseguida de duas formas distintas [1]:

- Focar o feixe de radiação numa junta formada pelas duas peças, que é diretamente absorvido pelos polímeros, levando a uma produção muito elevada de calor e consequentemente, à fusão dos materiais, na sua interface. Este processo é conhecido por soldadura a laser direta;

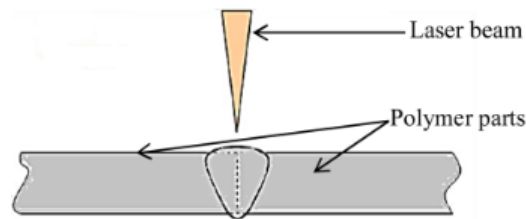


Figura 2.1: Ilustração de um processo de soldadura a laser direta [1].

- Soldadura por transmissão, que será o tema abordado de seguida.

A soldadura por transmissão de laser é um processo de ligação de dois componentes. Este utiliza um feixe laser (*light amplification by stimulated emission of radiation*) para garantir a ligação dos componentes. Os materiais das duas peças devem ter propriedades óticas diferentes - um material deve ser opaco ao feixe laser e o outro transparente (este não deve ser muito refletivo, para otimizar a transmissão da radiação). Esta transparência é uma propriedade que não depende apenas do material utilizado na peça. Está relacionada também com a aditivação do polímero e com a espessura da peça. Caso se garanta a transparência de uma peça, verifica-se a passagem da radiação e a sua absorção pela outra peça, opaca à radiação. Ao ser absorvida, a radiação funde o material. A energia absorvida é posteriormente transmitida por condução para o outro material criando-se um cordão de soldadura que une as peças, com o auxílio de uma força compressiva, que deve promover o contacto e otimizar a condução de calor entre as peças [1, 17–19]. O processo encontra-se esquematizado na figura 2.2.

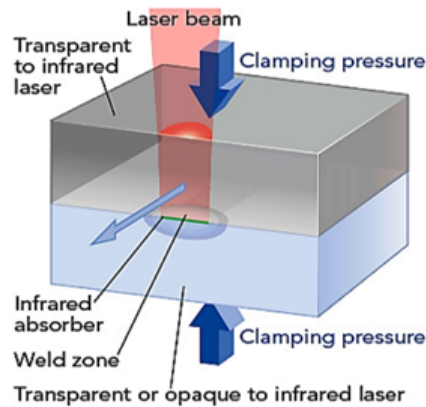


Figura 2.2: Ilustração do processo de soldadura por transmissão de laser [1].

Este é o processo de soldadura de polímeros mais relevante na indústria. Isto deve-se ao facto de não envolver contacto, contaminação e vibrações, permitindo a produção de junções de qualidade (com rigidez elevada e boa resistência química), com valor estético para uma grande variedade de produtos. Outros atributos do processo de soldadura por transmissão de laser são a sua fácil automatização e capacidade de integração num contexto de indústria 4.0, capacidade de produção em altas cadências, desgaste reduzido do equipamento, entre outros [1, 17, 19]. Por outro lado, é um processo que depende em grande parte das propriedades óticas dos materiais, tendo estes, muitas vezes, que ser aditivados com corantes e outros, que lhes permitam alterar as propriedades em questão.

2.1.2 Parâmetros do processo

Para que a soldadura por transmissão de laser seja bem-sucedida deve-se atender aos vários parâmetros do processo, tais como a configuração do próprio laser. Sendo este o meio pelo qual se transfere energia para a peça é importante perceber quais são os diferentes parâmetros e como estes influenciam o processo. Os processos de soldadura por laser são altamente influenciados pela interação entre a radiação laser e o material. Quando estes elementos entram em contacto, pode verificar-se a transmissão, reflexão ou absorção da radiação. Para ocorrer a união entre duas peças tem que se verificar a absorção da radiação, devendo também ocorrer transmissão no processo de soldadura por transmissão de laser. Quando se der a absorção da radiação, esta deve ocorrer de forma eficiente, na localização destinada ao cordão de soldadura das peças. Como já se explicou anteriormente, esta absorção de energia leva ao aquecimento dos polímeros, até à temperatura de transição vítrea para materiais amorfos e até à temperatura de fusão para materiais semi-cristalinos, culminando na sua fusão. Esta absorção de energia causa uma mudança irreversível de propriedades nos materiais, tanto mecânicas, como óticas, térmicas e morfológicas, devido a alterações químicas e na estrutura polimérica do material [17]. Nesta fase deve ocorrer a mistura das cadeias poliméricas de ambos os materiais, bem como o entrelaçamento das mesmas. Depois de concluída a transmissão de calor, durante o processo de arrefecimento, solidificam-se as partes fundidas dos materiais poliméricos, concluindo-se o processo de soldadura das peças [1, 18]. Este processo é conhecido como autoesão [1]. Qual dos três processos (absorção, reflexão ou transmissão) ocorre quando o laser e peça entram em contacto determina-se em função dos materiais

e aditivos utilizados, mas também do comprimento de onda, potência, tamanho e qualidade do feixe de laser.

No que toca ao comprimento de onda do feixe de laser, este é caracterizado por Acherjee et al. em [19] como o parâmetro mais importante quando se efetua a escolha de uma fonte laser, devido à sua direta correlação com as propriedades óticas dos materiais. Entende-se que os termoplásticos são, na sua maioria, transparentes a ondas com 400 a 1600 nanómetros de comprimento de onda [18]. Esta gama de comprimentos de onda é muitas vezes denominada de NIR (*Near Infra Red*). Deste modo, quando se trata do método de soldadura por transmissão de laser, devem ser usados comprimentos de onda compreendidos entre os dois valores mencionados [18, 19].

De acordo com Gonçalves et al. [18], a potência do laser é o parâmetro mais utilizado para mudar a temperatura da solda. De facto, a potência do laser é o principal responsável pela modificação da intensidade do feixe laser, podendo ocorrer vários cenários de acordo com a potência do feixe escolhida. Pode não ocorrer fusão, ocorrer fusão parcial, soldadura e soldadura com degradação do material (com vários graus de gravidade). O primeiro caso verifica-se quando a potência escolhida não consegue aquecer os materiais até às suas temperaturas de fusão ou temperaturas de transição vítrea, não ocorrendo por isso a união das peças. Por outro lado, caso se opte por uma potência demasiado elevada, ocorrerá a fusão de ambos os materiais, mas verifica-se também a decomposição e queima dos materiais [19]. A potência do feixe também pode influenciar o tamanho do cordão de soldadura [18]. Este parâmetro tem uma grande correlação com a quantidade de calor que é introduzida nos materiais e por isso também se cria uma correlação forte com a resistência e largura do cordão de soldadura e consequentemente com a qualidade da soldadura. O valor de potência ótima a utilizar para um certo processo de soldadura pode ser determinado pelo critério de máxima resistência de solda. Este critério visa a criação de um limiar a partir do qual o aumento de potência do laser deixa de ser benéfico para o processo, causando a degradação térmica do material e lesando a resistência do cordão de soldadura [19]. De salientar que a energia aplicada ao material também é influenciada por outros parâmetros igualmente importantes, tais como a velocidade de soldadura e tamanho do feixe laser [1, 18].

Como referido anteriormente, o tamanho do feixe laser é um fator determinante para a energia que é fornecida para as peças a unir, estando intimamente ligado à qualidade da peça [18]. Mais precisamente, este parâmetro influencia a densidade energética do feixe laser. Quanto menor for o seu tamanho/diâmetro, maior será a densidade energética do feixe. No entanto, este facto não torna a diminuição do foco do laser, isto é, o aumento do seu tamanho/diâmetro um efeito indesejável. Pelo contrário, com o aumento do tamanho do feixe, consegue-se aumentar a largura do cordão de soldadura e consequentemente aumentar a resistência mecânica do mesmo. Obtém-se também um gradiente térmico menos pronunciado, evitando-se deste modo, defeitos como por exemplo empenos e queima do material. Tal como para a potência, existe um ponto para o qual a desfocagem do feixe laser deixa de aumentar a resistência do cordão de soldadura, passando a ter o efeito inverso, devido à falta de fusão que se verifica nos materiais [19]. Ao contrário do que se verifica com a largura do cordão de soldadura, com a diminuição do foco do laser a profundidade do cordão diminui, enquanto que um feixe mais denso a

nível energético, leva a maiores profundidades de fusão do polímero. Por fim, Acherjee refere em [19] que a forma do feixe de laser também influencia a qualidade da soldadura. De facto, Coelho et al. verificaram em [2] como se pode ver pela figura 2.3, que para uma mesma potência fornecida a um equipamento de soldadura, conseguia-se obter uma maior densidade de energia com um feixe elíptico em vez de um feixe circular.

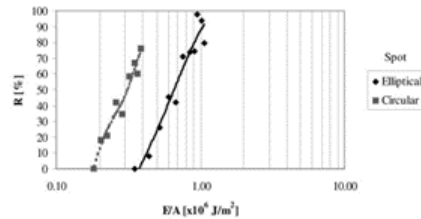


Figura 2.3: Feixe de laser Circular/Elíptico [2].

Significa isto, que se consegue obter a mesma qualidade de soldadura, com menor consumo de potência por parte do equipamento, quando se usa um feixe de laser elíptico.

Outro parâmetro de grande relevância para o processo de soldadura por transmissão, é a velocidade de soldadura. Tal como foi mencionado anteriormente, este parâmetro em conjunto com a potência do laser determinam a quantidade de energia que é transmitida para a junta de soldadura e por conseguinte, as características do cordão de soldadura. Estas características são, por exemplo, a largura, profundidade e resistência mecânica do cordão. Por extensão, a velocidade de soldadura está diretamente correlacionada com a qualidade da peça final [18, 19]. Verifica-se que quanto menor for a velocidade de soldadura, maior é o tempo de interação entre o laser e o polímero. Deste modo, deve-se ter em conta que se esta for muito baixa, pode causar a degradação do material na zona da junta e que se for muito alta, o tempo de interação entre laser e polímero pode não ser suficiente para causar a fusão dos materiais [19]. De notar, que para aumentar a largura, profundidade e resistência mecânica da junta deve optar-se por velocidades de soldadura mais baixas, ainda que estas propriedades também estejam sempre dependentes da potência de laser empregada. De facto, a velocidade ótima de soldadura para um certo material é função da difusividade térmica e absorvidade do material, bem como da potência do laser [19].

Para se obter um cordão de soldadura de qualidade é necessário o contacto próximo de ambas as partes de modo a otimizar a condução de calor da peça que absorve a radiação para a peça que transmite a radiação. Só desta forma, se consegue a fusão dos materiais, e a mistura das cadeias poliméricas de ambos as peças. A forma mais eficaz de garantir este contacto próximo é a aplicação de uma força de compressão nas peças. Esta força também facilita a obtenção das tolerâncias corretas e a prevenção de defeitos na peça final e pode ser aplicada por sistemas hidráulicos ou pneumáticos [18, 19].

2.2 Conceitos relacionados com o modelo preditivo

Neste subcapítulo serão apresentados os conceitos relevantes para a perceção do processo de desenvolvimento do modelo preditivo que foi realizado neste projeto, e que permi-

tem através dos dados gerados pelos dispositivos no chão de fábrica encontrar relações complexas entre os vários parâmetros medidos e otimizar os processos de produção [20].

2.2.1 *Machine learning*

De acordo com Hong em [21], *machine learning* (ML) insere-se no âmbito da inteligência artificial (IA) e tem como fim a criação de algoritmos que permitem que um software consiga aprender de forma autónoma a resolver problemas para os quais não foi programado a priori. Como não há *input* na forma de código de programação, usa-se como base para a aprendizagem do sistema exemplos de funcionamento correto do sistema em questão [6, 22, 23]. No entanto, não basta um algoritmo de ML recordar-se dos vários exemplos que lhe são apresentados e reproduzir os efeitos desejados. Este também deve conseguir extrapolar e prever possíveis variantes do problema com as quais não foi treinado e criar soluções pertinentes para essas mesmas. Deste modo, é muito importante a escolha de um algoritmo que construa modelos preditivos que consigam apresentar essas soluções relevantes para o problema apresentado, isto é, classificar ou regredir novas variantes [22, 23]. Trata-se de algoritmos de *supervised learning* (SL), que a partir de um conjunto de dados que lhes são fornecidos, conseguem fazer previsões. Estes algoritmos podem ser subdivididos em algoritmos de regressão ou classificação. Os algoritmos de regressão permitem obter previsões sobre os valores de propriedades contínuas, enquanto os algoritmos de classificação são usados para prever os valores de propriedades discretas. Verifica-se, no entanto, que se podem resolver problemas de regressão, através de algoritmos de classificação, através da discretização dos contínuos num conjunto de intervalos/classes [24]. Existem algoritmos com funções alternativas, nomeadamente os algoritmos de *unsupervised learning* (UL) que visam a descoberta de uma estrutura organizacional nos dados fornecidos e juntá-los em grupos. Por outro lado, os algoritmos de *reinforcement learning* (RL) ditam a ação do software, numa determinada situação, visando a recompensação de ações desejadas [3,6]. Para este trabalho, em que se pretende a previsão do valor da força de separação entre duas peças (um *front* e um *rear frame*), o algoritmo implementado tem que ter a capacidade de fazer previsões acertadas sobre qual a força de separação entre duas peças, se fossem unidas com recurso a um certo conjunto de parâmetros, para diminuir a quantidade de falhas do produto, pelo que os algoritmos de SL se encontram mais ajustados. Neste processo, os dados recebidos deverão ser divididos em conjuntos de treino e teste. Os algoritmos aprendem um conjunto de padrões a partir dos conjuntos de treino e aplicam esses padrões aos conjuntos de teste, para formarem as suas previsões [3]. Este conjunto de ações encontra-se esquematizada na figura 2.4.

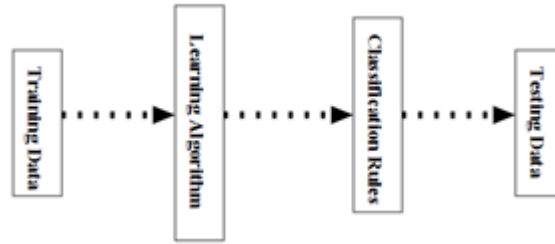


Figura 2.4: Processo de aprendizagem e teste de um algoritmo de classificação [3].

No que toca à escolha de algoritmo, esta é geralmente feita de acordo com a exatidão da previsão que o mesmo produz. Este pode ser classificado quanto à variância e distorção (*bias*) dos resultados que produz. A variância de um algoritmo reflete a sua capacidade de produzir classificações corretas para diferentes conjuntos de dados. Um algoritmo de variância alta, poderá apresentar resultados satisfatórios ou insatisfatórios dependendo do conjunto de dados com o qual for testado. Por outro lado, a distorção de um algoritmo reflete a sua capacidade de conseguir reproduzir a relação entre variáveis observável no mundo real, para um certo problema. Um algoritmo de alta distorção apresenta grandes lacunas na representação de uma relação verdadeira entre diferentes parâmetros e por isso realizará, consistentemente, previsões incorretas. Deste modo, deve tentar-se reduzir tanto a variância de um algoritmo, que quando muito alta pode representar um sintoma de *overfitting*, que será descrito adiante, de forma a reduzir a complexidade do algoritmo bem como a distorção do algoritmo, de forma a conseguir-se uma representação fidedigna do fenómeno em questão e não tornar o problema mais simples do que ele realmente é [23].

Redes neuronais artificiais

As redes neuronais artificiais (ANN) são um conjunto de elementos interligados, denominados de neurónios (devido à sua semelhança funcional com os neurónios do cérebro humano), que recebem informação de outros neurónios e processam o conjunto de dados através de funções não lineares complexas [3, 6, 25, 26]. Após este tratamento, a informação processada é transferida para outros neurónios. Este fator explica a importância das ANN, que permitiram a classificação/regressão viável de várias instâncias, que têm um relacionamento complexo e não linear. Este tipo de abordagem não teria sido conseguida com outro tipo de algoritmos que classificam as instâncias com base em relações lineares [26]. De um modo geral as estruturas que se associam a ANN são as redes neuronais *feedforward* e as redes neuronais *feedback* [4, 27]. Uma ANN *feedforward* tem uma passagem de informação de único sentido (dos neurónios de entrada para os neurónios de saída), enquanto que uma ANN *feedback* permite a passagem de informação no algoritmo de forma bidirecional, através da definição de ciclos no âmbito da rede [4]. Essa flexibilidade das redes *feedback* permite a criação e uso de uma memória interna da rede para processar a informação, ao contrário do que acontece nas ANN *feedforward*. A aplicação destas redes dá-se em tarefas como o reconhecimento de padrões sequenciais na fala e escrita, por exemplo [28]. Tendo em conta que estas aplicações não se assemelham ao propósito do algoritmo preditivo do presente projeto, a restante descrição irá focar-se em ANN *feedforward* que representam uma estrutura mais simples mas que também se adequa à realização de regressões não lineares. Para perceber melhor as diferenças entre

as ANN *feedforward* e *feedback* pode-se remeter à figura 2.5 onde as diferenças entre estas duas estruturas se encontram esquematizadas.

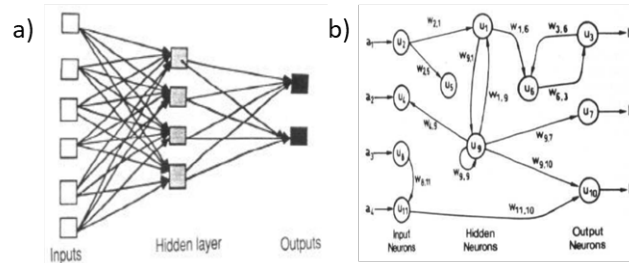


Figura 2.5: Representação gráfica de uma ANN *feedforward* (a)) e de uma ANN *feedback* (b)) [4].

Existem diferentes tipos de ANN *feedforward*. O mais simples é o *single-layer perceptron* que apresenta uma única camada de neurónios. Os outros dois tipos, com um nível de complexidade maior que o *single-layer perceptron* são o *multi-layer perceptron* e a *radial basis function network* [4]. Ao contrário dos *single-layer perceptrons*, os *multi-layer perceptrons* e *radial basis function networks* são compostos por várias camadas às quais a informação é alimentada de forma sequencial, diferindo apenas no facto de que as *radial basis function networks* apresentam funções de ativação de base radial [26]. O processamento da informação que ocorre num neurónio pode ser percecionado a partir da imagem na figura 2.6.

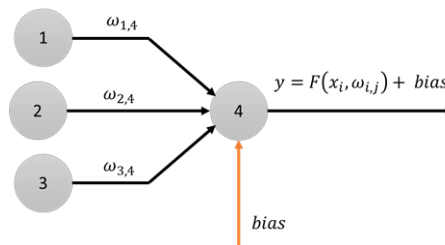


Figura 2.6: Representação de um neurónio artificial (4) e das suas interações e comportamento.

Como pode ser percecionado pela figura anterior, cada pedaço de informação que é passado de neurónio em neurónio (x_i) encontra-se associado a um certo peso ($\omega_{i,j}$) enquanto que cada neurónio associa-se a um certo valor de erro (*bias*). Estas variáveis são incorporadas na função de ativação do neurónio (F) para produzir a saída de informação (y). De facto, é na função de ativação que reside a capacidade deste tipo de algoritmo conseguir assimilar comportamentos complexos e não lineares, separando-o de modelos de regressão linear [5]. Podem existir várias funções de ativação associadas a cada camada de neurónios artificiais, podendo estas ser de diferentes tipos, mais ou menos complexos:

- Função Passo ou *step*;
- Função Linear;
- Função Sigmóide;

- Função Tangente hiperbólica;
- Função ReLU;
- Função ELU;
- Função *Swish*;
- Função *Softmax*.

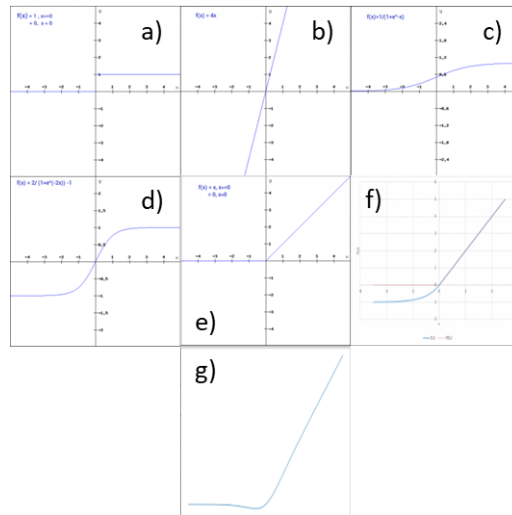


Figura 2.7: Representação gráfica das funções de ativação passo (a)), linear (b)), sigmóide (c)), tangente hiperbólica (d)), ReLU (e)), ELU (f)) e *swish* (g)) [5].

A partir dos gráficos apresentados na figura 2.7, consegue-se dividir as diferentes funções de ativação entre funções lineares (Passo, Linear) e não lineares (sigmóide, tangente hiperbólica, ReLU, ELU, *swish*). A função *softmax* também é não linear apesar de não estar presente na figura 2.7). São as funções não lineares que permitem aos algoritmos mapear os comportamentos não lineares dos fenómenos naturais estudados. É por isso de vital importância ter a capacidade de saber selecionar a função de ativação que mais se adequa ao algoritmo pretendido. Devido ao seu melhor desempenho geral é boa prática iniciar a criação do algoritmo com o uso de funções de ativação ReLU. Por outro lado, deve-se evitar aplicar as funções sigmóide e tangente hiperbólica em camadas escondidas, devido aos seus declives reduzidos para valores extremos de informação de entrada [5].

As ANN *feedforward* são sequenciadas em três tipos de camadas diferentes: camadas de entrada, camadas de saída e camadas escondidas (*hidden layers*), como se pode ver nas figuras 2.5a e 2.8. A camada de entrada tem como função recolher informação; as camadas escondidas devem processar esta informação e a camada de saída define os valores previstos para os diferentes parâmetros pretendidos [3, 26]. A determinação do número de camadas escondidas e do número de neurónios em cada camada escondida é fulcral para a eficácia do algoritmo definido. Caso se escolham menos neurónios que os necessários, são criados problemas a nível das capacidades de aproximação e generalização e caso se escolham mais que os necessários, a procura pelos parâmetros de saída ótimos torna-se mais difícil, devido a um problema de *overfitting* [26].

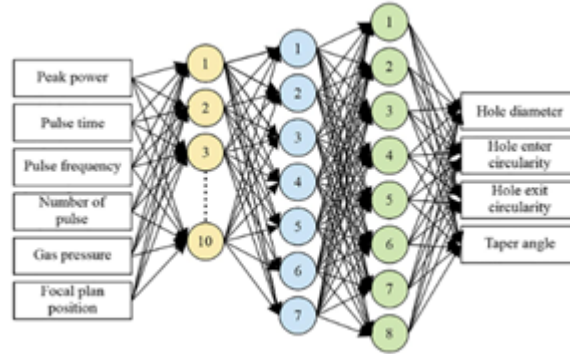


Figura 2.8: Exemplo de uma ANN com três camadas escondidas, aplicada na maquinagem a laser de aço [6].

Para esta estrutura de ANN, o processo de aprendizagem do algoritmo envolve a otimização das funções não lineares de cada conexão dentro da camada escondida da rede neuronal, isto é, a alteração do peso das conexões e do *bias* associado a cada neurónio. Inicialmente, é prática comum inicializar os pesos das conexões com valores aleatórios e o *bias* dos neurónios com valores nulos. Após um processo de introdução repetitiva de dados de treino no algoritmo, comparam-se os valores dos parâmetros de saída com os valores desejados para os mesmos e ajustam-se os valores dos pesos e *bias* de forma que haja uma aproximação dos parâmetros de saída aos valores pretendidos. Um dos algoritmos mais viáveis que assegura esta aprendizagem das redes neuronais é o algoritmo *backpropagation*, apresentado na figura ???. Este algoritmo é bastante utilizado devido à sua capacidade de conseguir mapear resultados em problemas multivariável [29]. Este, essencialmente, avalia os erros de previsão cometidos pelo algoritmo através da função de perda e altera de forma recursiva os valores de peso associados a cada conexão entre neurónios e o *bias* associado a cada neurónio de forma a minimizar esses mesmos erros. Estas alterações geralmente ocorrem no sentido em que o gradiente da função erro é mais acentuado e mais negativo. Este método que representa a forma mais comum de atingir os resultados ótimos de previsões através do algoritmo *backpropagation* denomina-se de *gradient descent* [27]. De salientar que as funções de ativação devem ser deriváveis para que se possa implementar este algoritmo [5]. A função de perda associada a este algoritmo pode ser dada por:

$$L = \sum_{i=1}^{n_L} (f_j - y_j)^2. \quad (2.1)$$

Nesta equação, o valor de f_j corresponde ao valor de uma previsão efetuada, y_j representa o valor pretendido para essa previsão e n_L representa o número de neurónios na camada de saída [29]. Deste modo, consegue-se perceber que a função de perda do algoritmo de *backpropagation* é uma soma de todos os erros de previsão cometidos pelos vários neurónios da camada de saída do algoritmo. É esta a função que é utilizada para investigar de que forma os pesos de cada conexão e *bias* de cada neurónio influenciam o resultado final, já que a previsão efetuada pelo algoritmo depende precisamente destas variáveis, como também se pode observar pela figura ???. Para se conseguir obter esta informação é necessário calcular a derivada da função de perda em relação, tanto aos pe-

dos das conexões, como em relação aos erros de cada neurónio, podendo ser esses valores determinados pela regra da cadeia [7]. Deste modo, pode avaliar-se o efeito que o *bias* de um neurónio tem na previsão final a partir da seguinte expressão, considerando apenas um neurónio de saída na rede.

$$\frac{\partial L}{\partial bias} = \frac{\partial L}{\partial f} \times \frac{\partial f}{\partial bias}. \quad (2.2)$$

A partir da expressão providenciada anteriormente e da expressão presente na figura ?? consegue-se calcular-se a fórmula que nos permite obter a variação do erro de previsão de acordo com o *bias* associado a um neurónio.

$$\frac{\partial L}{\partial bias} = \frac{\partial L}{\partial f}. \quad (2.3)$$

Conclui-se, portanto, que o efeito do *bias* de cada neurónio sobre a função de perda é igual ao efeito da previsão final efetuada pelo neurónio. Por outro lado, a derivada da função de perda em relação aos pesos das conexões realizadas com um neurónio podem ser dadas pela seguinte expressão.

$$\frac{\partial L}{\partial \omega} = \frac{\partial L}{\partial f} \times \frac{\partial f}{\partial \omega}. \quad (2.4)$$

A partir desta expressão, é possível obter o valor da derivada em questão, a partir da equação presente na figura 2.6.

$$\frac{\partial L}{\partial \omega} = \frac{\partial L}{\partial f} \times \frac{\partial F}{\partial \omega}. \quad (2.5)$$

Conclui-se, portanto, que o efeito do conjunto de pesos que afetam as diferentes conexões a um neurónio sobre a função de perda estão relacionadas com o efeito que a previsão do neurónio tem sobre a função de perda e com o efeito que os pesos têm nas funções de ativação de cada neurónio [7]. Uma vez recolhidas estas informações para todos os conjuntos de dados de treino de um algoritmo e para todos os neurónios do algoritmo, estes valores são acumulados nos gradientes $\Delta\omega$ e $\Delta bias$. Estes valores podem ser subsequentemente utilizados por um método de otimização como o *gradient descent* para atualizar os valores de peso e *bias* de forma a reduzir a função de perda [7]. No que diz respeito ao *gradient descent* este é um método que assenta nas duas equações seguintes:

$$W_{novo} = W_{velho} - \eta \times \Delta W, \quad (2.6)$$

$$b_{novo} = b_{velho} - \eta \times \Delta b. \quad (2.7)$$

Nestas equações η representa o ritmo de aprendizagem do algoritmo, que define o tamanho das alterações que são aplicadas a cada peso e *bias* [7]. Na seguinte figura 2.9, consegue-se perceber a diferença criada pela utilização de um ritmo de aprendizagem baixo quando comparado com um ritmo de aprendizagem alto.

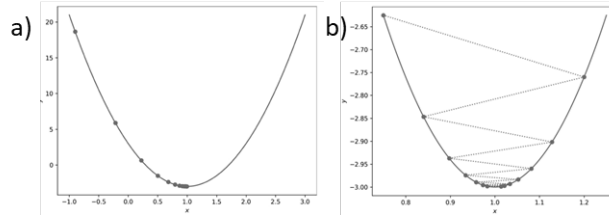


Figura 2.9: Comparação entre a otimização por *gradient descent* com o uso de um ritmo de aprendizagem baixo (a) e com o uso de um ritmo de aprendizagem alto (b) [7].

Como se pode observar pela figura anterior, um ritmo de aprendizagem mais reduzido conduz a alterações mais reduzidas nos valores da função perda na procura do seu valor mínimo. É de notar também que o tamanho dos passos reduz-se com o aproximar ao mínimo da função. Isto deve-se à redução do valor da derivada da função que comanda os valores dos gradientes calculados. Por outro lado, com valores de ritmo de aprendizagem mais elevados, os ajustes realizados causam uma sobre estimativa do valor do mínimo da função de perda. Deste modo verifica-se uma oscilação de valores em redor do mínimo da função, que dificulta a convergência no mesmo [7]. Deste modo, deve ser encontrado um equilíbrio para evitar o desenrolar de demasiadas iterações (aumento do número de *epoch*) deste processo causado por ritmos de aprendizagem baixos e a dificuldade em convergir no valor mínimo da função, causada por ritmos de aprendizagem elevados.

Verifica-se, no entanto, que este apresenta algumas dificuldades no que toca ao tempo de treino dos algoritmos que pode ser bastante extenso e na obtenção de resultados ótimos, já que em certas situações o algoritmo altera os pesos e *bias* de forma a atingir-se um mínimo local na função de perda da previsão, que não apresenta os valores que conduzem à produção de previsões ótimas, isto é, o mínimo global da função de perda não é atingido [7,26]. De forma alternativa, como forma de corrigir o problema do tempo de treino do algoritmo pode adotar-se o método *stochastic gradient descent*. Este é uma aproximação do *gradient descent* que permite reduzir o esforço computacional do treino do algoritmo, conseguindo-se convergências de uma forma mais célere [30]. Para reduzir o esforço computacional, em vez de utilizar o conjunto de dados de treino inteiro para aplicar o método de otimização, este é aplicado apenas para um subconjunto dos dados de treino. Este processo, para além de reduzir o tempo de convergência com o valor mínimo da função de perda, conduz à criação de gradientes com mais ruído, que permitem evitar mínimos locais que não representam o mínimo global da função de perda. De um modo geral neste método o subconjunto de dados de treino é gerado de forma aleatória e tem um tamanho ótimo variável que depende de caso para caso [7]. Uma outra alternativa que permite evitar os mínimos locais da função de perda é a aplicação de um novo hiperparâmetro denominado de momento. O efeito deste hiperparâmetro é baseado no conceito de inércia explanado pela terceira lei de Newton. Este permite introduzir na correção do valor do peso e erro um termo parcial de correções prévias que tenham sido realizadas. Deste modo, os valores atualizados dessas variáveis passam a ser dados pelas equações 2.8 e 2.9:

$$W_{novo} = W_{velho} + \mu \times v_W - \eta \times \Delta W, \quad (2.8)$$

$$b_{novo} = b_{velho} + \mu \times v_b - \eta \times \Delta b. \quad (2.9)$$

Nestas equações as variáveis v_W e v_b representam os vetores dos ajustes feitos tanto aos pesos como aos *bias*, pesados exponencialmente (tal que os ajustes mais recentes tenham uma maior relevância que os ajustes feitos anteriormente) e μ representa o fator de escala (entre 0 e 1) [7]. Após este processo de aprendizagem, estão criadas condições para a transformação dos dados iniciais/experimentais em previsões [6].

Tal como referido anteriormente sobre os algoritmos de ML, podem haver algoritmos com funções alternativas e que se condensam em algoritmos de SL, UL e RL. À semelhança disto, as ANN também podem ser classificadas da seguinte maneira [3]:

- *Supervised neural network*: nesta ANN, as saídas desejadas são conhecidas a priori. No final, existe uma comparação entre os valores pretendidos destas variáveis e os valores que são previstos pela rede neuronal. A partir desta comparação calcula-se o erro associado à previsão, e envia-se essa informação de volta para o algoritmo. É através desta informação, que este consegue aprender e mudar os seus parâmetros de forma autónoma;
- *Unsupervised neural network*: nesta configuração, não existe informação sobre quais os valores pretendidos das saídas, a priori. Nestas ANN, pretende-se apenas a classificação dos dados de acordo com a sua correlação;
- *Reinforced neural network*: numa ANN deste tipo, existe um *feedback* do ambiente que a rodeia relativamente à sua previsão. Esta pode ser classificada como sendo acertada ou errada. De acordo com esta classificação, a parte do algoritmo que fornece a previsão sobre o parâmetro é fortalecida ou enfraquecida, respetivamente. Tal como para as *unsupervised neural networks*, não existe informação a priori sobre os valores pretendidos para os parâmetros de saída.

No entanto, devido à complexidade das fórmulas que são desenvolvidas, as ANN são de difícil perceção no que diz respeito ao raciocínio que produz as previsões dos parâmetros de saída, pelo que este geralmente não consegue ser comunicado de forma eficaz [26].

Decision trees

Uma *decision tree* (DT) é um algoritmo que classifica instâncias de dados através de um conjunto de avaliações realizadas sobre os valores dos vários parâmetros que a si estão associados [6, 26]. Como o seu nome indica, a sua estrutura assemelha-se à de uma árvore, formada por vários ramos, que surgem de nós internos. Os nós internos representam condições impostas aos valores das propriedades das instâncias a avaliar e os ramos permitem a identificação do resultado da imposição da condição do nó precedente aos parâmetros da instância. No final do processo de classificação/regressão, é atribuída à instância uma classe específica ou valor numérico específico, consoante os valores dos seus parâmetros. Essas previsões são apresentadas em diferentes nós folha. Estes algoritmos podem ainda ser subdivididos em *classification trees* e *regression trees*, de acordo com o método praticado (classificação ou regressão). Um exemplo de uma DT encontra-se na figura 2.10.

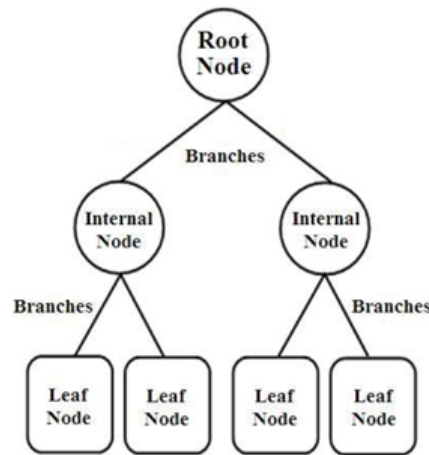


Figura 2.10: Exemplo de uma DT [8].

Uma DT deve ser o mais pequena e linear possível, sendo que a construção deste tipo de algoritmos, de forma ótima, por vezes é impossibilitada por problemas de *NP-completeness*, onde se verifica que o algoritmo não consegue encontrar uma solução para um certo conjunto de dados que lhe são fornecidos [23, 26]. Por outro lado, quanto mais reduzida/eficiente for uma DT, menos tempo é consumido a realizar as várias tarefas, quando os conjuntos de dados forem constituídos por várias instâncias. Consegue-se por exemplo reduzir a quantidade de tempo que se demora a estabelecer um limiar para a classificação/regressão de atributos numéricos das instâncias [26]. Existem vários métodos diferentes que permitem dividir os dados de forma ótima, não existindo um método único melhor que os restantes. Deve ser efetuado desta forma, um estudo comparativo das várias formas para dividir os dados específicos da aplicação em questão. Outro problema que pode ocorrer numa DT é o *overfitting* dos dados de treino, que ocorre quando o erro de uma solução desenvolvida pelo algoritmo com recurso a um conjunto de dados diferente do conjunto de dados de treino é maior que o erro de outra solução desenvolvida pelo algoritmo, quando se usa o conjunto de dados de treino (o algoritmo ajusta-se muito bem ao conjunto de dados de treino, mas mal a outros conjuntos de dados). Para se corrigir este problema, devem ser eliminados alguns ramos da “árvore”, para diminuir o tamanho da mesma [26]. A maior valência das DT, especialmente em comparação com as ANN, é a maior facilidade de compreensão e racionalização, por parte do humano, da classificação/regressão de uma instância feita pelo algoritmo [25, 26]. No entanto, este tipo de algoritmo, baseado na obtenção de resultados binários não é uma boa representação da realidade [25]. Apesar disto, a sua descoberta permitiu a derivação de vários algoritmos que, à semelhança das ANN conseguem fazer uma representação fidedigna de problemas reais com relações não lineares (permitem reduzir a distorção do algoritmo). Um destes algoritmos é o XGBoost que será estudado de seguida.

XGBoost

Para perceber melhor este algoritmo, é preciso conhecer o método *boosting*. Este método é usado em algoritmos como o AdaBoost e é apontado, como uma forma excelente de usar um algoritmo “fraco”, ou com previsões de pouca exatidão, como uma DT por exemplo,

e melhorá-la, criando um algoritmo “forte” [31]. Para que isto se verifique, têm que ser produzidos vários algoritmos base (mais “fracos”), a partir de subconjuntos de dados de treino, obtidos de forma aleatória. O peso de uma certa instância de dados, representa a forma fidedigna ou não, em como o algoritmo consegue produzir uma previsão correta. Quanto maior for o peso de uma instância, no conjunto de treino, pior ou menos exata será a previsão do algoritmo. Inicialmente, antes da construção do primeiro algoritmo “fraco” todos os dados apresentam pesos iguais. No entanto, após a recolha aleatória de dados e construção do primeiro algoritmo, adequado para os dados com qual é treinado, é efetuado um teste ao algoritmo “fraco” desenvolvido, com todo o conjunto de dados de treino. Uma vez feito este teste, consegue atribuir-se os pesos a cada pedaço de informação tendo em conta a exatidão das previsões realizadas. Quanto maior o peso da instância, maior a probabilidade de esta ser escolhida no subconjunto de dados de treino para treinar o algoritmo seguinte. A estrutura deste método permite um maior foco nos problemas mais exigentes associados ao algoritmo desenvolvido e procurar soluções para os mesmos, que o algoritmo “fraco” anterior não conseguiu encontrar [31]. Quando for desenvolvido o segundo algoritmo, com dados escolhidos com base no seu peso (quanto maior for, maior a probabilidade de ser incluído no subconjunto de dados), o teste com todo o conjunto de dados de treino é efetuado para a sequência, em série, dos algoritmos desenvolvidos, a partir da qual é efetuado o cálculo do erro. Este processo é efetuado um número limitado de vezes, verificando-se que a exatidão do algoritmo final é maior que a exatidão individual dos algoritmos “fracos” [9]. A figura 2.11 apresenta uma esquematização, de como o número de iterações deste processo influencia o erro cometido pelo algoritmo no processo de classificação:

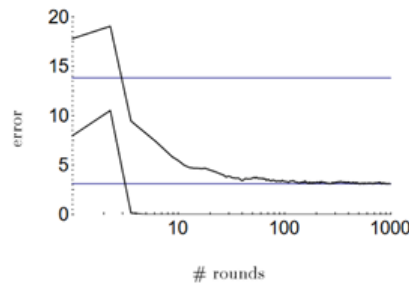


Figura 2.11: Efeito do *boosting* para dados de treino (em baixo) e dados de teste (em cima) [9].

Posto isto, conseguir-se-á uma melhor perceção do algoritmo XGBoost. O algoritmo XGBoost corresponde a uma adaptação do algoritmo DT. Este faz uso de *gradient boosted decision trees*, que permitem aumentar a escalabilidade do algoritmo e melhorar a exatidão das suas previsões [32, 33]. O *gradient boosting* apresenta algumas diferenças face ao *boosting* tradicional, sendo essencial para a acomodação de DT no algoritmo, uma vez que um *ensemble* de DT contém funções que não permitem a sua otimização num modo convencional [32]. A estrutura estatística do algoritmo consegue otimizá-lo através da adição das contribuições dos vários algoritmos “fracos” para se conseguir uma aproximação iterativa ao valor pretendido da previsão e diminuir o erro da mesma, seguindo-se a direção que é definida pelo gradiente negativo da função de erro [31]. Esta estrutura permitiu a integração de funções diferenciais no método de *boosting* permitindo obter melhores resultados para regressões e classificações não binárias. De resto, o processo é semelhante ao *boosting*, introduzindo-se novas DT em série, uma a uma, que têm como objetivo reduzir o erro na previsão do algoritmo “fraco” anterior (para isto se conseguir têm que ser feitas alterações aos parâmetros da DT). Quando o erro da previsão chegar a um nível em que já não consegue ser reduzido, ou se atingir o número máximo de DT no algoritmo, a adição destes algoritmos é parada. A forma como estas DT são construídas é “egoísta”, o que significa que, em qualquer altura, é tomada uma decisão que minimiza a função diferencial que descreve o erro da previsão. Isto, por um lado, facilita uma rápida obtenção de uma previsão ótima, mas também facilita o *overfitting* do conjunto de dados de treino. Para se evitar este problema podem ser usados métodos de regularização [31]. É esta regularização que permite diferenciar o XGBoost de outros algoritmos de *gradient boosting*. Alguns dos métodos implementados são:

- Restrições das DT: estas podem ser feitas de várias formas. É de salientar que quanto mais restrita a formação de DT, maior será o número de algoritmos “fracos” necessário. Pode-se restringir o número de DT (quanto menos DT, mais difícil será encontrar *overfitting*), a profundidade da DT, número de nós e folhas (estas últimas duas restrições visam a limitação do tamanho das DT, para que estes algoritmos não se tornem demasiados complexos – permitem reduzir o *overfitting*), o número de observações por nó (apresenta um valor mínimo de dados presentes num nó, antes que se possa aumentar o tamanho da DT), entre outros.
- Peso das contribuições: pode ser conseguido com um fator denominado de ritmo de aprendizagem (entre 0 e 1). Este visa diminuir o peso da contribuição que

cada DT tem para reduzir o erro das previsões, deixando espaço para DT futuras contribuírem para melhorar o modelo. Esta redução permite diminuir o efeito do *overfitting*. Por outro lado, quanto menor o ritmo de aprendizagem, mais tempo levará a concluir o processo de aprendizagem pelo algoritmo.

- Amostragem aleatória: são extraídas a partir do conjunto de dados de treino, subamostras de forma aleatória e “egoísta” para treinar os algoritmos “fracos”. Esta característica permite diminuir a relação e dependência entre as várias DT.
- Aprendizagem penalizada: introduz um termo adicional que penaliza a complexidade do algoritmo desenvolvido. Consegue-se desta forma contribuir para uma maior atenuação do *overfitting*.

No entanto, as maiores valências do algoritmo são a sua velocidade de execução, que é, pelo menos, 10 vezes maior que outros algoritmos populares, e também a possibilidade de ser escalado para vários problemas de diferentes naturezas, que é possível devido à otimização do próprio algoritmo para o efeito [32].

Hiperparâmetros de um algoritmo

Um conceito importante quando se discute a qualidade de um algoritmo de ML é o de hiperparâmetros. Este conceito é diferente dos parâmetros de um algoritmo, que são definidos pelo próprio ao tentar assimilar a informação proveniente dos dados que lhe são fornecidos. De facto, estes parâmetros não podem ser controlados pelo programador. Em contrapartida, os hiperparâmetros são definidos de forma prévia pelo programador e controlam a forma como o algoritmo se comporta na fase de aprendizagem. Por isto mesmo, os hiperparâmetros estão diretamente relacionados com a qualidade das previsões de um algoritmo. Deste modo, encontrar os seus valores ótimos é algo de extrema importância para não só atingir as melhores previsões possíveis, mas também conseguir atingir uma convergência de resultados em tempo útil. Estes hiperparâmetros variam de algoritmo para algoritmo, pelo que os que devem ser considerados para ANN não são os mesmos que devem ser estudados em DT, já que estes algoritmos têm arquiteturas e formas de processar a informação que lhes é alimentada diferentes. Deste modo, também os seus valores variam de acordo com as suas funções. Há hiperparâmetros que são definidos como variáveis discretas e outros como variáveis contínuas [34]. Alguns hiperparâmetros que se podem associar a DT e a ANN são, por exemplo:

- DT: Profundidade da árvore; número de amostras/instâncias que são necessárias para dividir um nó da árvore em mais folhas; número mínimo de amostras/instâncias num nó da árvore; entre outros.
- ANN: Número de camadas (define a capacidade de aprendizagem de modelos mais complexos); número de neurónios; função de ativação (introduz não linearidades na rede neuronal); função de perda; entre outros.

2.2.2 Design experimental

Um design experimental (DOE) é um método estatístico que conduz a realização de experiências de forma mais eficaz, permitindo a obtenção de informações sobre a influência de

certos parâmetros para um processo específico, com um número relativamente reduzido de experiências. Para aplicações em engenharia, este instrumento é de extrema relevância, uma vez que permite a geração de conhecimento sobre o processo em estudo a ritmos mais elevados que os métodos tradicionais, como por exemplo, o estudo individual de cada parâmetro e da sua influência de forma separada dos outros parâmetros [20]. Para além disso, este método pode permitir a identificação do conjunto de valores ótimos de cada parâmetro para a concretização do processo, ao contrário dos métodos tradicionais.

Um design experimental, na sua base, pode ser subdividido em quatro passos distintos. O primeiro passa por definir um conjunto de fatores; o segundo passo consiste na escolha de um número de níveis para esses fatores; no terceiro passo define-se um conjunto de combinações de níveis de fatores; por fim, no último passo são levadas a cabo as experiências de acordo com as combinações que foram previamente definidas. Neste contexto, um fator é uma variável independente, que representa um parâmetro que pode ter influência num determinado processo e um nível representa um valor definido para esse fator específico. Sabe-se que para um certo conjunto de fatores X e um certo número de níveis Y , o design experimental obtido poderá apresentar Y^X combinações, isto é, experiências possíveis. Este tipo de design experimental é conhecido como design fatorial e permite criar conhecimento no que diz respeito aos fatores que mais influenciam um determinado processo. Os resultados deste tipo de metodologia podem ser integrados com outras metodologias, como a *response surface methodology* (RSM), que visa obtenção de uma configuração ótima de parâmetros para a sua produção através desse mesmo processo. Devido a estas características a RSM é a principal metodologia associada ao design experimental, no que toca a experimentação industrial. No entanto, existem outras metodologias que também podem ser utilizadas. Um exemplo, paralelo à RSM, é o *robust parameter design* (RPD). Este visa a subdivisão dos fatores do processo em dois conjuntos. São estes os fatores de controlo, os quais podem ser controlados e os fatores de ruído que podem controlados num contexto experimental, mas não num contexto de produção. O objetivo de uma metodologia RPD visa a construção de uma configuração de fatores indiferente ao ruído, permitindo uma minimização da variância induzida pelos fatores de ruído. Um exemplo deste tipo de metodologia é o método de Taguchi [20].

Deste modo, dependendo da metodologia aplicada, um design experimental pode ser instrumentalizado para: perceber quais são os fatores relevantes num processo específico (utilizando um design fatorial), otimizar esse conjunto de fatores (para o qual se utiliza a RSM) ou tentar eliminar a variância no processo de produção criado por fatores de ruído.

Método de Taguchi

Como já foi referido anteriormente, o método de Taguchi inclui-se no conjunto de métodos considerados para um RPD. Este é geralmente implementado em contextos de engenharia quando se pretende investigar quais os efeitos que diferentes variáveis têm num certo processo e nos resultados que são produzidos [35]. Este método encontra-se subdividido em vários passos distintos. Inicialmente deve definir-se um valor objetivo (máximo, mínimo, ...) para a medida de performance do processo. Este passo define aquele que deve ser o objetivo do processo e permite a definição da função de perda. De

seguida, devem ser definidos os parâmetros que afetam o processo e criar uma listagem ortogonal para implementar o design experimental (o tamanho desta lista dependerá do número de variáveis que afetam o processo e do número de níveis a considerar para cada variável). Uma vez construída a lista devem ser efetuadas as experiências presentes na lista, de forma a recolher informação sobre o efeito das várias variáveis sobre o resultado final. Por fim, segue-se a análise estatística que pretende determinar o efeito dos diferentes parâmetros na medida de performance do sistema [35]. Para conseguir obter informações sobre o processo que está a ser desenvolvido, o método de Taguchi prevê a implementação de uma função de perda. Esta está diretamente relacionada com o erro entre o valor pretendido para a medida de performance do sistema e o valor real obtido experimentalmente, sob certas condições. Esta função é dada por:

$$l(y) = k \times (y - t)^2, \quad (2.10)$$

e por

$$k = \frac{C}{\Delta^2}. \quad (2.11)$$

Na equação 2.10, k representa uma constante que pode ser determinada a partir da equação 2.11, onde Δ representa o intervalo aceitável e C os limites das especificações. Na equação 2.10, t representa o valor pretendido para a medida de performance e y o seu valor experimental [35]. Sendo o objetivo do método de Taguchi reduzir os custos de um certo processo, este é conseguido através da minimização desta função de perda. Esta função é testada com vários ensaios que são planeados, como já se viu, através de uma listagem ortogonal. Para se definir esta listagem, em conjunto com os vários níveis das variáveis a serem estudadas, é necessário um conhecimento aprofundado do processo em questão. Só assim se conseguem realizar experiências dentro de valores mínimos e máximos admissíveis das variáveis em questão. Dependendo do tamanho do intervalo entre os valores máximo e mínimo admissíveis para uma certa variável, os níveis que são definidos devem ser mais ou menos numerosos. Para além deste intervalo, deve-se considerar também o custo de um ensaio. Conhecidas as variáveis e o número de níveis de cada variável é possível definir o tamanho da listagem ortogonal dos ensaios a realizar [35]. Esta deve garantir que todas variáveis são testadas de forma equitativa. Uma vez concluídas as experiências planeadas, é possível perceber os efeitos relativos de cada variável sobre o processo, na medida de performance. No método de Taguchi, este passo concretiza-se na descoberta do fator S/N (*signal to noise*). Este pode ser calculado de acordo com diferentes considerações, nomeadamente as de que se deve minimizar o valor da característica a ser medida (*smaller the better* ou STB), de que se deve maximizar o valor da característica a ser medida (*larger the better* ou LTB), ou de que esta deve ter um valor específico (*nominal the best* ou NTB). As equações que dão o valor desta variável são dadas de seguida:

$$SN_i = 10 \times \log \left(\frac{\bar{y}_i^2}{s_i^2} \right) \quad (2.12)$$

$$SN_i = -10 \times \log \left(\sum_{u=1}^{N_i} \frac{y_u^2}{N_i} \right) \quad (2.13)$$

$$SN_i = -10 \times \log \left(\frac{1}{N_i} \sum_{u=1}^{N_i} \frac{1}{y_u^2} \right) \quad (2.14)$$

$$\bar{y}_i = \frac{1}{N_i} \sum_{u=1}^{N_i} y_{i,u} \quad (2.15)$$

$$s_i^2 = \frac{1}{N_i - 1} \sum_{u=1}^{N_i} (y_{i,u} - \bar{y}_i)^2 \quad (2.16)$$

Na equação 2.12, que apresenta o rácio S/N NTB de uma certa experiência i , os termos \bar{y}_i e s_i são dados respetivamente pelas equações 2.15 e 2.16. Por outro lado, as equações 2.13 e 2.14 apresentam respetivamente o rácio S/N STB e LTB de uma certa experiência i . Nestas equações 2.12, 2.13 e 2.14, u representa o número de ensaio de uma certa experiência e N_i representa o número de ensaios realizados para uma única experiência. Depois de calculados os valores de rácio S/N para cada experiência, estes valores são definidos para cada variável em estudo e para cada nível desta mesma variável. Sabendo estes valores é possível retirar elações sobre os efeitos que cada variável tem sobre o processo. Estas elações são feitas com base na dispersão do valor de S/N (S/N máximo menos S/N mínimo), sendo que, quanto maior este valor maior o efeito que a variável terá no valor final da medida de performance [35].

2.3 Trabalhos relacionados

Nesta secção serão apresentados alguns trabalhos que foram estudados, e que de alguma forma inspiraram a criação da solução proposta no capítulo 3. Este subcapítulo terá um maior enfoque sobre os avanços da inteligência artificial no campo da soldadura por transmissão de laser e também, com menor enfoque, noutros processos industriais. De facto, têm sido bastantes os progressos que têm sido atingidos pela comunidade científica nos últimos anos, no que diz respeito à soldadura por transmissão de laser. A capacidade de modelar e simular os vários fenómenos associados ao processo, conseguem auxiliar a seleção dos parâmetros que melhor se adequam a um caso específico, prever falhas e a entender melhor as relações não lineares e complexas entre os diferentes parâmetros, sem ser necessário recorrer a experiências físicas [36]. Pode, por exemplo, considerar-se o efeito do método de elementos finitos e outras abordagens de modelação numérica na obtenção de soluções aproximadas para problemas complexos. No contexto do seu uso na soldadura por transmissão de laser, estas são usadas por exemplo para determinar a zona afetada pelo calor, simular a transferência de calor entre peças, entre outros [6]. Muitas vezes, no entanto, verifica-se a existência de um limite prático do poder de modelação do processo. Este assenta no facto de muitos dos efeitos associados ao processo serem não determinísticos o que dificulta a sua modelação teórica [6]. Como alternativa, pode recorrer-se a abordagens de ML, que utilizam dados experimentais sobre o processo para conseguir fornecer respostas sobre o mesmo, sem existir um completo conhecimento teórico sobre todas as suas variantes [6]. Dentro desta abordagem podemos encontrar os mais variados avanços no contexto da soldadura por transmissão de laser, sendo os mais relevantes aqueles que lidam com o tratamento dos parâmetros do processo, tais como a otimização dos parâmetros do processo e a previsão de defeitos nas peças [6,28,36]. Deste modo, o subcapítulo seguinte irá ter um grande enfoque nestes dois tópicos associados à

abordagem de ML. Serão também estudadas outras alternativas a esta abordagem, que também permitem otimizar parâmetros de processos e prever defeitos de peças, e como estas podem complementar a ML. Por fim, será também feita uma referência a outros processos produtivos e aos avanços que têm sido atingidos no contexto de ML e ajuste fino de parâmetros de produção.

2.3.1 Otimização da soldadura por transmissão de laser

No que diz respeito a ML e a sua ligação com os processos de soldadura a laser, podemos encontrar vários trabalhos relevantes na literatura. Mehrpouya et. al. [10] desenvolveram um algoritmo baseado em ANN para determinar a força de separação e alongação de rotura. Ainda que no contexto da tentativa de descobrir qual a melhor composição para uma mistura PET/PEVA, este trabalho apresenta bons resultados na previsão da qualidade da união de duas peças pelo processo de soldadura por transmissão de laser. Foram estudados dois algoritmos separados. O primeiro é um algoritmo baseado num *multi-layer perceptron* e o segundo baseado em ANN de funções com base radial. Ambas as ANN estavam associadas ao algoritmo de *backpropagation* e a cinco *hidden layers*. Para se obterem os resultados pretendidos foram usadas como variáveis de entrada nos algoritmos a velocidade de soldadura e potência do laser. O conjunto de dados foi dividido em três subconjuntos. O subconjunto de dados de treino corresponde a setenta por cento dos dados totais, enquanto que os subconjuntos usados para implementar a *cross-validation* (que avalia o erro médio e o coeficiente de correlação) e teste do algoritmo correspondem ambos a quinze por cento do conjunto total de dados. Na figura 2.12 conseguimos visualizar os resultados que foram obtidos pelo teste do algoritmo:

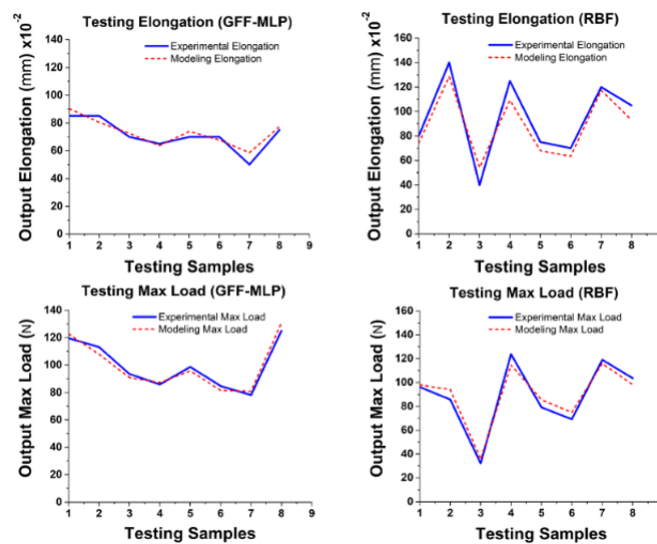


Figura 2.12: Resultados do teste dos algoritmos baseados no *multi-layer perceptron* (à esquerda) e em ANN de funções de base radial (à direita), tanto para a alongação na rutura (em cima), como para a força de separação (em baixo) [10].

Verificou-se que o algoritmo baseado no *multi-layer perceptron* apresentava melhores resultados, isto é, com erros menores dados pelo processo de *cross-validation*. Concluiu-se

no trabalho que o *multi-layer perceptron* é uma boa ferramenta de previsão, conseguindo obter uma exatidão de 97% no processo de *cross-validation* (um valor superior ao verificado com o outro algoritmo a ser avaliado).

Outro trabalho desenvolvido na área da soldadura por transmissão de laser em termoplásticos foi desenvolvido por Acherjee et al. [11]. Neste artigo, foram consideradas quatro variáveis de entrada no algoritmo (a potência, a velocidade de soldadura, distância entre peças e força de compactação) de forma a conseguir-se prever o desempenho de uma junta soldada, que se materializa, neste caso, em previsões da força de separação das peças e largura do cordão de soldadura. Para esse efeito foi usado um algoritmo baseado em ANN *feedforward* com *backpropagation*. Como se sabe, pelo que é referido anteriormente, o desempenho de um algoritmo depende dos hiperparâmetros que são escolhidos. Para as ANN, alguns dos hiperparâmetros mais relevantes são o número de *hidden layers* e o número de nós que as compõem. Para averiguar qual a melhor combinação destes hiperparâmetros o autor testou a performance de diferentes combinações de números de nós e de *hidden layers*, selecionando-se aquela combinação com o menor erro médio de previsão. Para todos os casos, não se utilizou uma função de transferência para a camada de entrada, enquanto que nas outras camadas se usaram funções de transferência sigmóides. Um esquema da ANN escolhida pode ser visualizado na figura 2.13:

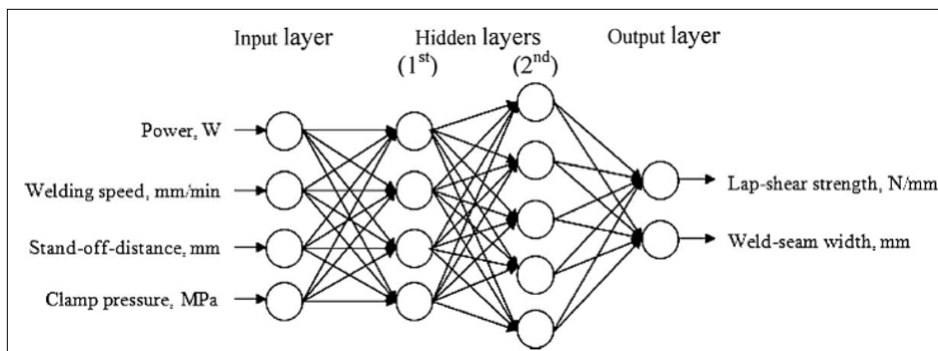


Figura 2.13: ANN utilizada na previsão da força de separação e largura do cordão de soldadura, com recurso à potência do laser, velocidade de laser, distância entre peças e força de compressão [11].

É efetuada também uma divisão entre dados de treino e dados de teste. Neste caso, o autor optou por utilizar 20 dos 26 registos totais para treinar o algoritmo e os outros 6 registos para testar o algoritmo. Durante a fase de treino os parâmetros do algoritmo são alterados de forma a minimizar o erro médio das previsões. Para além disso, o ritmo de aprendizagem é alvo de uma experiência para averiguar qual o melhor valor inicial deste hiperparâmetro no processo de treino. Na experiência foram escolhidos diferentes valores do ritmo de aprendizagem mantendo-se os restantes hiperparâmetros constantes. No fim, selecionou-se o ritmo de aprendizagem que minimizava o erro médio das previsões. Uma vez treinado o algoritmo, consegue-se utilizar os valores do subconjunto de teste para averiguar a adequação das previsões que se conseguem realizar. Demonstrou-se um erro médio de aproximadamente 4% e 5% para a força de separação e largura da junta, respetivamente. Conclui-se, portanto, que o algoritmo elaborado era capaz de produzir previsões sobre a qualidade de uma peça soldada por transmissão de laser com um grande

grau de fiabilidade.

Como complemento à solução apresentada anteriormente existe o artigo publicado por Nakhaei et al. [12], que mais uma vez tenta utilizar um algoritmo de ANN para prever a qualidade de uma junta soldada. Tal como em [11], foram definidas como variáveis de entrada no algoritmo a potência do laser, a velocidade do laser, a distância entre as peças e a força de compactação, para obter neste caso a força de separação das peças. Para se conseguir essa previsão foi utilizada uma ANN *feedforward* com *backpropagation*. Neste caso, considerou-se apenas uma *hidden layer*, utilizando-se 10 nós nessa mesma. Não foi utilizada uma função de transferência nos nós de entrada, mas foi utilizada uma função sigmóide na *hidden layer* e uma função linear na camada de saída. Para o treino do algoritmo foram utilizados 80% dos 25 registos disponíveis, enquanto que os restantes foram utilizados para testá-lo. Com esta configuração, encontrou-se um erro de previsão de aproximadamente 2%, apresentando-se ótimos resultados na previsão da força de separação da junta. De seguida podem ser visualizados na figura 2.14 os resultados obtidos com os registos utilizados para o teste do algoritmo:

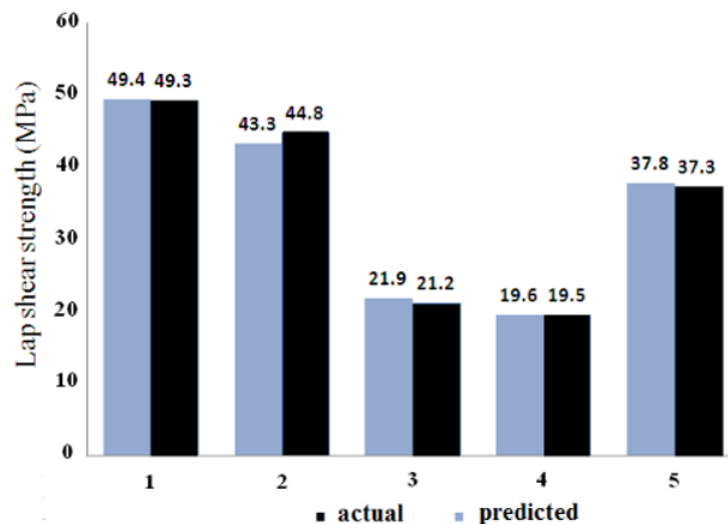


Figura 2.14: Comparação entre o valor de força de separação previsto pelo algoritmo com os valores reais [12].

Como se pode constatar pelos artigos anteriormente expostos, a relação entre os parâmetros de produção associados ao processo de soldadura por transmissão de laser e a força de separação das peças está bem estabelecida, conseguindo-se realizar previsões sobre esta última variável com graus de exatidão notáveis. Para se realizarem estas previsões são utilizadas em todos os trabalhos ANN, que recorrem aos algoritmos *feedforward* e *backpropagation* para otimizar as previsões efetuadas. Geralmente as ANN construídas não são muito complexas, não contendo mais do que duas *hidden layers* e 10 nós em cada camada individual. No entanto, há que ter em conta que nestes estudos não é levada em linha de conta a influência que certos empenos nas peças possam ter na qualidade final do produto. O acréscimo desta variável poderá levar a maiores complicações nas previsões obtidas e exigir a construção de algoritmos mais complexos.

Para além das ANN, uma possível alternativa para este tipo de abordagens é a utilização de métodos estatísticos como o método de Taguchi. O estudo da sua performance na previsão de parâmetros de qualidade de uma peça no âmbito da soldadura por transmissão de laser é bem conhecido [37,38], ainda que apresente uma eficácia mais reduzida que as ANN. No entanto, este também pode ser usado como complemento às ANN. Uma tal abordagem foi utilizada, no âmbito da soldadura a laser por Olabi et al. [39]. Neste estudo é implementada uma combinação do método de Taguchi com um algoritmo baseado em ANN. Na verdade, são usados dois algoritmos distintos. O primeiro é utilizado como um substituto à realização de ensaios reais de soldadura, para complementar o conjunto de dados (baseado em dados de uma experiência anterior àquela que é explanada no estudo) a utilizar na previsão de parâmetros ótimos que não apresentava todas as combinações exigidas pelo design fatorial escolhido (L9 do método de Taguchi). Nesta previsão são definidas as combinações dos níveis de potência do laser, velocidade de soldadura e distância focal que faltavam na matriz de experiências e obtiveram-se as previsões de penetração de fusão e penetração da zona afetada pelo calor. Para a ANN em questão foram usadas duas *hidden layers*, cada uma com 21 nós, que auferiram ao algoritmo uma capacidade excelente de generalizar o conhecimento que lhe foi transmitido para outros dados. Este tipo de metodologia permite eliminar a variabilidade que se produziria com a realização de dois conjuntos de ensaios distintos. Uma vez concluído este processo foi possível determinar a partir do rácio S/N qual o conjunto de níveis de parâmetros que produz as previsões ótimas das características do cordão de soldadura pretendidas. Uma vez selecionado este conjunto de parâmetros foi possível construir uma outra ANN que permitisse conhecer as características do cordão de soldadura produzidas com o conjunto ótimo de parâmetros de soldadura. Os autores consideraram os resultados razoáveis e a abordagem como algo que poderia ser replicado noutros estudos da soldadura a laser.

2.3.2 Otimização de outros processos produtivos

Num estudo, desenvolvido por Fu et al. [13], verificou-se que durante o processo de união de tubos de PVC, a alteração das condições ambientais exteriores previstas influenciava a contração ou expansão térmica do material, que pode levar à existência de defeitos nos tubos, quando unidos uns com os outros. Este tipo de situação, também acaba por induzir danos excessivos nos equipamentos responsáveis, sendo a substituição constante de peças nos aparelhos uma realidade insustentável para muitas pequenas e médias empresas. Estas dependem, em grande parte de trabalhadores experientes para efetuar as alterações atempadas dos parâmetros do processo de acordo com as condições exteriores. Este tipo de trabalho requer vários anos de experiência e é muito baseado em conhecimento obtido por tentativa-erro. Neste contexto, viu-se uma grande importância em encontrar um método, que a partir das condições exteriores consiga definir os parâmetros de processo que garantam que não existem erros na produção dos tubos de PVC. Neste estudo, desenvolveu-se um algoritmo com recurso a uma ANN, com *backpropagation*, integrado com o método de Taguchi, usado para se obter uma combinação ótima de hiperparâmetros do algoritmo (metodologia RPD de design experimental). Neste caso, os parâmetros que foram considerados para o processo são os apresentados na figura 2.15. Para que se facilitasse a obtenção de uma exatidão satisfatória e maximizasse a velocidade de aprendizagem do algoritmo, teve que se normalizar os parâmetros em questão:

Inputs			Output
X_1	Environment temperature		
X_2	Rotating speed of soft plastic extruder		
X_3	Rotating speed of hard plastic extruder	Y	Pulling speeds by the haul unit
X_4	Rotating speed of winding forming unit		
X_5	Rotating speed of forward hose movement		

Figura 2.15: Parâmetros relevantes para o processo de união de tubos PVC [13].

O método de Taguchi, neste contexto, permitiu o cálculo dos erros de previsão para 16 estruturas diferentes, considerando-se na matriz L16 do design experimental, 4 níveis para cada um dos 5 hiperparâmetros das ANN a controlar. A partir destes valores obteve-se um erro médio, que permitiu o cálculo do rácio S/N. A experiência que apresentou o maior S/N, foi a experiência 12, pelo que uma combinação ótima de parâmetros é aquela que foi usada nessa mesma experiência.

Level	Neuron number of hidden layer (A)	Number of learning (B)	Learning rate (C)	Momentum factor (D)	Testing ratio (E)
Level 1	2	500	0.1	0.1	37 (1/2)
Level 2	3	1000	0.4	0.4	25 (1/3)
Level 3	6	2000	0.7	0.7	19 (1/4)
Level 4	12	4000	0.9	0.9	15 (1/5)

Figura 2.16: Níveis controlados associados a cada parâmetro relevante [13].

Experiment	Neuron number of hidden layer (A)	Number of learning (B)	Learning rate (C)	Momentum factor (D)	Testing ratio (E)
1	1	1	1	1	1
2	1	2	2	2	2
3	1	3	3	3	3
4	1	4	4	4	4
5	2	1	2	3	4
6	2	2	1	4	3
7	2	3	4	1	2
8	2	4	3	2	1
9	3	1	3	4	2
10	3	2	4	3	1
11	3	3	1	2	4
12	3	4	2	1	3
13	4	1	4	2	3
14	4	2	3	1	4
15	4	3	2	4	1
16	4	4	1	3	2

Figura 2.17: Design experimental utilizado [13].

Existe, no entanto, outra possível combinação ótima de hiperparâmetros. Esta é dada pelos níveis dos hiperparâmetros onde o rácio S/N é, em média, mais elevado. Para saber qual, dentro destas duas opções, é a melhor, foi efetuado um teste com 3 experiências diferentes. O S/N de cada um destes foi maior que o S/N da 12^a experiência do teste anterior. Desta forma, concluiu-se que a 12^a experiência do design L16 não representaria a combinação ótima de hiperparâmetros. Esta é dada pela outra combinação possível.

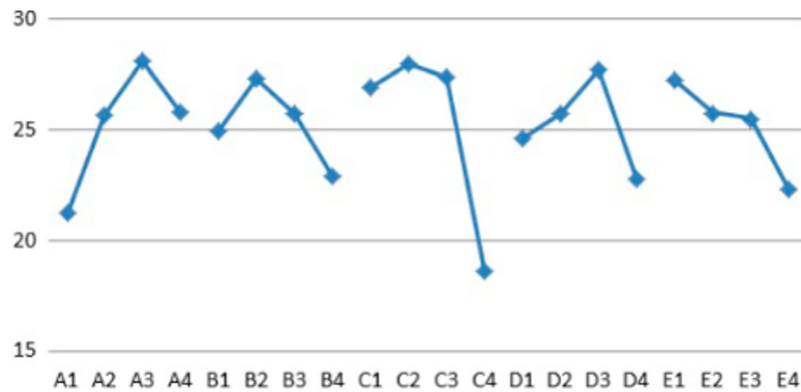


Figura 2.18: Visualização gráfica do valor médio de S/N (eixo vertical) para cada nível controlado de cada parâmetro considerado (eixo horizontal) [13].

Experiments	Training RMSE	Testing RMSE	Average	Standard deviation	S/N ratio
1	0.036142	0.0403	0.038221	0.00294	28.353959
2	0.037475	0.037853	0.037664	0.000267	28.481471
3	0.034443	0.043154	0.038799	0.00616	28.223697

Figura 2.19: Resultados das experiências realizadas com o conjunto de níveis que em média maximizam o valor de S/N [13].

De seguida foi efetuado um teste ANOVA, para perceber quais as contribuições dos hiperparâmetros para a performance da ANN. Uma vez realizado este teste, foram calculados os erros de previsão do algoritmo desenvolvido tanto para o conjunto de dados de treino e teste, tendo sido obtidos erros de aproximadamente 4%. Por fim, foi verificado o impacto da combinação ótima de hiperparâmetros da ANN, conseguidos através do método de Taguchi, através da comparação da previsão da variável de saída do algoritmo (*pulling speed*), com o valor pretendido dessa variável, em 6 experiências diferentes. Em todos os casos, o erro foi menor que 1%, e por isso a exatidão foi superior a 99%. Concluíram os autores, que o modelo desenvolvido é muito exato, conseguindo desta forma alterar automaticamente e satisfatoriamente a *pulling speed*, tendo em conta as condições de processamento que são impostas.

Experiment	Environment temperature X_1	Rotating speed of soft plastic extruder X_2	Rotating speed of hard plastic extruder X_3	Rotating speed of winding forming unit X_4	Rotating speed of hose moving X_5	Actual value of pulling speeds by the haul unit	Predicted value of pulling speeds by the haul unit	Error (%)
1	27.5	1357	1245	1031	689	383	383.0645	0.02
2	28.8	1358	1242	1032	689	380	383.1721	0.83
3	24.8	1360	1240	1055	701	384	385.2488	0.32
4	26.1	1364	1241	1056	708	386	386.0306	0.01
5	21	1483	1351	1203	838	432	434.2373	0.52
6	22	1400	1380	1198	836	431	434.67	0.84

Figura 2.20: Ensaios de teste conduzidos [13].

De uma forma geral, o trabalho desenvolvido neste estudo é bastante informativo e apresenta uma solução com grande relevância para o mundo industrial. Este serviu de fonte de inspiração para o desenvolvimento do projeto exposto neste documento, especialmente no que toca à otimização de hiperparâmetros de um algoritmo com recurso ao método de Taguchi, no caso da soldadura por transmissão de laser.

No âmbito do projeto Tooling4G, foi desenvolvida por Prates et al. uma plataforma *open-source* que permite a monitorização e controlo inteligente do processo de moldação por injeção [14]. Para compreender melhor este projeto, foi essencial o auxílio do Professor Pedro Prates, que fez parte da equipa que o realizou. Tal como o nome sugere, pretendia-se garantir que os parâmetros que são alimentados ao equipamento não sejam indutores de defeitos na peça. Desta forma, caso se verifique que os valores lidos nos sensores estão fora dos valores previstos para a produção de uma peça, deve-se realizar o ajuste fino de parâmetros de processo de forma a evitar a existência de defeitos no produto. Este processo também depende em grande parte do sistema de controlo de qualidade das peças, que permite obter informações sobre como os parâmetros influenciam a existência de defeitos nas mesmas. A integração de todos estes conjuntos de informação permitiu a criação de um algoritmo preditivo, desenvolvido em Python, que a partir dos valores de parâmetros de produção consegue prever a qualidade final da peça. Determinou-se que a qualidade da peça estava relacionada com o seu peso e com uma medida característica “p8”. A partir das previsões destes parâmetros que são obtidas pelo algoritmo preditivo, consegue-se alterar os parâmetros do processo de forma a corrigir eventuais anormalidades. Para perceber como estas alterações devem ocorrer desenvolveu-se um design experimental baseado em manipulação de resultados, que permitiu construir relações matemáticas entre os vários parâmetros do processo e parâmetros de qualidade e criar uma base de dados com os dados de entrada e saída do algoritmo preditivo. Alguns resultados dessas experiências, que refletem as relações entre as várias variáveis a considerar encontram-se na figura 2.21:

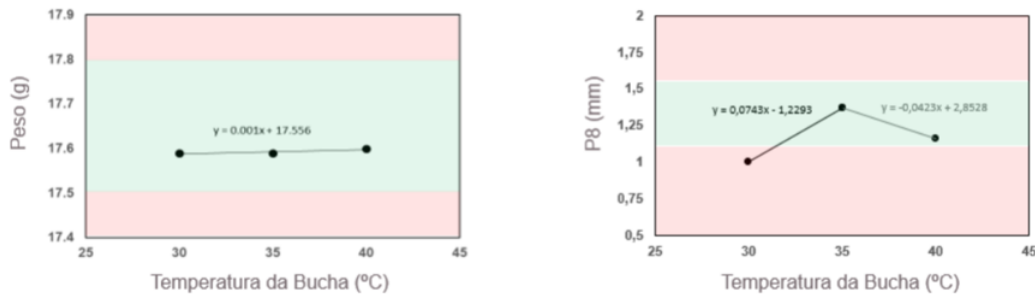


Figura 2.21: Relação entre a temperatura da bucha e as variáveis de qualidade a controlar [14].

No total foram desenvolvidos três algoritmos distintos, conseguindo-se uma maior fiabilidade na previsão de parâmetros de qualidade da peça com o algoritmo *random forest*. Este conseguiu prever corretamente entre 86 e 96% das ocorrências de defeitos nos produtos. Este projeto, apresenta uma abordagem holística e completa na construção de um sistema de controlo e ajuste fino de parâmetros de produção, com o desenvolvimento de uma interface gráfica e sistema de monitorização do equipamento.

Por fim, deve referenciar-se o trabalho desenvolvido por Hoang et al. [15], onde se realizou, com recurso a ANN, o ajuste fino de parâmetros do processo de maquinaria CNC. À semelhança do processo de soldadura por transmissão de laser, a relação entre parâmetros de processo e parâmetros de qualidade da peça são desconhecidos devido à grande complexidade e não linearidade na relação entre as grandezas em jogo. Por forma a adquirir conhecimento sobre o processo foi executado um teste ANOVA. Este foi realizado após a concretização de 11 ensaios onde se estudou o efeito da velocidade de corte, velocidade de avanço e profundidade de corte radial sobre a rugosidade superficial da peça. Cada uma das variáveis de entrada apresentava 3 níveis distintos que foram considerados durante os testes. Os resultados da aplicação do teste ANOVA podem ser visualizados na figura 2.22:

Regression statistics					
Multiple R	0.956340479	-	-	-	-
R square	0.937205092	-	-	-	-
Standard error	0.125806478	-	-	-	-
Observations	11	-	-	-	-
ANOVA					
	DF	SS	MS	F	Significance F
Regression	3	0.148101838	0.049367279	5.119028745	0.007348675
Residual	7	0.110794412	0.015827773	-	-
Total	10	0.258896250	-	-	-
	Coefficients		Standard error		
Intercept	-1.937080761	1.872516072	-	-	-
Variable 1	-0.302270348	0.186072594	-	-	-
Variable 2	0.382423871	0.186106481	-	-	-
Variable 3	0.057248255	0.036951295	-	-	-

Figura 2.22: Resultados da aplicação do teste ANOVA [15].

Como pode ser verificado pela imagem anterior, conseguiu-se estabelecer um modelo matemático para a relação pretendida, que prevê o comportamento do processo com algum grau de confiança. O próximo passo do estudo foi determinar através de um algoritmo preditivo o desgaste da ferramenta. De facto, é conhecida a relação que este tem com a rugosidade superficial da peça e consequentemente com a sua qualidade.

Para implementar este passo foi utilizada uma ANN otimizada através de *backpropagation* com uma *hidden layer* com 10 nós. Foram utilizados 6 parâmetros de entrada no algoritmo, entre os quais o desgaste inicial da ferramenta. A configuração desta ANN pode ser visualizada na figura 2.23. Os valores que resultaram das previsões realizadas com o algoritmo em questão, que estão associadas a um erro médio de 4%, podem ser visualizados na figura 2.24:

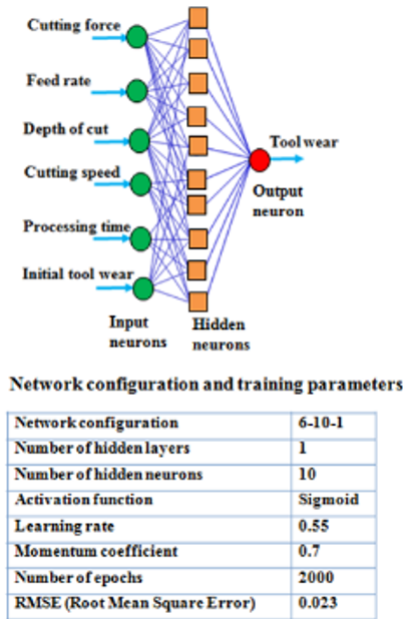


Figura 2.23: ANN usada na previsão do desgaste da ferramenta [15].

Test no.	Cutting force (N)	Feed rate (mm/min.)	Radial cutting depth (mm)	Processing time (min.)	Cutting speed (m/min.)	Initial tool wear (mm)	Actual tool wear (mm)	Predicted tool wear (mm)	Error (%)
1	214.90	2357	0.1	3.5	370	0	0.017	0.016	5.88
2	172.26	2357	0.1	3.5	595	0	0.039	0.037	5.12
3	272.68	3790	0.1	5.4	370	0	0.051	0.049	3.92
4	618.07	2357	0.95	8.7	595	0	0.102	0.099	2.94
5	831.50	3790	0.95	13	370	0	0.161	0.155	3.72
6	653.19	3790	0.95	13	595	0	0.267	0.260	2.62

Figura 2.24: Previsões do desgaste da ferramenta [15].

De seguida, a partir dos valores previstos de desgaste da ferramenta são calculados novos valores de velocidade de avanço da ferramenta, com recurso a uma nova ANN. Este tipo de informação é importante para perceber se é necessário efetuar uma troca de ferramenta ou se basta apenas fazer o ajuste fino dos parâmetros para evitar a produção de uma peça defeituosa. A configuração desta ANN pode ser percecionada através da figura 2.25:

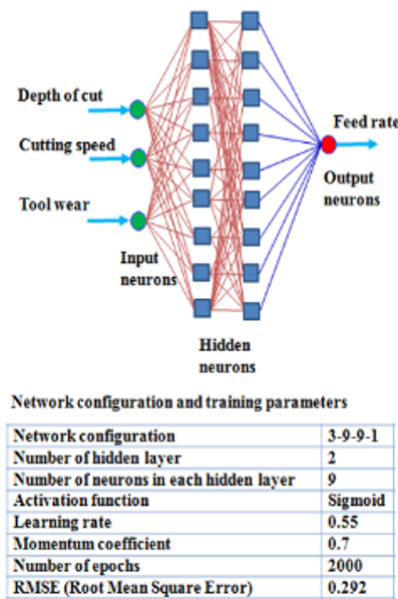


Figura 2.25: ANN usada para prever o valor de velocidade de avanço [15].

Para além do modelo preditivo de ajuste fino de parâmetros (neste caso da velocidade de avanço), fizeram também parte do sistema desenvolvido a criação de uma interface gráfica e a introdução da informação referente aos ensaios numa base de dados. Tal como o trabalho desenvolvido por Prates et al., este projeto toma uma abordagem mais holística sobre todo o processo de controlo do equipamento, preocupando-se não só com o ajuste de parâmetros, mas também com a sua atuação no equipamento. No entanto, há que notar o erro relativamente elevado que se encontra associado à previsão do ajuste da velocidade de avanço da ferramenta. Um estudo sobre qual a melhor arquitetura para a ANN, tal como foi realizado por Fu et al., poderia ser altamente interessante como forma de complementar este projeto.

2.3.3 Síntese dos trabalhos relacionados

Em jeito de conclusão, uma vez revisto o estado da arte no que diz respeito à inteligência artificial no processo de soldadura por transmissão de laser, pode-se concluir que o uso de ANN para o efeito é algo bem difundido na literatura e partilhado por quase todos os estudos. Alguns fazem uso de outros métodos, tais como o método de Taguchi, que carecem da exatidão nas previsões de parâmetros de qualidade que são conseguidas com ANN. Uma solução otimizada pode ser conseguida com a integração destes métodos estatísticos com ANN. Na literatura são encontrados alguns exemplos de como esta combinação pode ser frutífera, nomeadamente na obtenção de dados para treinar um algoritmo, verificando-se que o método de Taguchi é geralmente o método estatístico de eleição para combinar com ANN. De notar também, que não existe na literatura nenhum estudo sobre a possibilidade de ajustar os parâmetros de soldadura do equipamento de acordo com o valor da previsão da qualidade da peça. Por outro lado, no que toca à previsão de parâmetros ótimos em outros processos produtivos, têm havido grandes progressos nos últimos anos. De facto, comparativamente com o campo da soldadura por transmissão de laser onde os avanços neste campo são praticamente nulos, verifica-se

que noutros processos produtivos existe um esforço para conseguir ajustar os parâmetros de um processo de forma automática e inteligente. As formas como os algoritmos são treinados e essas previsões são realizadas, por sua vez, também são inovadoras. Desde a integração de métodos estatísticos na otimização dos hiperparâmetros das ANN, até ao uso de algoritmos baseados em DT (*random forest*) para criar as previsões, as abordagens utilizadas são mais variadas do que aquelas que são apresentadas no campo da soldadura por laser. Estas abordagens serviram de inspiração, quando se desenvolveu o projeto apresentado, ajudando a definir o plano de trabalhos.

Na próxima secção deste documento será apresentada a solução proposta para o projeto, que resultou do trabalho desenvolvido, inspirado nos trabalhos apresentados.

Esta página foi intencionalmente deixada em branco.

Capítulo 3

Solução proposta e Implementação

Como foi referido na parte introdutória deste documento, o projeto apresentado pretende aproveitar o conhecimento existente de cada peça, integrado numa base de dados Microsoft SQL Server e a rastreabilidade que deverá ser garantida através de um código de barras associado a cada peça para resolver os problemas relacionados com o processo de soldadura por transmissão de laser que nesta altura persistem. Para uma melhor perceção da solução proposta é apresentado um esquema das principais funções do sistema, na figura 3.1, e onde se resume a forma como este será integrado na produção da Iber-Oleff. Pretende-se, portanto, que os códigos de barras associados a cada produto permitam aceder em tempo real aos parâmetros padrão que realizam a união de um *front frame* e de um *rear frame* através do processo de soldadura por transmissão de laser. Para isso, utilizou-se um ESP8266, que permite a comunicação com um leitor de código de barras e a receção dos números de série das peças em questão. Este módulo comunica com um servidor desenvolvido em C#, que interage com a base de dados Microsoft SQL Server e extrai os parâmetros padrão que realizam a união das peças, mas também outras informações relevantes como o empeno das duas peças. Esta informação é depois reencaminhada para um algoritmo preditivo desenvolvido em Python, que prevê a força de separação das peças, quando unidas com os parâmetros padrão e considerando os seus empenos. O resultado da previsão do algoritmo é encaminhado, em conjunto com todas as informações relevantes sobre os processos desenvolvidos no servidor para uma interface gráfica desenvolvida em Node-Red. Nesta, o operador conseguirá avaliar a previsão do algoritmo preditivo e determinar se os parâmetros padrão devem ser ajustados de forma a prevenir a ocorrência de defeitos na peça final. Para além disso, também se permite a visualização gráfica de dados da base de dados, bem como atualizar registos dessa mesma. Uma vez comunicada a intenção do operador via a interface gráfica, é feita a comunicação final de parâmetros para o autómato programável associado ao equipamento através do servidor, criando-se condições para a união das peças, salvaguardando-se a qualidade do produto final. Deste modo a alteração e ajuste dos parâmetros tem um apoio muito mais robusto e pode ser realizada de forma mais célere, pelo que não será necessário usar-se apenas um único conjunto de parâmetros para produzir todo o mesmo tipo de peças. Este sistema permite flexibilizar a produção das peças, garantido que estas verificam os requisitos mínimos de qualidade. Como o seu nome indica, trata-se de um sistema de atualização automática de parâmetros de produção e como se encontra auxiliado por um algoritmo preditivo, facilita a tomada de decisões mais inteligentes por parte de quem opera o equipamento. Para além de todos estes módulos, existe ainda uma REST API,

desenvolvida em C#, que permite a comunicação de outros equipamentos com a base de dados através do protocolo de comunicação HTTP. Deste modo, facilita-se a integração de toda a informação sobre uma determinada peça e o seu processo de produção. A forma como todas as diferentes partes do sistema interagem encontra-se esquematizada na figura 3.2. De salientar que nesta figura, “Fa” representa a força de separação entre as peças soldadas.

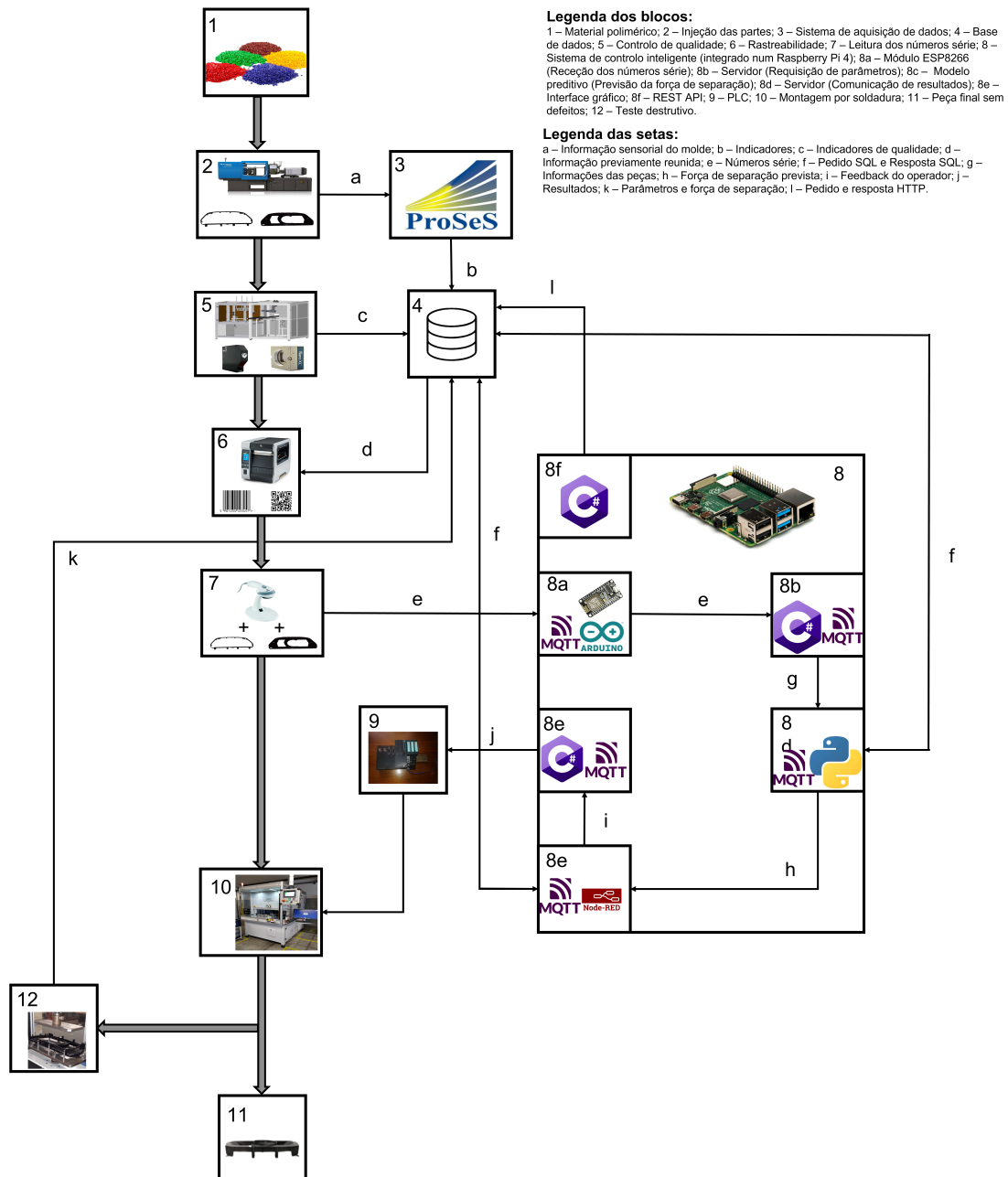


Figura 3.1: Esquema da solução proposta, integrada na linha de produção da Iber-Oleff.

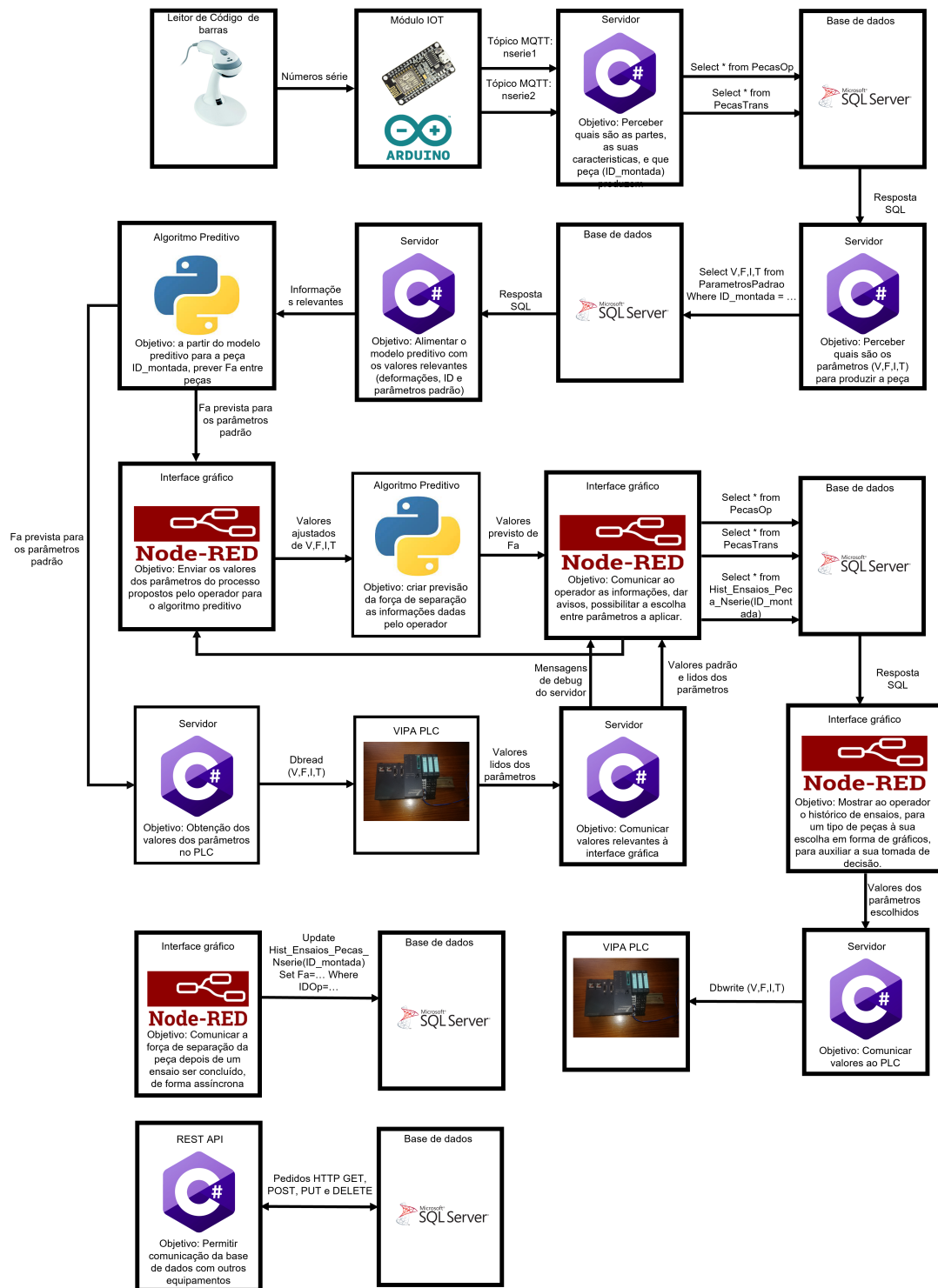


Figura 3.2: Interações e trocas de informação entre os vários módulos da solução proposta.

De forma a permitir uma implementação flexível e duradoura da solução criada, foram criadas imagens virtuais de cada um dos módulos, através do software Docker. Este permite que os programas desenvolvidos sejam corridos em diferentes sistemas operativos e torna o funcionamento do sistema imune a atualizações de software e outras eventua-

lidades que possam de alguma forma interferir com o seu funcionamento. Estas imagens virtuais foram carregadas num Raspberry Pi 4, para que pudessem ser implementadas de imediato na instalação fabril da Iber-Oleff. Nos seguintes subcapítulos serão retratados em pormenor os desenvolvimentos de cada módulo individual que faz parte do sistema retratado: base de dados, ESP8266, servidor C#, algoritmo preditivo, interface gráfica, REST API e autómato (PLC). É necessário, também, materializar essas funcionalidades na sequência pretendida de forma a que o sistema possa ser aplicado num contexto industrial, de produção. Será, por isso, apresentado o método a partir do qual se conseguiu implementar a solução, que permite ler os números de série associados a duas peças particulares, extrair informações da base de dados, alimentar um algoritmo preditivo que auxilia os ajustes a praticar sobre os parâmetros do processo (velocidade de soldadura, força de compactação, intensidade do feixe de laser) e atuá-los sobre o PLC associado ao equipamento de soldadura. Para esse efeito foi usado um Raspberry Pi 4, um equipamento compacto com a capacidade para hospedar as aplicações em questão. Para transferir os módulos para este dispositivo foi utilizado o software Docker, em conjugação com a plataforma de gestão de projetos Balena. Serão dedicados dois subcapítulos à explicação da introdução do sistema de controlo inteligente no Raspberry Pi 4.

3.1 Base de dados com a informação relativa aos *front* e *rear frames*

Como foi explicitado anteriormente, uma vez realizada a injeção das peças poliméricas, estas passam por uma célula de controlo de qualidade que analisa a qualidade dimensional e superficial de cada peça. As informações que são retiradas desta célula são armazenadas numa base de dados Microsoft SQL. Para se conseguir aceder às informações de cada peça específica e da sua produção, garantindo a sua rastreabilidade, definiu-se um código de barras que se associa a cada peça específica. Decidiu-se por isso integrar a existência de códigos de barras associados a cada peça produzida aos parâmetros de produção para o processo de soldadura por transmissão de laser. No entanto, devido à falta de acesso à informação da base de dados da empresa é necessário criar uma estrutura que permita a sua integração/simulação. Para isso foi desenvolvido um exemplo de uma base de dados relacional a usar em conjunto com o sistema de controlo inteligente. O processo de criação da base de dados foi desenvolvido com o auxílio do software SQL Server Management Studio. Esta é composta por quatro relações distintas: uma relação dedicada à descrição de cada peça transparente à radiação laser (*rear frame*), uma relação dedicada à descrição de cada peça opaca à radiação laser (*front frame*), uma relação com os parâmetros padrão necessários para unir um *front frame* a um *rear frame* e por fim uma tabela com o historial de ensaios destrutivos praticados sobre um tipo de peça final, depois de montada. O diagrama relacional entidade-relacionamento destas tabelas encontra-se na figura 3.3.

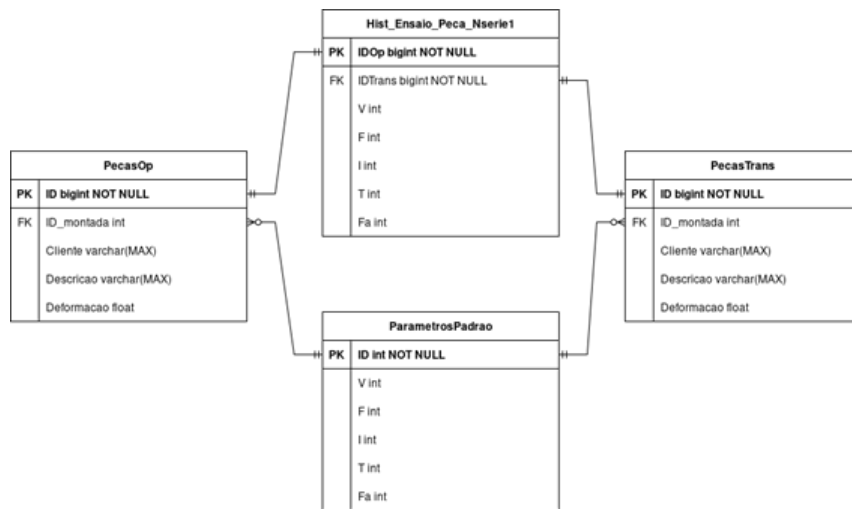


Figura 3.3: Diagrama relacional entidade relacionamento da base de dados criada.

Como se pode verificar, tanto a tabela dedicada à descrição de cada *rear frame* como a tabela dedicada à descrição de cada *front frame* apresentam cinco atributos:

- “ID” – diz respeito ao número série específico da peça. Como um registo deste atributo corresponde unicamente a uma peça específica, este corresponde à chave primária das tabelas;
- “Cliente” – refere qual o nome do cliente;
- “Descricao” – dá uma descrição da peça e o seu propósito;
- “ID_montada” – refere-se à identificação da peça final que pode ser montada com recurso à peça identificada;
- “Deformacao” – dado numérico indicativo da maior distorção dimensional da peça.

Em relação, à tabela dos parâmetros necessários para unir um *rear frame* a um *front frame*, existem 6 atributos:

- “ID” – refere-se à identificação da união entre duas peças (tipo de peça final). Como um registo deste atributo corresponde unicamente a um tipo de união específica, esta corresponde à chave primária das tabelas;
- “V” – refere-se à velocidade de soldadura (parâmetro de processo) padrão para a produção da peça;
- “F” – refere-se à força de compactação (parâmetro de processo) padrão para a produção da peça;
- “I” – refere-se à intensidade do feixe laser (parâmetro de processo) padrão para a produção da peça;
- “T” – refere-se ao tempo de arrefecimento (parâmetro de processo) padrão para a produção da peça;

Esta relação apresenta dois relacionamentos. Estes permitem relacionar vários registos dos atributos “ID_montada” das tabelas “PecasOp” e “PecasTrans” com um único do atributo “ID” da relação “ParametrosPadrao”. Por fim, apresenta-se ainda uma quarta relação, com o historial dos ensaios destrutivos, realizados para uma união entre um *rear frame* e um *front frame*, de uma mesma peça final soldada. Será necessário no futuro criar mais relações deste tipo, de acordo com a realização de ensaios destrutivos a peças montadas diferentes. Esta relação apresenta sete atributos:

- “IDOp” – diz respeito ao número série específico do *front frame* da peça. Este atributo é a chave primária da relação;
- “IDTrans” – diz respeito ao número série específico do *rear frame* da peça;
- “V” – refere-se à velocidade de soldadura (parâmetro de processo) usada na união das duas peças identificadas;
- “F” – refere-se à força de compactação (parâmetro de processo) usada na união das duas peças identificadas;
- “I” – refere-se à intensidade do feixe laser (parâmetro de processo) usada na união das duas peças identificadas;
- “T” – refere-se ao tempo de arrefecimento (parâmetro de processo) usado na união das duas partes identificadas;
- “Fa” – refere-se à força de separação necessária para separar um *front frame* e um *rear frame* particulares.

Esta relação apresenta dois relacionamentos. Estes permitem relacionar um e apenas um registo dos atributos “IDOp” e “IDTrans” com um e apenas um registo dos atributos “ID” das tabelas “PecasOp” e “PecasTrans”.

3.2 Módulo de leitura dos números série de peças (ESP8266)

Uma vez estabelecida a estrutura da base de dados, é possível aceder aos parâmetros de processo que se pretende para uma certa combinação de peças, através do reconhecimento do seu número de série. Este reconhecimento, na Iber-Oleff, é realizado através de leitores de códigos de barras. No chão de fábrica, estes têm a capacidade de comunicação por Wi-Fi. No entanto, como no decorrer deste projeto, não foi possível a obtenção de um exemplar, foi utilizado um leitor de códigos de barras com a capacidade de comunicação via Rs232, para efetuar a leitura de números de série. O leitor de códigos de barras usado foi um exemplar do modelo EV-J08. Este pode ser visualizado na figura 3.4.



Figura 3.4: Leitor de código de barras utilizado no projeto.

Este produto tem a capacidade para comunicar via USB ou Rs232, de acordo com a configuração escolhida pelo utilizador. Para se configurar o dispositivo, basta passar o leitor pelo código de barras que diz respeito ao tipo de comunicação que se pretende. Neste caso, deve-se passar o leitor pelo código de barras associado ao protocolo Rs232, que se encontra na figura 3.5.

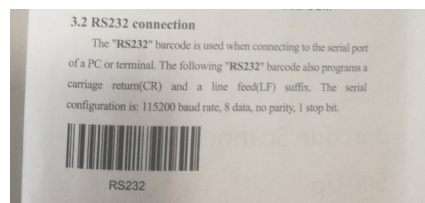


Figura 3.5: Configuração do leitor de código de barras para comunicar via Rs232.

As configurações padrão para comunicação via Rs232 deste equipamento definem um *baud rate* de 115200 bits/s, 8 bits de dados, comunicação sem paridade e 1 bit de stop. As ligações que permitem a comunicação através do protocolo Rs232, foram desenvolvidas pelo colega Pedro Moreira no contexto do seu projeto de dissertação, que antecedeu o projeto que é apresentado neste documento [40]. A comunicação deste equipamento é feita com um ESP8266. Este módulo IOT, tem a capacidade de comunicação via TCP/IP, conseguindo facilmente recolher a informação que lhe é endereçada a partir do leitor de código de barras e redirecioná-la para um servidor responsável pelo seu processamento. As configurações definidas para este equipamento, no que toca à comunicação por Rs232 devem ser iguais àquelas que são utilizadas para o leitor de códigos de barras. A sua programação foi realizada através do software *open-source* Arduino IDE. De seguida são apresentadas, respetivamente, nas figuras 3.6 e 3.7 uma esquematização da ligação física entre o leitor de códigos de barras e o ESP 8266 e uma demonstração da ligação efetiva que foi realizada.

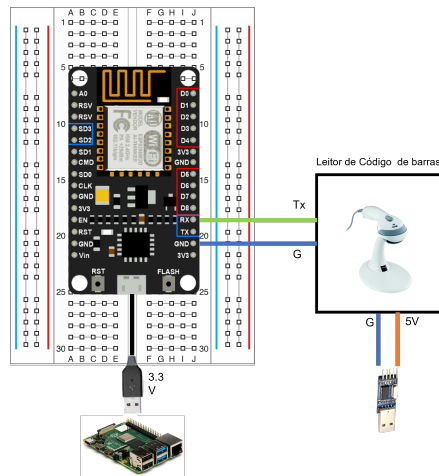


Figura 3.6: Esquema da ligação entre o leitor de código de barras, Raspberry Pi 4 e ESP8266.

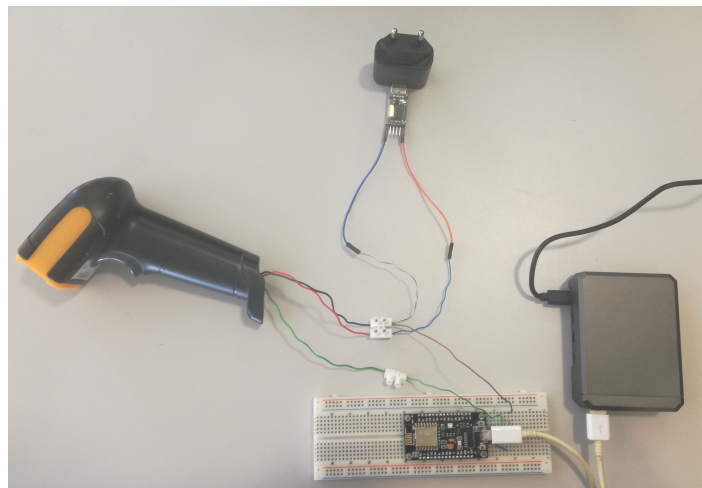


Figura 3.7: Ligações físicas associadas ao ESP8266.

Como se pode perceber pela figura 3.6, só existe uma ligação entre os pinos Tx e Rx do leitor de código de barras e ESP8266, respetivamente. Isto deve-se ao facto de não se pretender uma resposta por parte do módulo IOT para o leitor de código de barras, aquando do envio do número série de uma peça. Deste modo, não é necessário que a configuração seja feita nesse sentido. Adicionalmente, é necessário ligar o pino G do ESP8266 com o “ground” do leitor de código de barras. Estes são, por sua vez, ligados ao “ground” da alimentação do leitor de código de barras. Para além disso, é necessário garantir uma alimentação de 5 Volt ao leitor de código de barras e outra de 3.3 Volt ao ESP8266. No caso do ESP esta é garantida através de uma ligação USB com o Raspberry Pi 4, enquanto que a alimentação do leitor de códigos de barras depende de um conversor Rs232/USB ligado à corrente. Uma vez recebida a informação pelo ESP8266, via Rs232, este comunica os números série da combinação de peças que foi inserida para um *broker* remoto através do protocolo MQTT, publicando nos tópicos “nserie1” e “nserie2”. Esta comunicação é apenas feita caso o seu botão Flash (pode ser

visualizado na figura 3.6) seja premido, e caso o *array* que guarda os números de série lidos pelo leitor tenha as duas posições preenchidas. Na programação do ESP8266 realizada neste projeto, o *broker* MQTT utilizado para permitir a transferência de informações necessárias foi o computador local. Neste encontra-se hospedado o software Mosquitto, que permite a subscrição e publicação de informações a tópicos MQTT feitas numa porta do dispositivo. Como predefinição a porta utilizada é a 1883, sendo que foi esta a porta utilizada para implementar o protocolo MQTT neste projeto.

3.3 Servidor C# e comunicação com os vários módulos

Uma vez efetuada a publicação dos números série nos dois tópicos para o efeito, a sua leitura é efetuada por um servidor desenvolvida em C#. Nesta aplicação estabelece-se uma conexão à base de dados. Esta é efetuada para se conseguir identificar inconformidades na seleção das duas peças. Depois de efetuada a conexão com a base de dados, pode-se aceder a toda a informação que lá se encontra presente. Neste sentido, pode ser utilizada a informação que diz respeito aos *front frames* e *rear frames* e perceber se a escolha de peças pelo operador é passível de ser utilizada para produzir uma peça soldada. Primeiro, deve-se perceber se foram escolhidos dois *rear frames* ou dois *front frames*, o que impossibilita a produção de uma peça por soldadura por transmissão de laser. Caso sejam escolhidas dois *front frames*, a radiação será absorvida pela peça que entrar em contacto em primeiro lugar com o laser, não chegando a ser atingida a interface das peças e impedindo-se a fusão de uma peça à outra. Por outro lado, se forem selecionados dois *rear frames*, a radiação laser não será absorvida em quantidades que permitam a fusão dos materiais. Esta informação pode ser recolhida através do estudo das tabelas “PecasOp” e “PecasTrans”. Caso seja escolhido um *front frame* e um *rear frame*, decifra-se que número série corresponde ao *front frame* e qual número série corresponde ao *rear frame*. Também são extraídas as informações relativas à peça soldada que cada uma das peças pode ser utilizada para produzir e sobre os empenos de cada uma. Estas informações serão necessárias para passos subsequentes. De seguida é necessário perceber se estas peças constroem uma combinação existente. Passada a avaliação dos números série deve ser averiguado se as duas peças, caso sejam um par *front frame/rear frame*, conseguem produzir uma peça por soldadura por transmissão de laser. Para isso, averigua-se qual o índice da peça soldada que cada uma das peças pode ser usada para produzir. Sendo estes números iguais em ambas as peças, é possível produzir-se uma peça com o conjunto selecionado, e por isso é possível extrair os parâmetros padrão para a sua produção, a partir da tabela “ParametrosPadrao”. Caso não haja uma combinação possível, reservam-se os valores dos parâmetros como nulos. Estes valores são publicados nos tópicos “vpadrao”, “fpadrao”, “ipadrao” e “tpadrao” (via MQTT).

O passo seguinte passa pelo fornecimento da informação que é necessária para a criação da previsão da força de separação das peças ao modelo preditivo. As informações que são necessárias para a sua definição são o número identificador da peça final e os empenos de cada peça. O primeiro permite aceder ao modelo preditivo treinado para prever a força de separação entre as duas peças de um tipo de peça soldada final específico. O número identificador da peça final está também associado às tabelas do histórico de cada peça que são usadas para treinar os diferentes algoritmos. Por sua vez os valores de

empeno das peças são relevantes para prever a força de separação das peças. Por limitação dos dados usados, considerou-se apenas a deformação dos *front frames* nas previsões. Estes valores são publicados nos tópicos “deformacaoopaca”, “deformacaotransparente” e “idpeca” (via MQTT). Caso não haja informação nem sobre o empeno das peças definidas e sobre o número identificador da peça final a produzir é enviado o valor “n” para os tópicos mencionados. Depois de recebidos os valores previstos de força de separação, por MQTT, vindos do algoritmo preditivo é enviado para a interface gráfica toda a informação recolhida pelo servidor C#, em formato de texto, por forma a permitir o *debug* da execução do programa e para que erros possam ser detetados de forma eficaz. Uma vez recebido o *feedback* do operador através da interface gráfica, os parâmetros no PLC são atualizados em conformidade (os parâmetros padrão ou os parâmetros definidos pelo utilizador são comunicados ao PLC). Esta comunicação com o PLC é efetuada através do protocolo de comunicação S7 da Siemens. Mais à frente será explanada como a atualização das posições de memória do PLC é concretizada. Todos os processos do servidor anteriormente explicitados podem ser resumidos no fluxograma, representado na figura 3.8. Os blocos pretos na figura referida refletem o comportamento do servidor caso não hajam problemas na escolha das peças ou na ligação ao PLC. Por outro lado, os blocos vermelhos representam o seu comportamento quando estes problemas ocorrem.

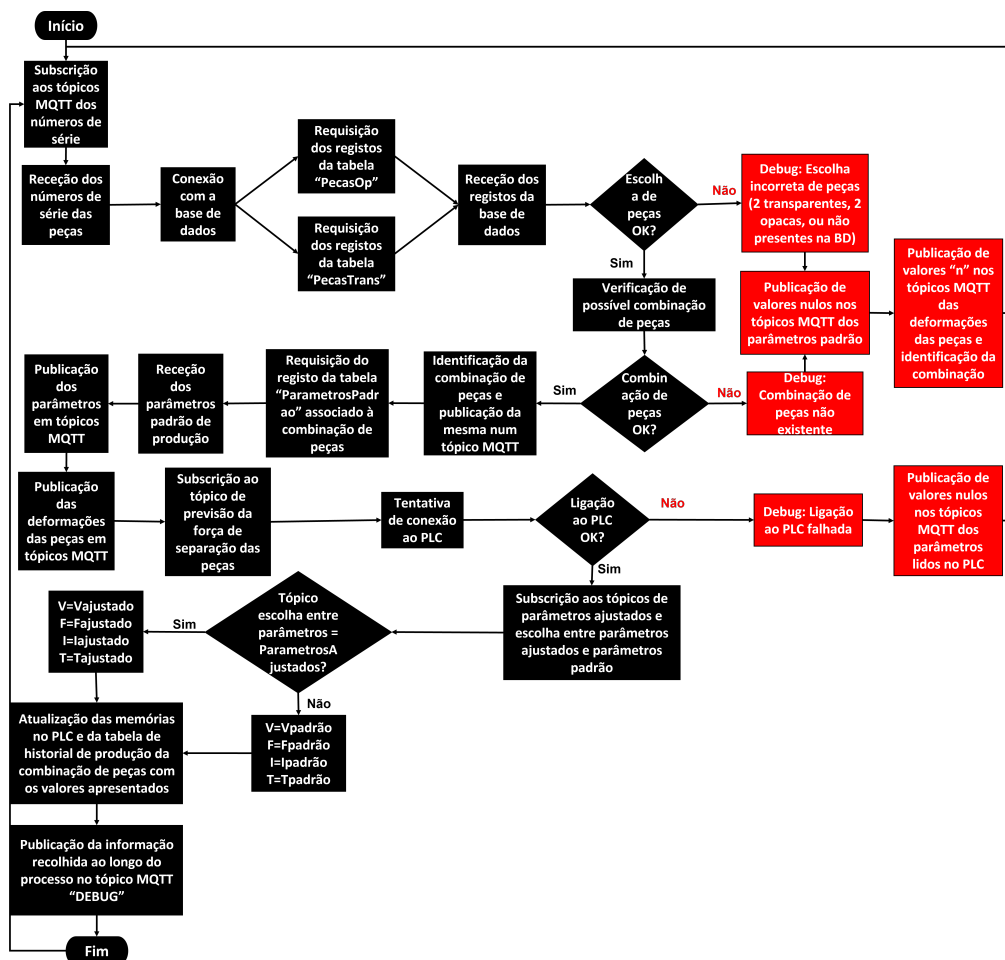


Figura 3.8: Fluxograma do funcionamento do servidor desenvolvido.

3.4 Interface gráfica com o operador da máquina

De seguida, é lógico abordar a criação da interface gráfica, que permite apresentar a informação recolhida ao operador da máquina. Esta foi desenvolvida em Node-Red e apresenta três páginas distintas. Uma delas está reservada a uma caixa de texto, que apresenta informação sobre eventuais erros que possam acontecer durante a escolha de diferentes peças a soldar, recebida através de uma ligação MQTT (não haverem dados para as peças na base de dados, seleção feita de dois *front frames* ou dois *rear frames*, entre outros). Outra página apresenta os valores de cada parâmetro de montagem por soldadura por transmissão de laser, quer sejam estes os valores padrão, os valores das previsões do algoritmo preditivo ou os valores de parâmetros lidos nas posições de memória do PLC, também eles lidos a partir dos valores publicados em tópicos MQTT, tanto no servidor e na aplicação em Python que apresenta o modelo preditivo. Existem também nesta página dois botões que permitem discriminar a preferência do operador da máquina pelos parâmetros padrão ou por parâmetros ajustados que este define manualmente. De notar que se o valor da força de separação previsto diferir em 5% do valor padrão respetivo é lançado um alarme para chamar a atenção do operador para essa ocorrência. Quando existe uma previsão com uma dimensão tal, considera-se que os parâmetros não se adequam de todo à montagem da peça, pelo que essa informação deve ser apresentada. Uma vez pressionado um dos botões, é enviada uma resposta para o servidor. Nesta página, também é possível atualizar a base de dados com informação sobre a força de separação entre duas peças, depois de serem sujeitas a um teste destrutivo. Para isto, basta indicar o número série do *front frame* e a força de separação, sendo a *query* construída automaticamente a partir destas informações. Por fim, é apresentada uma página com diferentes gráficos, que tratam a informação relativa ao histórico de ensaios destrutivos de uma peça de um certo cliente e com uma certa descrição. Tanto o cliente como a descrição do produto são apresentadas como escolhas para o operador, sendo estas extraídas a partir dos valores que existem na base de dados Microsoft SQL Server. São apresentados um “*Line Chart*” que permite visualizar os parâmetros que foram utilizados na montagem de peças que foram sujeitas a testes destrutivos, um “*Pie Chart*” que apresenta a relação entre o número de peças que verificavam o requisito mínimo de qualidade e o número de peças que não verificavam o requisito mínimo de qualidade e um outro “*Line Chart*”, que apresenta os valores de deformação dos *frames*, divididos em séries de acordo com a classificação binária de qualidade (força de separação é inferior ou superior a 95% do valor padrão), para cada ensaio destrutivo. Esta página está também associada a um botão que permite fazer um “*refresh*” dos valores que são recolhidos da base de dados para formar os gráficos. Para se aceder à interface gráfica deve-se introduzir num *browser* o seguinte URL: “xxx.xxx.x.xx:1880/ui”, onde “xxx.xxx.x.xx” representa a morada IP do Raspberry Pi 4. As imagens seguintes nas figuras 3.9, 3.10, 3.11 e 3.12 permitem ilustrar as páginas mencionadas.

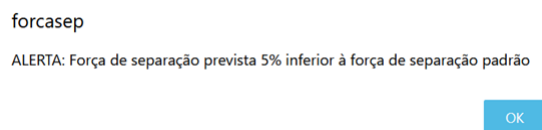


Figura 3.9: Mensagem de alerta da interface.



Figura 3.10: Página de *debug*.



Figura 3.11: Página de parametrização.

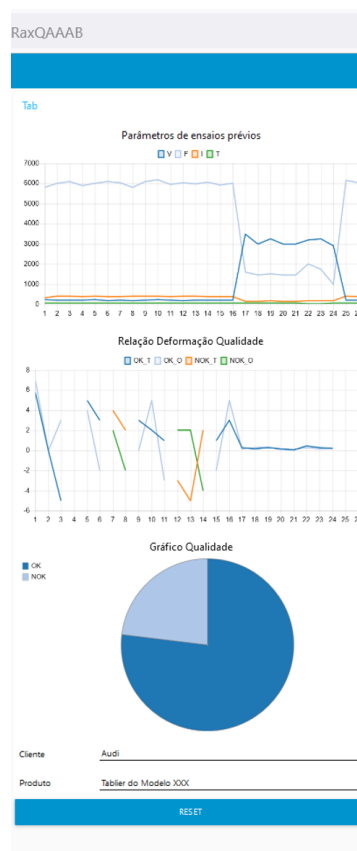


Figura 3.12: Página de gráficos.

Quando recebida a informação relativa aos parâmetros preferidos pelo operador estes são comunicados à base de dados adicionando-se um novo registo à tabela que guarda o historial de produção da peça específica em consideração (operação realizada pelo servidor). No fim, é possível terminar a conexão MQTT, já que não há mais informação a ser partilhada através desse protocolo. De seguida, na figura 3.13, pode ser encontrado um fluxograma que permite esquematizar o funcionamento desta interface gráfica.

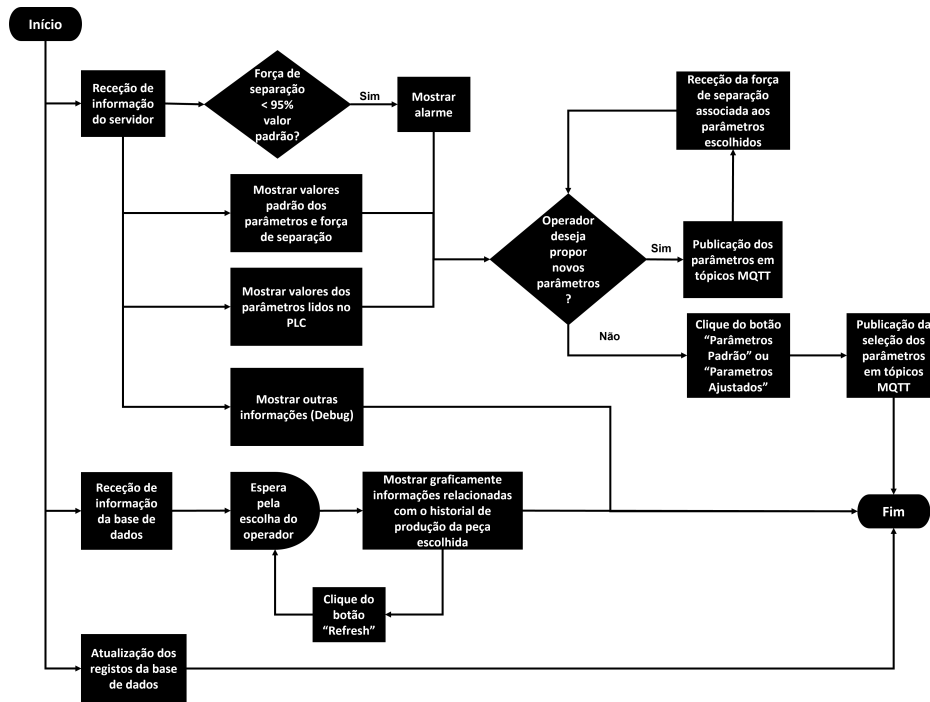


Figura 3.13: Fluxograma do funcionamento da interface gráfica.

3.5 REST API para interação com a base de dados

À semelhança da interface gráfica, que permite a interação de seres humanos com o sistema é necessário construir um módulo adicional que permita a comunicação do sistema com equipamentos no chão de fábrica. Esta comunicação deverá permitir o acesso à informação presente na base de dados e alterá-la se for necessário. Como o servidor anteriormente descrito, este foi desenvolvido com recurso à linguagem de programação C#. Neste contexto, foi usado o ambiente de programação Visual Studio para implementar um programa, baseado na arquitetura “*ASP.NET Core Web Application*”, do tipo “*API*”. Foram integradas no programa as bibliotecas “*Microsoft.EntityFrameworkCore.SqlServer*” e “*Microsoft.EntityFrameworkCore.Design*”. De seguida utilizou-se o método *Scaffolding*, que permite criar código de forma rápida e eficaz que implementa diferentes tipos de comunicação com uma certa base de dados. Para utilizar este método recorreu-se à *Package Manager Console* do Visual Basic, onde se introduziu o texto que será apresentado de seguida. A fração de texto entre aspas representa a *string* que permite realizar a conexão à base de dados pretendida. De forma a conseguir-se conectar com outra base de dados, esta teria que ser alterada:

Scaffold-DbContext "Server=192.168.1.88, 1433; Database= LTWIO; User Id= IBER-OLEFF; Password = IBER-OLEFF;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Model -Context "WebApiContext" -DataAnnotations

Com este texto, são criados diferentes ficheiros no projeto que permitem criar o modelo da aplicação. Este é composto por um conjunto de diferentes classes que contêm a informação que a aplicação é capaz de processar. Um destes denominado de "*WebApiContext*" apresenta a classe "*DbContext*" que coordena as trocas de informação que são efetuadas nesta aplicação. Aparecem também outros ficheiros, contendo cada um deles a sua classe, onde é definido o esqueleto das tabelas presentes na base de dados (os seus atributos, relacionamentos, chaves primárias, entre outros). Para se conseguir interagir com a base de dados, no entanto, ainda é necessário definir controladores para cada uma das classes que contêm o esqueleto de uma tabela na base de dados. Os controladores utilizados nesta aplicação são, tal como as classes no modelo, pré-formatados para aplicações "*API*" que usam a *Entity Framework*. Basta, para produzir o controlador de uma tabela na base de dados, clicar com o botão direito na pasta "*Controllers*" e adicionar um "*New Scaffolding Item*". Numa nova página, deve referir-se qual a classe do modelo e a "*DbContext*" que se pretende utilizar e o programa gera um controlador capaz de processar mensagens HTTP GET (para ler os registos da tabela), POST (para inserir novos registos na tabela), PUT (para atualizar registos na tabela) e DELETE (para apagar registos da tabela). Este processo encontra-se sequencializado nas figuras 3.14, 3.15 e 3.16. Nas figuras 3.17 e 3.18, podemos visualizar a interface criada através desta aplicação com o auxílio do software Swagger, onde este é expresso em formato JSON. Esta aplicação pode ser acedida ao introduzir no *browser* o seguinte URL: "*xxx.xxx.x.xx:5001/swagger*", onde "*xxx.xxx.x.xx*" representa a morada IP do Raspberry Pi 4.

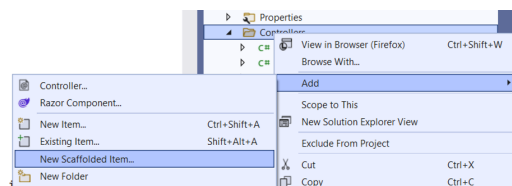


Figura 3.14: Criação de um novo "*Scaffolding item*".

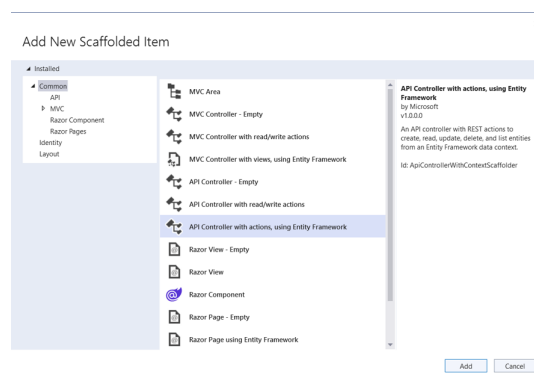


Figura 3.15: Seleção do item.

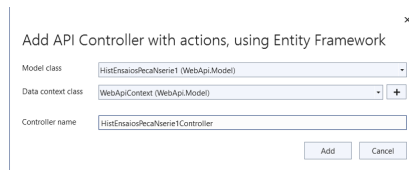


Figura 3.16: Referência ao “DbContext”.

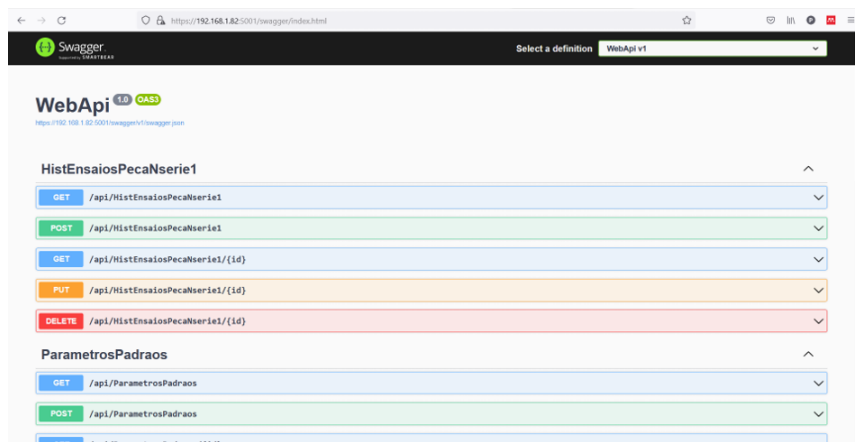


Figura 3.17: Interface gráfico do software Swagger e tipos de mensagens que são admitidas.

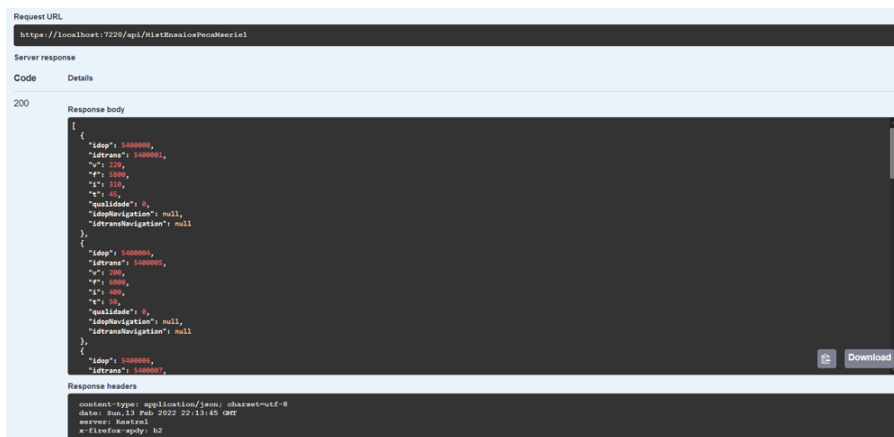


Figura 3.18: Exemplo do resultado de um pedido HTTP GET.

3.6 Autômato e atuação dos parâmetros do processo

Para se saber se é necessário alterar os parâmetros do processo de soldadura deve primeiro conhecer-se os parâmetros que estão atuados no equipamento de soldadura. Para se resolver esta questão é necessário estabelecer uma comunicação com o autômato programável (PLC) e extrair os parâmetros presentes nas suas posições de memória. Para simular a comunicação dos parâmetros de produção ao PLC da Iber-Oleff (S300 da marca Siemens), foi utilizado um PLC VIPA, ligado por cabo físico a um Raspberry Pi 4. Para programar a comunicação com este dispositivo, foi utilizado o software *open-source* TIA PORTAL fornecido pela Siemens. Antes

de se programar o PLC, é necessário definir os seus parâmetros de comunicação de tal forma que esta seja possível. Sendo a comunicação garantida através do protocolo ethernet (sobre o protocolo TCP/IP) é necessário que o PLC apresente uma morada IP no mesmo campo, mas diferente da morada IP do Raspberry Pi 4. Desta forma, e considerando que a morada IP da porta ethernet do Raspberry Pi 4 é “192.168.0.5”, atribui-se ao PLC a morada “192.168.0.10”, como se pode verificar pela figura 3.19.

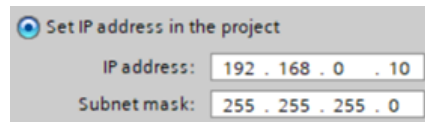


Figura 3.19: Endereço IP do PLC.

Foi definida também a interface entre o CPU e PLC. Esta foi definida através do protocolo PROFINET associada a uma subrede PN/IE e pode ser observada na figura 3.20.

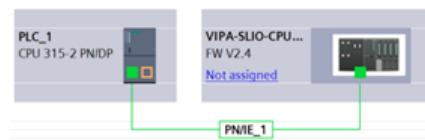


Figura 3.20: Interface entre CPU e PLC.

De salientar também as posições do CPU, apresentadas na figura 3.21. Estas serão necessárias mais adiante quando se definir a comunicação com o Raspberry Pi 4.

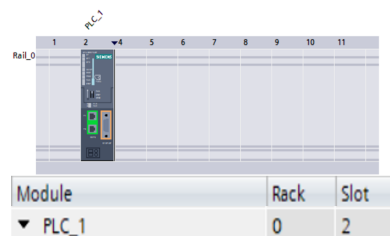


Figura 3.21: Indicação das posições do CPU.

A configuração utilizada dependeu da utilização de dois blocos de dados (DB). Um destes contém a informação que diz respeito aos parâmetros a alterar no equipamento de soldadura e o outro contém uma variável booleana (binária). Esta permite perceber quando devem ser alteradas as posições de memória do autómato. De seguida são apresentados esses dois blocos nas figuras 3.22 e 3.23.

Data_block_1							
	Name	Data type	Offset	Start value	Retain	Visible in ...	Setpoint
1	Static						
2	V	Int	0.0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
3	F	Int	2.0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	I	Int	4.0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
5	T	Int	6.0	0	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figura 3.22: Data block 1.

Data_block_2							
	Name	Data type	Offset	Start value	Retain	Visible in ...	Setpoint
1	Static				<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Estado	Bool	0.0	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figura 3.23: Data block 2.

Podemos verificar que existe uma diferença entre os dois DB. No “Data_block_1” todas as variáveis apresentam o valor “Retain”, ao contrário do que se verifica no “Data_block_2”. Este valor permite manter o valor de uma variável, mesmo se o equipamento for desligado. Pelo contrário, caso uma variável não tenha este valor, depois de desligado o equipamento, quando se reiniciar a produção novamente, os valores voltam para o seu valor padrão. Considerou-se que a retenção dos valores das variáveis que dizem respeito aos parâmetros de produção fazia sentido, já que permitem continuar a produção de uma certa peça em diferentes dias de trabalho, sem ser necessário alterá-los no início do dia, caso assim seja desejado pelo operador. Por outro lado, não se verifica essa necessidade com a variável do “Data_block_2”. Por agora, é apenas relevante o “Data_block_1”. Será a partir deste que são extraídos os parâmetros atuados no PLC, pelo servidor. Para este efeito foi utilizada a biblioteca Sharp7, que define as funções para C#, que constroem o protocolo S7 da Siemens utilizado para efetuar comunicações com PLCs. Depois de lidos os valores, estes são publicados nos tópicos MQTT “vlida”, “flida”, “ilida” e “tlida”. Caso não se consiga efetuar a ligação ao PLC ou não se consiga extrair uma leitura do “Data_block_1”, publicam-se nos tópicos anteriores valores nulos.

Como último passo, é necessário definir os parâmetros no PLC de acordo com a preferência do operador. Tal como a leitura dos parâmetros do PLC, é utilizado o protocolo S7, para efetuar a comunicação com o PLC. Estes parâmetros serão definidos apenas se os parâmetros escolhidos pelo operador forem diferentes daqueles presentes no “Data_block_1”. Caso se verifique essa condição, os valores do “Data_block_1” são atualizados e altera-se a variável “Estado” do “Data_block_2” para o valor “True”. Nestas circunstâncias o valor das posições de memória associadas a cada parâmetro são atualizadas com os valores pretendidos. Depois o valor dessa variável é definido como “False” novamente, interrompendo a transmissão de dados. A programação do autómato foi realizada em Ladder, e pode ser visualizada nas figuras 3.24 e 3.25, que descrevem o modo como a velocidade do laser é comunicada ao autómato.

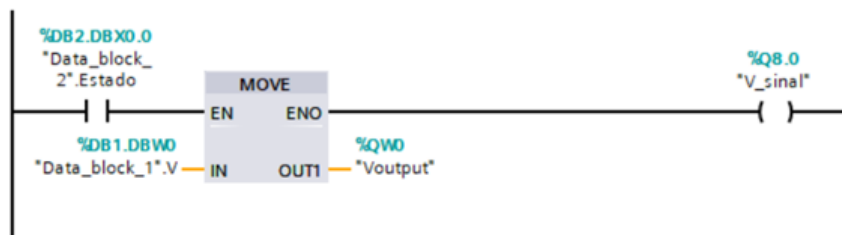


Figura 3.24: Atualização da posição de memória V.

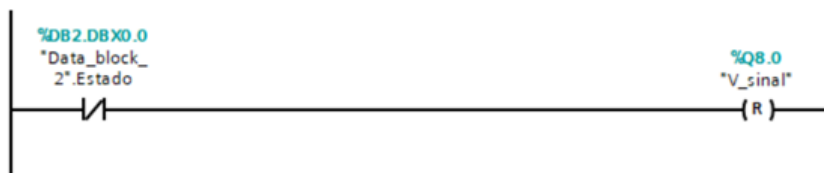


Figura 3.25: Reset do indicador de alteração de parâmetros.

Como se pode verificar, quando a variável “Estado” é ativada (“Estado”=true), é acionada

uma “*move operation*”, que coloca o valor presente no “*Data_block_1*”, na posição de memória correspondente. Ao mesmo tempo é acionada um bit de saída para demonstrar que os parâmetros foram transferidos com sucesso. Quando, a variável “Estado” se encontra desativada (“Estado”=*false*), então esse bit é desativado.

3.7 Algoritmo preditivo da força de separação dos *front* e *rear frames*

O algoritmo preditivo é a parte da solução proposta que se dedica à previsão da força de separação entre duas peças para que o ajuste de parâmetros do processo de soldadura por transmissão de laser, tenha o devido suporte e possam ser evitados defeitos na produção. Na definição deste módulo foi utilizada a linguagem de programação Python, pela facilidade comparativa na escrita de código em relação a C# e também pela existência de várias *frameworks* como o Pytorch e Sklearn associadas à criação de algoritmos preditivos de ML. Como conjunto de dados para criar este algoritmo foram utilizados dados de um projeto anterior desenvolvido pela Iber-Oleff, o PolyWeld. Neste foram realizados vários ensaios onde se conseguiu deduzir a relação entre diferentes variáveis do processo de soldadura com a força de arranque ou separação das peças soldadas. A partir deste projeto conclui-se que as variáveis mais relevantes para a determinação da força de arranque são a intensidade do feixe laser, a força de compactação, a velocidade do laser e uma deformação ou desvio no “*rib*” do *front frame*. Serão, portanto, apenas consideradas estas variáveis neste projeto, uma vez que são as que os dados existentes controlam. O conjunto de dados recolhido, encontra-se representado na tabela 3.1:

Tabela 3.1: Conjunto de dados recolhidos associados ao projeto PolyWeld.

Registo	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
1	0	130	1350	2850	837
2	0	150	1450	3000	803
3	0	170	1550	3150	795
4	0.2	150	1450	3150	714
5	0.2	170	1550	2850	745
6	0.2	130	1350	3000	568
7	0.4	130	1550	3000	525
8	0.4	150	1350	3150	486
9	0.4	170	1450	2850	636
10	0	170	1450	3000	817
11	0	130	1550	3150	738
12	0	150	1350	2850	907
13	0.2	170	1350	3150	640
14	0.2	130	1450	2850	734
15	0.2	150	1550	3000	873
16	0.4	150	1550	2850	631
17	0.4	170	1350	3000	622
18	0.4	130	1450	3150	578
19	0	140	1450	2910	776
20	0	150	1500	2850	720
21	0	160	1550	2790	734
22	0.1	140	1500	2790	730
23	0.1	150	1550	2910	779

Continua na página seguinte

Tabela 3.1 – continuação da página anterior

Registo	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
24	0.1	160	1450	2850	660
25	0.2	140	1550	2850	621
26	0.2	150	1450	2790	591
27	0.2	160	1550	2910	591
28	0	140	1550	2910	650

Este conjunto de dados foi utilizado para treinar e testar 2 diferentes tipos de algoritmos diferentes: XGBoost e ANN. Estes foram usados para criar um modelo preditivo capaz de prever a força de separação entre as peças a partir dos parâmetros a serem utilizados. No entanto, o conjunto de dados a que se teve acesso é manifestamente reduzido o que tem diretas implicações na performance dos algoritmos desenvolvidos. De facto, quanto menor a quantidade de informação que é alimentada a um algoritmo preditivo, menos instrumentos este terá para se conseguir conformar às leis matemáticas que definem um processo não linear e complexo como a soldadura por transmissão de laser. Os algoritmos para além de terem um comportamento pior também apresentarão um comportamento mais instável, isto é, com maior variância. Com o reduzir do número de registos, eventuais experiências que devido ao erro humano do operador ou outros fatores não representem o comportamento esperado do processo têm um impacto maior no comportamento do algoritmo. Assim, a divisão dos dados entre dados de treino e dados de teste também terá uma maior relevância sobre o conjunto de previsões que o algoritmo consegue realizar. Para além disso, não são acauteladas neste conjunto de dados variáveis como o tempo de arrefecimento da peça e deformação da peça transparente que influenciam o processo de soldadura por transmissão de laser. De forma a demonstrar o efeito que o conjunto de dados tem sobre a performance de um algoritmo, foram realizados testes que permitiram comparar a exatidão média e o valor da função de erro médio usando num caso o conjunto de dados apresentado na tabela 3.1 e noutro um conjunto de dados mistos que usa tanto os dados reais como um conjunto de dados sintéticos.

Uma vez realizada a seleção dos dados reais a serem utilizados, é necessário definir um método para gerar dados sintéticos. Uma forma possível de conseguir estes dados seria através de métodos de simulação numérica. No entanto, optou-se por gerar dados de forma aleatória, mas que apresentassem as mesmas características estatísticas (média e covariância) que o conjunto de dados reais. Para gerar estes dados admitiu-se que cada distribuição corresponderia a uma distribuição normal, uma vez que de acordo com o teste de Henze-Zirkler é possível conformar os dados existentes a uma distribuição normal multivariável [41]. Este teste faz uma comparação entre o valor de diferentes parâmetros estatísticos da amostra e os de uma distribuição normal. Este método produz um valor de “p-value”, que caso seja superior ao valor do nível de significância utilizado (para o qual se considerou um valor de 0.05), pode-se concluir que o conjunto de dados é dado por uma distribuição normal multivariável. Depois de realizado este teste, definiu-se uma amostragem aleatória de 200 registos da distribuição normal com a média e covariância que foram calculadas a partir do conjunto de dados reais e foi aplicado o teste de Kolmogorov-Smirnov a cada coluna das várias amostras selecionadas para perceber se estas eram estatisticamente semelhantes ao conjunto de dados reais. Para isso, compararam-se os valores retirados de cada teste com o valor crítico associado ao número de amostras presentes no conjunto de dados reais. Neste caso como a amostra de dados reais em questão é composta por 28 registos, o valor do teste deve ser menor que 0.25, se for considerada uma confiança de 0.95 [42]. A amostragem foi repetida até ser encontrado um conjunto de dados, onde se verificasse que o valor do teste de Kolmogorov-Smirnov fosse menor que o valor crítico em todas as colunas de dados. Para além disso, para se perceber se as relações entre colunas eram mantidas, foi também realizado um teste de correlação Pearson. Este permite avaliar o grau de correlação entre duas variáveis num conjunto de dados, sendo que ao comparar os valores associados a este teste, para o conjunto real de dados e para

o conjunto sintético de dados conseguimos perceber se de facto a criação de dados sintéticos é fiel ao conjunto de dados inicial. Os resultados deste teste podem ser observados de forma gráfica na figura 3.26. De salientar que nesta figura os valores “0”, “1”, “2”, “3” e “4” no gráfico da esquerda representam a deformação do “rib” do *front frame*, a intensidade do feixe laser, a força de compactação, a velocidade do feixe laser e a força de separação das peças, respetivamente. A mesma legenda pode-se aplicar a “DefOp”, “I”, “F”, “V” e “Fa” do gráfico da direita.

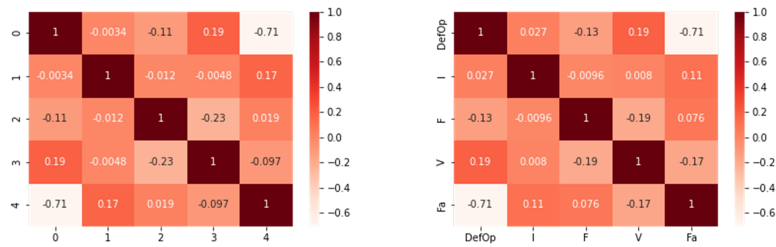


Figura 3.26: Teste de correlação de Pearson aplicado ao conjunto de dados mistos (à esquerda) e ao conjunto de dados reais (à direita).

Pode-se concluir que com a exceção da relação entre a força de separação e a velocidade do laser, onde existe a diferença de uma ordem de grandeza no grau de correlação entre as duas variáveis para os dois conjuntos de dados, os restantes valores são bastantes semelhantes, pelo que é admissível usar o conjunto de dados mistos como uma extensão do conjunto de dados reais. De facto, não está a ser criado conhecimento sobre o processo de soldadura por transmissão de laser, pelo que se um algoritmo não se conforma de forma correta com o comportamento real do processo, não conseguirá fazê-lo com acesso a dados sintéticos. No entanto, consegue-se criar uma melhor definição sobre as diferentes combinações que o algoritmo consegue prever, permitindo melhorar-se o conhecimento que existe sobre o comportamento do mesmo. Os dados mistos, já standardizados, resultantes deste processo podem ser visualizados na tabela C.1, que se encontra em anexo a este documento.

De seguida, com os conjuntos de dados criados, foram efetuadas 10 divisões aleatórias desses mesmos em dados de treino e dados de teste, para avaliar como o valor da função de perda e exatidão variam com diferentes conjuntos de dados de treino e teste. Estes foram normalizados antes de serem divididos de forma a que os seus valores se encontrassem entre 0 e 1. Este passo visou aumentar a estabilidade numérica do algoritmo, já que são realizados ajustes aos algoritmos mais reduzidos o que por sua vez facilita a convergência, bem como aumenta a velocidade do processo de treino. Foram tomados passos análogos para o conjunto real de dados. Uma vez concretizados todos estes passos, foi possível treinar os algoritmos. Este treino, consistiu no uso iterativo do método de Taguchi até se achar um conjunto de hiperparâmetros ótimos para o processo. Como tanto para o caso das ANN e do XGBoost foram controlados 8 hiperparâmetros, cada um com três níveis associados e utilizou-se a tabela L27 associada ao método de Taguchi para conduzir os ensaios iterativos. Este design experimental, com as suas 27 experiências e níveis de cada hiperparâmetro a serem utilizados podem ser encontrados na tabela 3.2.

Tabela 3.2: Tabela Taguchi L27 que permitiu conduzir a otimização de hiperparâmetros.

Taguchi L27	Parâmetros							
	X1	X2	X3	X4	X5	X6	X7	X8
Experiências	1	1	1	1	1	1	1	1
	2	1	1	1	1	2	2	2

Continua na página seguinte

Tabela 3.2 – continuação da página anterior

Taguchi L27	Parâmetros								
	X1	X2	X3	X4	X5	X6	X7	X8	
	3	1	1	1	1	3	3	3	3
	4	1	2	2	2	1	1	1	2
	5	1	2	2	2	2	2	2	3
	6	1	2	2	2	3	3	3	1
	7	1	3	3	3	1	1	1	3
	8	1	3	3	3	2	2	2	1
	9	1	3	3	3	3	3	3	2
	10	2	1	2	3	1	2	3	1
	11	2	1	2	3	2	3	1	2
	12	2	1	2	3	3	1	2	3
	13	2	2	3	1	1	2	3	2
	14	2	2	3	1	2	3	1	3
Experiências	15	2	2	3	1	3	1	2	1
	16	2	3	1	2	1	2	3	3
	17	2	3	1	2	2	3	1	1
	18	2	3	1	2	3	1	2	2
	19	3	1	3	2	1	3	2	1
	20	3	1	3	2	2	1	3	2
	21	3	1	3	2	3	2	1	3
	22	3	2	1	3	1	3	2	2
	23	3	2	1	3	2	1	3	3
	24	3	2	1	3	3	2	1	1
	25	3	3	2	1	1	3	2	3
	26	3	3	2	1	2	1	3	1
	27	3	3	2	1	3	2	1	2

Em cada um destes ensaios iterativos, foram realizadas 27 experiências com diferentes combinações de níveis de hiperparâmetros, repetidas pelos dez conjuntos de dados de treino e teste. No final de cada ensaio, foram calculados os valores médios da métrica a considerar. No treino dos algoritmos consideraram-se duas métricas diferentes: a média dos valores do erro quadrático médio (RMSE) nos conjuntos de dados de treino e teste e o valor de RMSE no conjunto de dados de teste. Para cada uma destas métricas, foram calculados os valores associados ao rácio S/N STB de cada experiência. A partir destes 27 valores foi possível perceber qual o valor médio do rácio S/N associado a cada nível de cada parâmetro. Quanto maior este valor para um certo nível, em teoria, mais próximo este se encontrará do valor ótimo do hiperparâmetro. Deste modo, para além dos valores de S/N calculados para cada experiência é necessário executar uma experiência adicional com o conjunto dos níveis dos hiperparâmetros que maximizam o valor do rácio S/N. No final de executada esta experiência é possível perceber qual o conjunto ótimo de níveis dos hiperparâmetros, que maximizam o valor de S/N. De forma a poder conseguir-se uma convergência aos valores ótimos dos hiperparâmetros, o ensaio seguinte deve ser centrado à volta dos níveis ótimos dos hiperparâmetros obtidos no ensaio anterior. Considerou-se que os hiperparâmetros ótimos eram atingidos quando o valor máximo de S/N produzido num ensaio era menor que o valor máximo de S/N obtido no ensaio anterior ou quando a variação de S/N de um ensaio para outro era menor que 0.1%. Para além de serem conduzidas otimizações com métricas diferentes, também foram conduzidos ensaios com conjuntos de dados diferentes, de forma a permitir-se uma comparação da performance dos algoritmos com o conjunto de dados reduzido, mas real e com o conjunto de dados misto, mas mais composto em termos de registos. De seguida serão apresentados os algoritmos testados e os hiperparâmetros a serem controlados, bem como pressupostos tomados e como foram definidos os níveis dos hiperparâmetros de cada ensaio. Para mais informações sobre como este processo foi desenvolvido, é possível encontrar

no anexo B, tabelas que apresentam de uma forma mais pormenorizada os resultados obtidos neste processo de otimização de hiperparâmetros. Poderão ser encontradas tabelas que revelam quais os níveis considerados em cada ensaio, os valores de S/N resultantes de cada experiência, os valores médios de S/N para cada nível de cada hiperparâmetro, os valores de S/N associados à experiência adicional que considerou como níveis aqueles que maximizavam o valor de S/N em cada ensaio, as exatidões e RMSE associados em cada repetição da experiência que produziu os valores ótimos de S/N e por fim os gráficos que representam a evolução das exatidões e RMSE durante o treino das ANN e os gráficos que permitem comparar as previsões dos algoritmos XGBoost com os dados de teste e treino usados.

3.7.1 Rede neuronal artificial

Para as ANN, foram usadas de forma transversal para todos os conjuntos de dados redes totalmente conectadas (*fully connected*), com 4 nós de entrada e 1 nó de saída. Todas as ANN desenvolvidas encontram-se associadas ao algoritmo *batch normalization*, em todas as suas camadas escondidas. Este algoritmo permite atenuar o problema de dissipação de gradientes que dificulta a atualização de pesos e *bias* em ANN mais profundas, através da normalização das contribuições acrescentadas a estes parâmetros nas camadas da rede neuronal. Este método permite também acelerar o processo de aprendizagem do algoritmo [43]. Como método de otimização foi utilizado o *stochastic gradient descent*. Todos os restantes hiperparâmetros que serão mencionados encontraram-se sujeitos a alterações, sendo considerados variáveis a controlar. São estas o número de camadas escondidas (*hidden*), o número de nós por camada escondida (*n_hidden*), a função de ativação das camadas escondidas (*fa_hidden*), a função de erro a ser utilizada (*f_loss*), o momento (*m*), o ritmo de aprendizagem (*lr*), a probabilidade (*p*) associada à regularização *dropout*, que visa atenuar o problema de *overfitting* (a probabilidade enunciada refere-se à probabilidade de, em cada camada, um neurónio não ser usado no processo de treino) e o número de passagens do conjunto de dados de treino pela rede neuronal (*epoch*). Como forma de simplificação do estudo a ser realizado, considerou-se que todas as camadas escondidas seriam compostas pelo mesmo número de nós e pela mesma função de ativação. Para iniciar o treino das redes neuronais com os vários conjuntos de dados foram utilizadas redes com a mesma estrutura e hiperparâmetros iniciais, que se podem encontrar na tabela 3.3:

Tabela 3.3: Parâmetros utilizados na primeira iteração da otimização de hiperparâmetros da ANN.

Parâmetros	hidden	n_hidden	fa_hidden	f_loss	m	lr	p	epoch
1 ^a iteração	3	100	ReLU	MSE	0.9	0.001	0	100

Para avaliar a performance do algoritmo foi utilizado o valor do erro quadrático médio, tanto para os conjuntos de dados de treino e teste. A partir do cálculo deste erro, permitiu-se também calcular o rácio S/N STB. Para se efetuar este passo foram realizados dois conjuntos de ensaios distintos. Um deles considerava para o cálculo do rácio S/N a média do erro quadrático médio entre os conjuntos de treino e teste, enquanto que no outro considerava-se apenas o erro quadrático médio nos conjuntos de teste. Foi também calculado o valor de exatidão do algoritmo, ainda que este não tenha sido integrado no método de Taguchi. A exatidão do algoritmo pode ser dada pela equação 3.1:

$$E = 1 - \frac{|P - R|}{R} \quad (3.1)$$

Nesta, E representa a exatidão do algoritmo, P representa o valor da previsão do algoritmo e R o valor verdadeiro da grandeza a ser prevista pelo algoritmo.

3.7.2 XGBoost

Os algoritmos XGBoost foram desenvolvidos a partir de uma *framework* Python denominada de `xgboost`. Os hiperparâmetros associado à classe de regressores XGBoost disponibilizados por esta *framework* que serão considerados são: o número de DT associadas ao algoritmo (`n_dt`), a profundidade máxima de cada DT (`prof_max`), o número máximo de nós folha (`folha_max`), o ritmo de aprendizagem do algoritmo (`lr`), o valor mínimo de variação de erro para criar uma separação num nó (`div_min`), o valor mínimo da soma dos pesos das observações para se criar um novo nó (`peso_min`), a função de erro a utilizar no treino do algoritmo (`loss`) e uma constante (`lambda`) associada à regularização L2 do algoritmo, também conhecida como *Lasso regression*. Para estas variáveis, numa primeira iteração, foram utilizados valores baseados nas predefinições da *framework* `xgboost`. Os valores predefinidos pela classe são os que são apresentados na tabela 3.4:

Tabela 3.4: Parâmetros utilizados na primeira iteração da otimização de hiperparâmetros do modelo XGBoost.

Parâmetros	<code>n_dt</code>	<code>prof_max</code>	<code>folha_max</code>	<code>lr</code>	<code>div_min</code>	<code>loss</code>	<code>peso_min</code>	<code>lambda</code>
1ª iteração	100	6	Nenhuma	0.3	0	MSE	1	1

Tal como anteriormente, foram utilizadas como métricas para avaliar a performance do algoritmo a exatidão e erro quadrático médio das previsões, em 10 diferentes conjuntos de dados de treino e teste. Para além disso, foram usadas duas métricas para calcular o rácio S/N, nomeadamente a média do erro quadrático médio nos conjuntos de dados de treino e teste e o valor do erro quadrático médio nos conjuntos de dados de teste.

3.7.3 Implementação do algoritmo

De forma a conseguir-se uma implementação prática do algoritmo, foi desenvolvida uma aplicação em Python, que permite receber informações através do protocolo MQTT e a partir dessas informações produzir uma previsão da força de separação entre peças. Mais concretamente, essas informações dizem respeito à deformação do *front frame* e o número identificador da peça final que deverá ser obtida. De acordo com estas informações, publicadas pelo servidor, é escolhido o algoritmo que deve ser usado para se realizar a previsão e é passado pelo algoritmo um registo composto pela deformação do *front frame* e pelos parâmetros padrão para essa peça em concreto. Como *output*, obtém-se a previsão da força de separação das duas peças que compõem o produto soldado com o conjunto de parâmetros padrão. De seguida, o programa fica à escuta de mensagens MQTT, caso o operador deseje sugerir novos parâmetros para produzir a peça sendo produzidas novas previsões de força de separação das peças, mas usando neste caso os parâmetros ajustados como *input*, em vez dos parâmetros padrão. Este processo é corrido ciclicamente, até que o operador opte ou pelos parâmetros padrão ou pelos parâmetros ajustados, para produzir a peça. De seguida, o programa é reiniciado, esperando por mais informação vinda do servidor. Seguidamente é apresentado, na figura 3.27, um fluxograma que permite esquematizar de forma gráfica o funcionamento da aplicação:

3.8 Containerização dos módulos através do software Docker

O Docker permite criar de pacotes de outros softwares, virtualizando-os. Esta virtualização é diferente daquela que se encontra em máquinas virtuais, onde todo o sistema operativo é replicado. Isto não é eficiente já que exige a simulação de um sistema operativo para cada aplicação. Assim, caso se pretendam virtualizar várias aplicações, como neste caso, irá ser ocupado muito espaço da memória do dispositivo de forma desnecessária. Em contraste, a

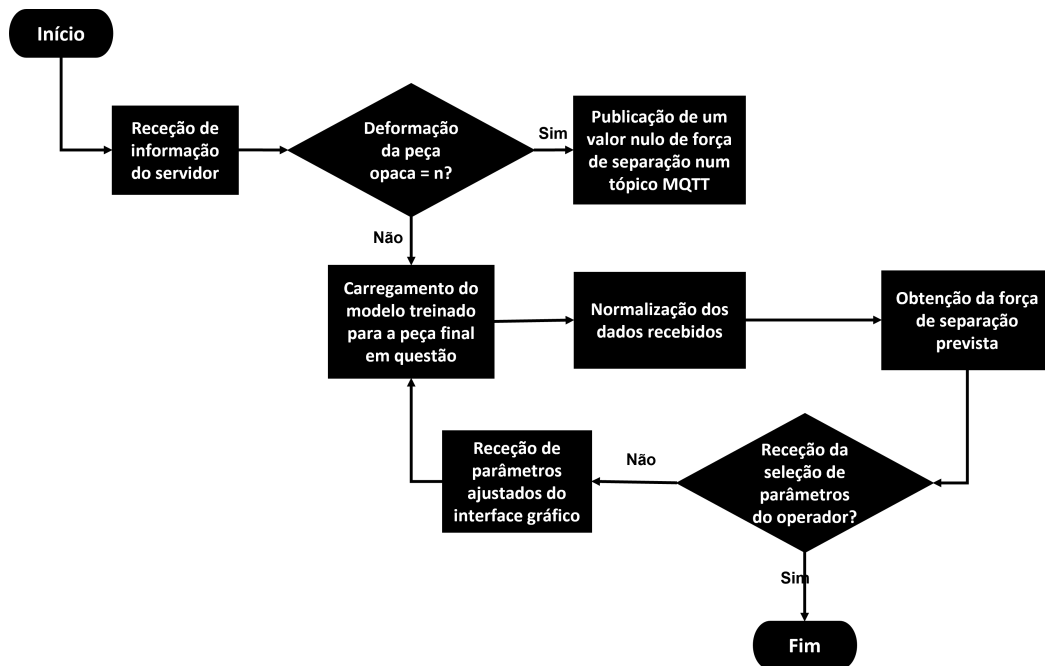


Figura 3.27: Fluxograma do funcionamento da aplicação que permite implementar o algoritmo preditivo.

virtualização ou containerização praticada pelo Docker é feita ao nível do sistema operativo. Isto significa que todas as aplicações empacotadas pelo Docker partilham o mesmo kernel com o sistema operativo. Desta forma, uma vez virtualizada uma aplicação ou um conjunto de aplicações através do Docker, estas podem ser usadas em diferentes sistemas operativos e funcionar de igual forma. Cada aplicação, depois de inserida num pacote conjuntamente com as suas dependências, cria uma imagem. Uma instância desta imagem pode ser posteriormente corrida num ambiente normalizado ou “*container*”, através do Docker Engine. A diferença entre um “*container*” e uma máquina virtual, pode ser percecionada através da figura 3.28:

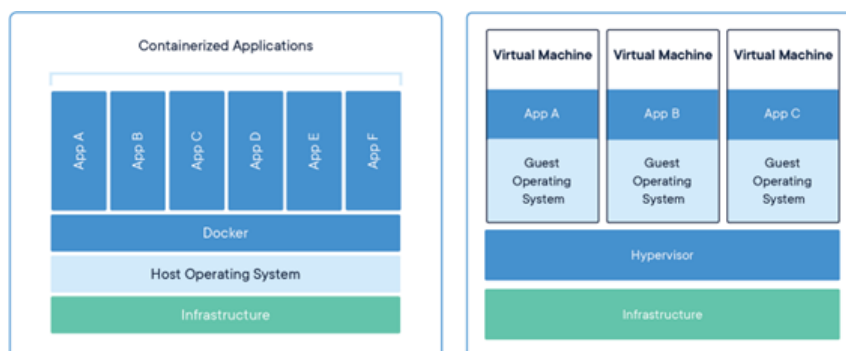


Figura 3.28: Comparação da estruturação de máquinas virtuais com um conjunto de “*containers*” Docker [16].

Para além de garantir a possibilidade de correr um conjunto de aplicações em sistemas operativos diferentes, o Docker também permite que estas aplicações corram em qualquer dispositivo independentemente das versões de software que se encontram instaladas. Isto cria uma clara vantagem no que diz respeito à escalabilidade de um programa ou conjunto de programas. Pri-

meiramente, para se conseguir correr um conjunto de aplicações em “*containers*”, é preciso criar imagens virtuais dessas mesmas aplicações. Este processo é executado através de diferentes “*Dockerfiles*”. Um “*Dockerfile*” é um ficheiro de texto localizado no diretório de cada projeto que se pretende virtualizar e que contém os comandos necessários para se proceder à criação da imagem de um programa com todas as suas dependências. Este ficheiro é composto, portanto, por várias instruções e pelos argumentos respetivos. Para facilitar a perceção de ambos, convencionou-se que as instruções sejam escritas em letras maiúsculas e os seus argumentos em letras minúsculas. No que diz respeito às instruções que podem ser executadas, estas podem ser de variadas naturezas. Num ficheiro “*Dockerfile*”, podem ser utilizadas instruções como “*FROM*”, “*COPY*”, “*EXPOSE*”, “*RUN*”, entre outras. De salientar que todos estes ficheiros devem ser inicializados com uma instrução “*FROM*”. Esta instrução permite definir a imagem base, a partir da qual se irá criar a imagem da aplicação pretendida. Em geral, existem no repositório Docker Hub, diversas imagens base oficiais de diferentes softwares, linguagens de programação e sistemas operativos que podem ser utilizados neste tipo de instrução. De seguida, é boa prática definir o diretório sobre o qual devem ser executadas quaisquer instruções subsequentes. Esta tarefa pode ser conseguida através da instrução “*WORKDIR*”. Caso o diretório indicado não exista, este será criado com a execução da instrução. Uma outra instrução relevante é “*COPY*”. Esta permite fazer a cópia de ficheiros ou pastas presentes no computador local para o diretório do “*container*” definido na instrução “*WORKDIR*”. Uma vez copiados ficheiros para o diretório no “*container*” é possível executar tarefas sobre esses mesmos, que conduzem à criação de uma imagem virtual dos programas. Para se realizar este passo, podem ser usadas diferentes instruções no “*Dockerfile*”. Uma destas é a instrução “*RUN*”. Esta permite executar qualquer comando, criando uma nova imagem atualizada em relação à imagem existente antes da sua aplicação. Esta pode ser executado de duas formas: “*shell*” ou “*exec*”. Uma outra instrução relevante num “*Dockerfile*” é a “*ENTRYPOINT*”. Esta instrução permite configurar uma aplicação que irá ser usada como uma aplicação executável. Apenas a última instrução “*ENTRYPOINT*” será válida no ficheiro “*Dockerfile*”. Por fim, a instrução “*CMD*” permite escolher predefinições para “*containers*” executáveis. Estas podem incluir um executável, ou em caso contrário, este poderá ser indicado com a instrução “*ENTRYPOINT*”.

Para criar as imagens virtuais dos programas em questão, através dos ficheiros “*Dockerfile*”, foram utilizadas imagens oficiais dos vários softwares presentes no repositório DockerHub. Para o servidor e REST API, foi utilizada a imagem do SDK .NET 6.0, para a interface gráfica foi utilizada a imagem do software Node-Red, para o algoritmo preditivo foi utilizada a imagem oficial do sistema operativo Ubuntu 20.04. Para criar a imagem do servidor foi utilizada uma imagem já existente e oficial do SDK NET6.0 e foi copiado para o interior do “*container*” o ficheiro CS-PROJ, que contém informações sobre as configurações e pacotes usados na aplicação. Depois de efetuado este passo, o programa foi restaurado dentro do “*container*” com o comando “*dotnet restore*”. Posto isto, o resto dos ficheiros foram copiados para uma pasta dentro da imagem, ponto a partir do qual se tornou possível publicar a aplicação no âmbito da imagem, através do comando “*dotnet publish*”. Este processo cria um ficheiro executável DLL que ao ser corrido executa a aplicação pretendida. A outra aplicação desenvolvida em C# apresenta uma estrutura do “*Dockerfile*” mais complexa, já que se trata de uma aplicação desenvolvida com base na *framework* ASP NET CORE, mas que necessita do SDK NET6.0 para poder restaurar e publicar o projeto. Felizmente, o software Visual Studio consegue gerar um “*Dockerfile*” de forma automática com uso da opção de apoio Docker. Esta imagem utiliza as duas imagens oficiais das *frameworks* referidas anteriormente de forma intercalada. A este código foram adicionados dois comandos “*COPY*” de ficheiros que permitem completar a autenticação e ter acesso à aplicação. No que toca à interface gráfica, é importante mencionar a existência de diferentes ficheiros relevantes à criação da sua imagem. São estes o “*package.json*”, que contém todos os pacotes que foram instalados para se garantir a funcionalidade da interface gráfica, o “*settings.js*” que contém as configurações pretendidas para a aplicação, o ficheiro “*flows.json*” que contém toda a informação sobre os nós introduzidos no módulo e que fazem a interface gráfica e o ficheiro “*flows_cred.json*” que contém as credenciais que permitem aceder ao programa. Todos estes ficheiros devem ser

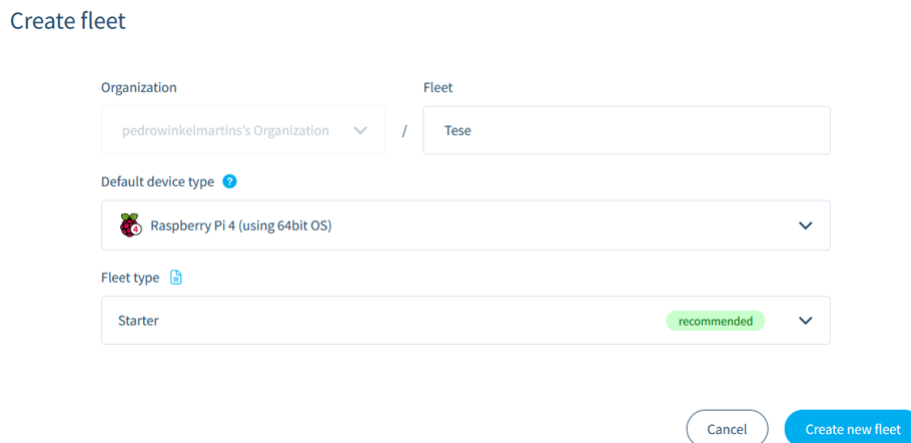
copiados para uma pasta no interior da imagem. Posteriormente, aplica-se um comando *“npm install”* para que todas as dependências e pacotes necessários possam ser instalados. Por fim, foi instalado também o pacote *node-red-node-smooth* que permite ter acesso a pacotes adicionais. Por último, temos que considerar o algoritmo preditivo. Este como já foi referido, foi construído a partir de uma imagem do sistema operativo Ubuntu 20.04. De notar que se deve atender à arquitetura da imagem utilizada. Isto porque, como mais tarde se perceberá, podem ser criadas incompatibilidades com o sistema operativo utilizado no Raspberry Pi 4. Neste caso, para garantir essa compatibilidade usou-se uma imagem com a arquitetura ARM64. Uma vez carregada esta imagem, é efetuada uma atualização dos seus pacotes e são instalados outros que permitem realizar certas operações na linha de comandos, fazer o transporte de pacotes via o protocolo HTTPS e que permitem instalar versões específicas do Python. De seguida realiza-se a instalação do Python e também do pip, que permite que posteriormente sejam instaladas as *frameworks* necessárias ao funcionamento do programa, dentro do *“container”*. Depois de instaladas as *frameworks* necessárias e de copiados tanto o programa desenvolvido em Python, bem como o algoritmo preditivo criado, inicia-se a aplicação.

Uma vez criados os documentos *“Dockerfile”* é possível construir as imagens das diferentes aplicações. Isto é possível, quando se aplica o comando *“docker build”*, no diretório em que se encontra o *“Dockerfile”* pretendido. Depois de criada uma imagem, esta pode ser corrida num *“container”* a partir do comando *“docker run”*. No entanto, se queremos correr múltiplos *“containers”* com diferentes imagens, este processo pode ser bastante moroso e pouco flexível. Neste contexto, pode-se usar a ferramenta *“compose”* para obter uma melhor eficiência neste processo. Esta faz uso de um ficheiro YAML para definir as configurações e dependências das várias aplicações que se pretende que corram em simultâneo. De seguida utiliza-se o comando *“docker-compose up”* para correr todas as aplicações em simultâneo nos seus *“containers”* respetivos.

No documento *“docker-compose.yml”* deve definir-se a versão do *“compose”* que será implementada. No documento desenvolvido no contexto deste projeto foi usada a versão 2, já que é a versão mais recente do *“compose”* que é suportada para o balenaOS (será explicitado mais adiante). Estão assim criadas condições para, de seguida, se definirem os serviços ou aplicações a serem instanciadas nos diferentes *“containers”*. Depois de se definir o nome de cada imagem, pode definir-se a localização do *“Dockerfile”* com a propriedade *“build”*, variáveis de ambiente com a propriedade *“environment”*, condições para reiniciação do *“container”* com a propriedade *“restart”*, dependências sobre outras aplicações com a propriedade *“depends_on”*, portas que permitem a comunicação com outros equipamentos com a propriedade *“ports”* e garantir a persistência dos dados gerados no interior do *“container”* através da propriedade *“volumes”*. Para todas as aplicações foi utilizada a propriedade *“restart”* com o valor *“always”* de forma a que sempre que ocorra um erro estas possam reiniciar. Por outro lado, foram definidas variáveis de ambiente para o servidor, para o modelo preditivo e para a REST API. Nas duas primeiras usou-se a variável *“IPMQTT”* para definir a morada IP do *broker* MQTT. No servidor foram definidas duas variáveis adicionais, a *“IPSQL”* e a *“IPPLC”* que nos dão informação sobre a morada IP do servidor da base de dados e sobre a morada IP da interface ethernet do PLC utilizado. As variáveis de ambiente da REST API são a *“ASPNETCORE_ENVIRONMENT”* e *“ASPNETCORE_Urls”*, que definem respetivamente o ambiente do tempo de execução da aplicação e as portas do *“container”* que comunicam através dos protocolos HTTPS e HTTP. Por outro lado, tanto na REST API como na interface gráfica foram definidas comunicações entre as portas 5000/5001 e 1880, dos *“containers”* e do Raspberry Pi 4, respetivamente. Na REST API, a porta 5000 é utilizada para realizar as comunicações através do protocolo HTTP e a porta 5001 é utilizada para realizar as comunicações através do protocolo HTTPS. Desta forma será possível aceder à interface gráfica bem como à REST API num *browser* de um qualquer dispositivo, conhecendo-se a morada IP do Raspberry Pi 4. Para obter mais informação sobre o software Docker, os *“Dockerfile”* e a ferramenta *“compose”* pode recorrer-se a [16, 44, 45].

3.9 Transporte dos módulos para um Raspberry Pi 4 através da plataforma Balena

O Balena é uma plataforma que permite o desenvolvimento e organização de projetos que implicam a conexão de módulos associados à *Internet of Things* (IOT). Esta apresenta várias ferramentas que, associadas ao Docker permitem correr instâncias de imagens em diferentes dispositivos como um Raspberry Pi 4, por exemplo. Para este efeito a plataforma apresenta uma aplicação, o balenaEtcher, que consegue criar imagens de sistemas operativos e transferi-las para cartões SD que podem posteriormente ser integrados nos dispositivos. Neste caso, o sistema operativo transferido para os cartões SD é o balenaOS, que tem a capacidade de correr instâncias de imagens adaptadas para Linux em “*containers*” Docker. A arquitetura do sistema operativo é a “aarch64”, compatível apenas com arquiteturas ARM Linux. Deste modo justifica-se a escolha anterior de uma imagem Ubuntu 20.04 com arquitetura ARM, para o algoritmo preditivo. Para se conseguir correr “*containers*” nestes dispositivos, é possível utilizar a plataforma *online* balenaCloud, que permite a gestão dos projetos IOT a partir de um *browser*. Dentro desta plataforma é possível criar novos projetos, associá-los a dispositivos, aceder às suas informações mais relevantes e supervisionar o seu funcionamento. Um projeto é composto por uma “*fleet*” ou frota, estando o seu nome associado a um tipo de dispositivo (que permite conhecer a arquitetura dos dispositivos que podem ser adicionados ao projeto) e a um tipo de projeto. Para este projeto, foi criada uma “*fleet*” do tipo *starter* (corresponde a um tipo de projeto não pago e que tem uma limitação no número máximo de 10 dispositivos associados ao projeto), associado a dispositivos Raspberry Pi 4, como se pode verificar pela figura 3.29:



The screenshot shows a web form titled "Create fleet". It has two columns: "Organization" and "Fleet". The "Organization" dropdown is set to "pedrowinkelmartins's Organization". The "Fleet" text input is "Tese". Below these is a "Default device type" dropdown set to "Raspberry Pi 4 (using 64bit OS)". Underneath is a "Fleet type" dropdown set to "Starter", with a green "recommended" badge. At the bottom right, there are "Cancel" and "Create new fleet" buttons.

Figura 3.29: Criação da “*fleet*” de dispositivos Raspberry Pi 4.

De seguida, há condições para associar o projeto a um novo dispositivo. Para se concluir esta etapa deve-se mais uma vez escolher o tipo de dispositivo em questão (Raspberry Pi 4), bem como o sistema operativo que se pretende que fique hospedado nos dispositivos (neste caso opta-se pelo balenaOS). Devem ser ainda definidos os parâmetros (SSID e Password) que permitem fazer a ligação do dispositivo à rede local. Só assim se conseguirá realizar a sua supervisão a partir do balenaCloud. Um exemplo dessa configuração pode ser observada através da figura 3.30:

Figura 3.30: Adição de um novo dispositivo à “fleet”.

No entanto, estas definições não são suficientes para garantir as funcionalidades do sistema proposto. Estas preveem a possibilidade de comunicação com o PLC através de uma ligação física, para a qual o Raspberry não está configurado por definição. Esta pode ser garantida através da criação de um documento semelhante àquele que define a comunicação do Raspberry por Wi-Fi com a balenaCloud. Deste modo, depois de realizado o “flash” do sistema operativo para um cartão SD, deve-se criar um documento na pasta “system-connections” presente neste. O conteúdo do ficheiro utilizado neste projeto é apresentado de seguida:

```
[connection]
id=balena-ethernet-01
type=ethernet
interface-name=eth0
[ethernet]
mac-address-blacklist=
[ipv4]
never-default=true
route-metric=2000
address1=192.168.0.5
method=manual
```

Neste documento é definida a morada IP estática da porta ethernet já anteriormente referida no campo *address1* e são adicionados dois campos “*never-default*” e “*route-metric*” com valores que permitem dar prioridade à conexão do Raspberry Pi 4, à balenaCloud através da conexão Wi-Fi. Consegue-se agora realizar a comunicação do equipamento com o PLC, por ligação física. Quando criados novos dispositivos numa “fleet”, é possível aceder às suas informações e especificações na plataforma balenaCloud. Nesta, podemos encontrar informações sobre o tipo de dispositivo em questão, as moradas IP e MAC, bem como o estado do CPU, da memória, entre outros, como se pode observar pela figura 3.31:

Existindo dispositivos numa “fleet”, é possível carregar os vários serviços do projeto nesses mesmos dispositivos. Para se realizar esta tarefa é necessário, em primeiro lugar, instalar o balena

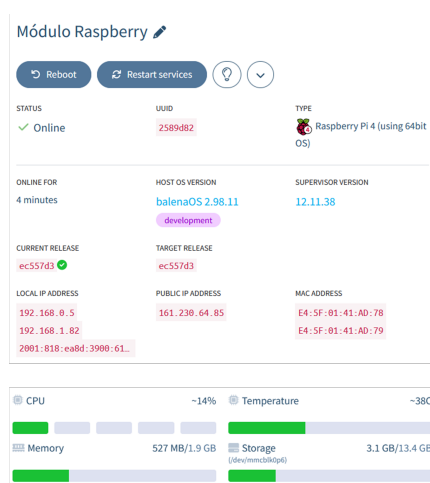


Figura 3.31: Informações relativas ao dispositivo disponibilizadas através da balenaCloud.

CLI que permite criar uma interface numa linha de comandos com a plataforma balenaCloud. Uma vez concluído este passo, deve-se realizar numa linha de comandos o *login*, na plataforma balenaCloud, como se pode visualizar na figura 3.32:

```
C:\Users\VIP\Desktop\Universidade\5 ano\tese>balena login
balena
Logging in to balena-cloud.com
How would you like to login?
> Web authorization (recommended)
Credentials
Authentication token
I don't have a balena account!
```

Figura 3.32: Login na plataforma balenaCloud através da linha de comandos.

Uma vez concluído o *login* na balenaCloud é possível fazer o “*push*” das aplicações pretendidas para os dispositivos de uma “*fleet*” específica. Este comando usa um ficheiro “*docker-compose.yml*”, presente no diretório indicado na linha de comandos, para criar cópias de diferentes serviços discriminados nesse mesmo ficheiro. Caso esta operação seja concluída com sucesso, devem ser criadas diferentes imagens dos serviços discriminados no ficheiro “*docker-compose.yml*”, devendo a mensagem recebida na linha de comandos ser semelhante, àquela presente na figura 3.33:



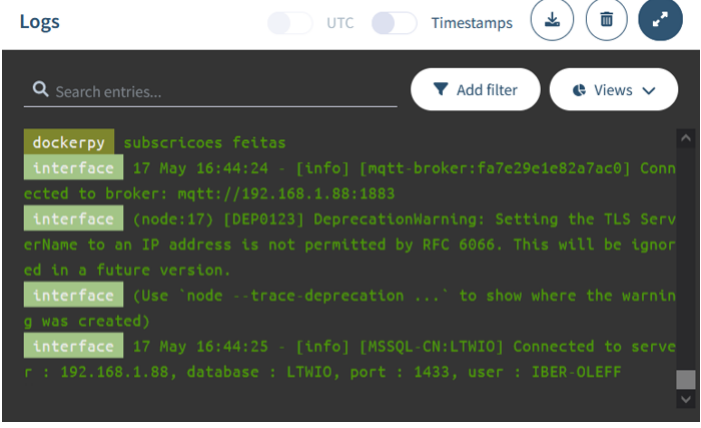
Figura 3.33: Comando balena “push” executado com sucesso.

Uma vez concluída a etapa descrita deve ser possível acompanhar o carregamento dos vários serviços nos dispositivos presentes na “fleet” para a qual o “push” foi efetuado, tal como na figura 3.34. Caso este seja efetuado com sucesso, será possível perceber se os programas estarão a ser executados ou não, a partir do terminal associado ao dispositivo, como representado na figura 3.35. Neste caso, verifica-se que o sistema é capaz de concluir o seu ciclo de tarefas de forma eficaz, desde a receção dos números de série até à comunicação dos parâmetros ao PLC:

Service	Last known status	Release
interface	Downloading 20%	99aa86d
webapi	Downloading 5%	99aa86d
dockkerpy	Downloading 15%	99aa86d
appcsharp	Downloading 0%	99aa86d

Service	Status	Release
appcsharp	Running	ec557d3
dockkerpy	Running	ec557d3
interface	Running	ec557d3
webapi	Running	ec557d3

Figura 3.34: Carregamento dos ficheiros no dispositivo efetuado com sucesso.



The screenshot shows a Docker logs window titled "Logs". At the top, there are toggle switches for "UTC" and "Timestamps", along with icons for download, delete, and refresh. Below the header is a search bar labeled "Search entries...", an "Add filter" button, and a "Views" dropdown menu. The main area displays terminal output for a container named "dockerpy". The output includes the message "subscricoes feitas", followed by an "interface" log entry: "17 May 16:44:24 - [info] [mqtt-broker:fa7e29e1e82a7ac0] Connected to broker: mqtt://192.168.1.88:1883". This is followed by a deprecation warning: "(node:17) [DEP0123] DeprecationWarning: Setting the TLS ServerName to an IP address is not permitted by RFC 6066. This will be ignored in a future version." and a suggestion: "(Use 'node --trace-deprecation ...' to show where the warning was created)". The final log entry is: "17 May 16:44:25 - [info] [MSSQL-CN:LTHIO] Connected to server : 192.168.1.88, database : LTHIO, port : 1433, user : IBER-OLEFF".

Figura 3.35: Mensagens recebidas no terminal que permite avaliar o funcionamento dos diferentes programas.

Para mais informações sobre esta plataforma, pode consultar-se a referência [46].

Esta página foi intencionalmente deixada em branco.

Capítulo 4

Resultados

Uma vez concluída a explicação de como os vários módulos desempenham as suas funções e de como estes foram implementados em conjunto, num Raspberry Pi 4, e perante a impossibilidade de implementar a solução num ambiente de produção, é possível avaliar a sua performance. Verifica-se, em primeiro lugar, que a leitura dos números série de várias peças consegue ser realizado com sucesso. Ainda que fora de um ambiente industrial, o sistema construído consegue ler o número série de diferentes objetos associados a códigos de barras, como garrafas de água, caixas de cartão e outros objetos domésticos com sucesso. A comunicação via protocolo de comunicação MQTT também é efetuada com sucesso, utilizando o computador local como *broker*. Esta realidade, no entanto, não corresponde ao que ocorreria no chão de fábrica da empresa. De facto, o computador local presente ao pé do equipamento de soldadura da empresa não tem a capacidade de funcionar como um *broker* de tópicos MQTT. Deste modo, teria que se determinar uma outra solução para se garantir esta e outras comunicações implementadas através do protocolo MQTT. O servidor responsável pela receção dos números série das peças, consegue receber de forma eficaz a comunicação do módulo ESP8266 e consegue comunicar de forma eficaz com a base de dados presente no computador local. Mais uma vez, como este equipamento não fará parte da linha de montagem, os parâmetros de comunicação com a base de dados terão que ser alterados quando se realizar a sua implementação no chão de fábrica. A extração da informação é feita de forma célere, bem como a comunicação dos parâmetros para o modelo preditivo. Este, por sua vez, é capaz de produzir previsões da força de separação entre duas peças, tendo em conta a deformação associada “*rib*” do *front frame* e os parâmetros padrão de produção de um produto específico. Neste caso estes parâmetros são a velocidade e intensidade do feixe laser e a força de compactação dos dois *frames*. A interface gráfica desenvolvida em Node-Red também é capaz de receber as informações necessárias ao seu funcionamento através do protocolo MQTT. Nesta fase, recebidas as previsões do modelo preditivo, o operador deve, caso deseje, propor novos valores para unir as duas peças pretendidas. Tal como explicitado no capítulo 3, o operador recebe um aviso de quando a força de separação se desvia em 5% do valor padrão, para que este saiba quando deve procurar alterar os parâmetros em relação aos seus valores padrão. Este processo pode ser realizado quantas vezes o operador quiser, sendo mostrada em cada iteração o valor previsto de força de separação entre as peças para os parâmetros propostos e deformação do “*rib*” do *front frame*. Quando o valor de força de separação prevista pelo algoritmo aprovar ao operador este pode premir o botão “ParâmetrosAjustados” para comunicar os últimos parâmetros que propôs no PLC. Caso contrário, e não sendo necessários ajustes aos parâmetros padrão o operador pode selecionar o botão “ParâmetrosPadrão” para comunicar os parâmetros padrão ao PLC. Ainda sobre a interface gráfica, esta é capaz de extrair informação da base de dados e apresentá-la de forma gráfica, para que o operador tenha uma maior perceção de como os parâmetros influenciam o processo de soldadura laser. Para além disso, também é possível atualizar os registos da base de dados (tabela de histórico de produção) a partir da interface gráfica. No que toca à comunicação dos parâmetros sobre o PLC, esta é feita com recurso ao protocolo S7. Infelizmente, o processo que faz atuar os parâmetros sobre o equipamento de soldadura é desconhecido, por

isso é impossível saber se a solução preconizada é passível de ser enquadrada no equipamento no chão de fábrica. No entanto, verifica-se que as configurações da porta ethernet do Raspberry Pi 4 e o código do servidor permitem a atualização das posições de memória do autómato, pelo que o método implementado é capaz de, pelo menos, garantir a comunicação com o autómato por ligação física. Verifica-se que o processo é capaz de ser realizado de forma cíclica, com um uso máximo do CPU do equipamento de 56% (processador Quad-core Cortex-A72 (ARM v8) 64-bit SoC, com capacidade de 1.5 GHz), garantindo-se a reprodutibilidade das suas operações no tempo sem ser necessária uma intervenção exterior.

No que toca ao modelo preditivo mais em concreto, verifica-se que o método de Taguchi pode ser usado com grande efeito para determinar o conjunto de hiperparâmetros que melhor se adequa a um determinado algoritmo. No entanto, este é muito dependente da métrica usada. Como se referiu no capítulo 3 foram usadas duas métricas diferentes, tanto para o algoritmo XGBoost como para as ANN. Verifica-se que ao considerar-se como métrica o valor médio do RMSE nos conjuntos de treino e teste, o método de Taguchi conduz a otimização dos hiperparâmetros para valores que induzem *overfitting* no algoritmo. Isto deve-se maioritariamente ao facto de uma diminuição do RMSE nos conjuntos de treino e uma manutenção do RMSE nos conjuntos de teste não ser penalizada pela métrica usada, bem pelo contrário. Por outro lado, ao considerar-se como métrica apenas o valor de RMSE nos conjuntos de dados de teste, verifica-se que apesar de o valor de exatidão do algoritmo nos conjuntos de treino ser menor, em comparação com a métrica anterior, o valor de exatidão nos conjuntos de teste é em geral maior. Isto revela uma maior capacidade de generalização do algoritmo e, conseqüentemente, uma melhor compreensão do processo e de como os parâmetros o influenciam. Para além de se comparar o comportamento dos algoritmos, considerando duas métricas diferentes, também foram considerados dois conjuntos de dados distintos. O primeiro foi um conjunto de dados obtidos por experiências reais e era composto por 28 registos. O segundo, acrescentava ao primeiro conjunto de dados 200 registos gerados a partir de uma distribuição normal dada pelos valores das médias e covariâncias que podiam ser encontradas no primeiro conjunto de dados. Neste aspeto, verificou-se como seria de esperar uma melhoria da performance do algoritmo, quando treinado com mais registos de dados, uma redução do *overfitting* experienciado e uma menor variação na performance do algoritmo de acordo com o conjunto de dados com o qual era treinado, isto é, uma menor variância. No entanto, deve-se salientar que a geração de dados sintéticos não permite de todo gerar conhecimento sobre um processo. Permite, porém, aumentar a definição e realçar o conhecimento que já se encontrava disponível nos registos iniciais. Após todas as considerações, optou-se pela escolha de um algoritmo de ANN, treinado com um conjunto de dados misto (real + sintético) e avaliado pelo método de Taguchi através do valor de RMSE nos conjuntos de dados de teste. De seguida são apresentados os valores de exatidão e RMSE para cada um dos 10 conjuntos de treino e teste usados, bem como os seus valores médios e desvios padrão na tabela 4.1. Para além disso, são apresentadas graficamente as variações da exatidão do algoritmo e do valor da função de perda com o avançar das iterações do treino do algoritmo, nas figuras 4.1 e 4.2. Por fim é apresentada na figura 4.3 uma imagem do sistema físico implementado.

Tabela 4.1: Valores de exatidão e RMSE para o melhor algoritmo desenvolvido nos 10 conjuntos de dados de treino e teste, com os seus valores médios e desvios padrão também ilustrados.

	Conjuntos de treino e teste										Médias	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.9338	0.9364	0.9333	0.9344	0.9354	0.9345	0.9348	0.9344	0.9352	0.9327	0.9345	0.001
Exatidão teste	0.9297	0.9205	0.933	0.9283	0.9268	0.93	0.9278	0.9296	0.9298	0.9329	0.9288	0.0033
RMSE treino	0.1309	0.1289	0.1305	0.1271	0.1272	0.1288	0.1287	0.1312	0.1292	0.1339	0.1297	0.0019
RMSE teste	0.1303	0.141	0.1353	0.1479	0.1412	0.1341	0.136	0.1348	0.1396	0.1311	0.1371	0.0051
RMSE médio	0.1306	0.135	0.1329	0.1375	0.1342	0.1314	0.1324	0.133	0.1344	0.1325	0.1334	0.0019

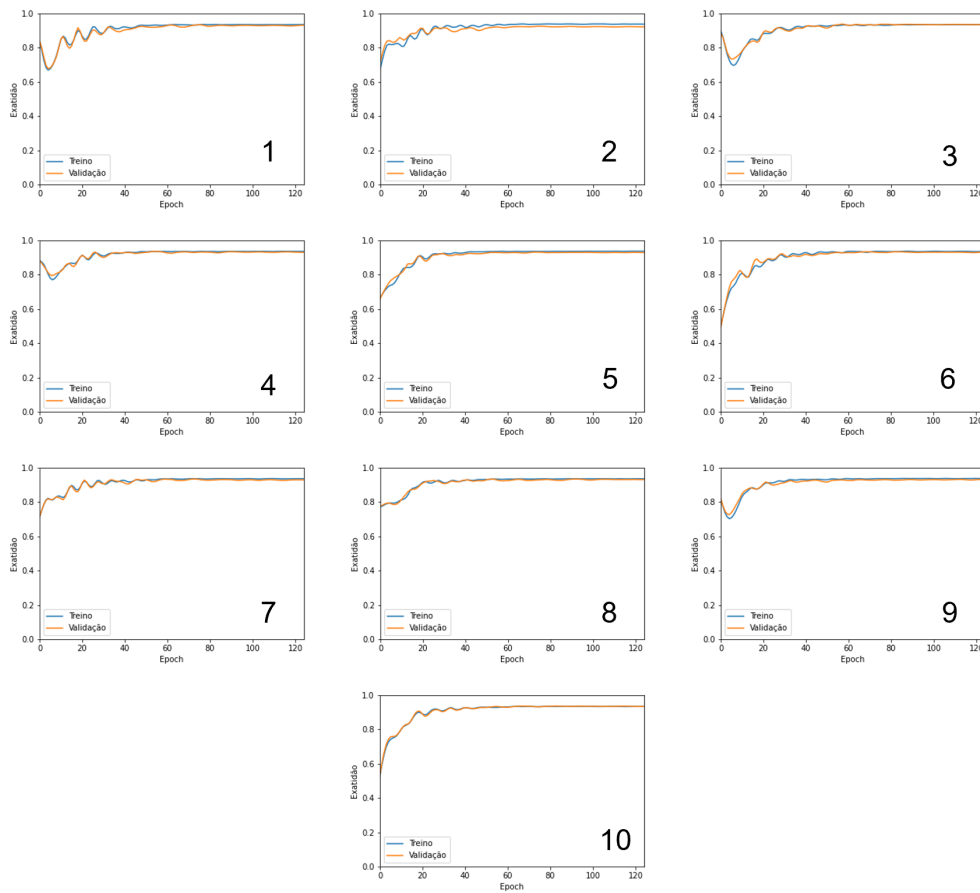


Figura 4.1: Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.

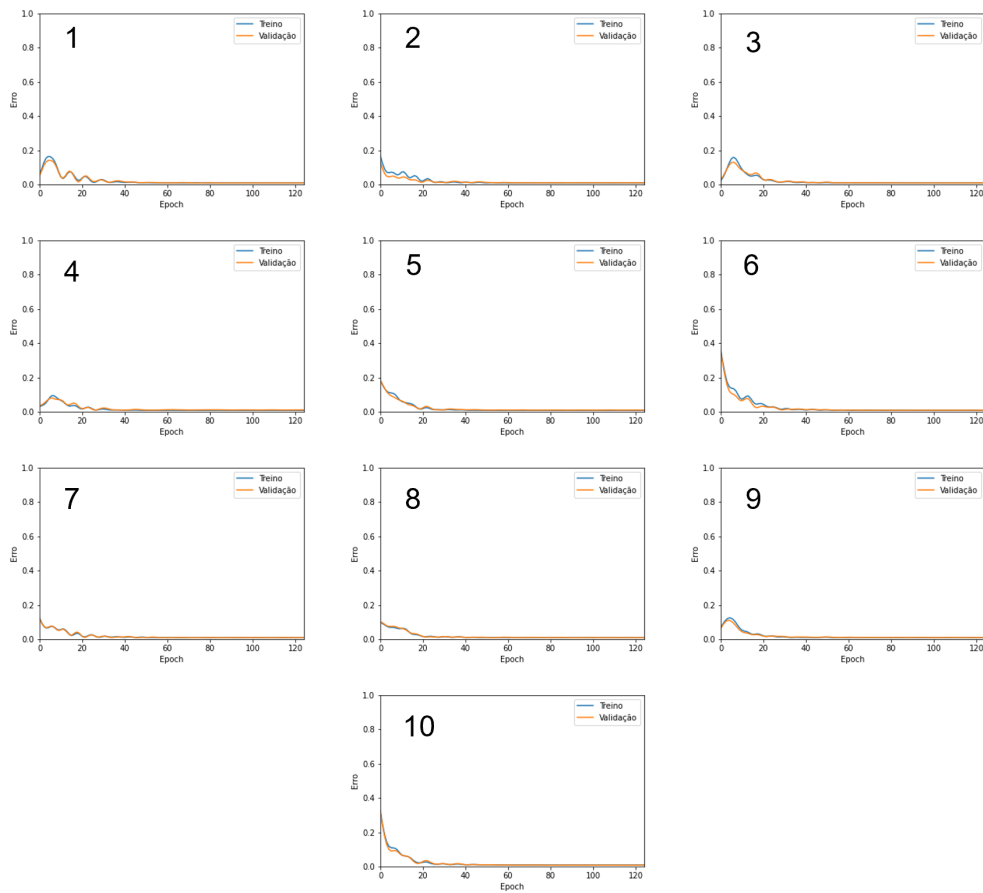


Figura 4.2: Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.

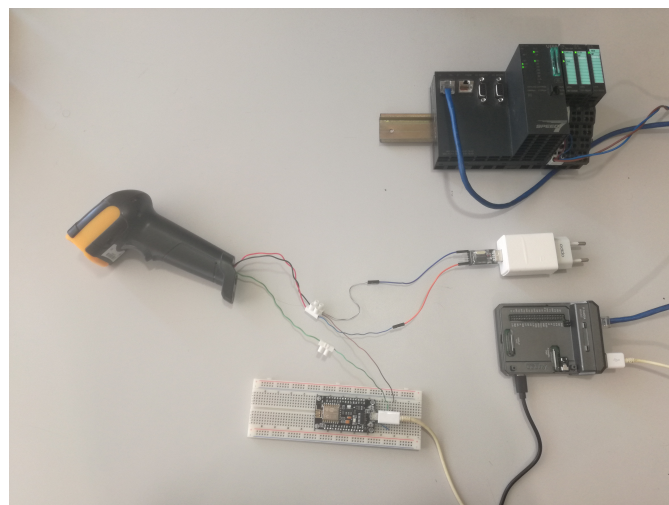


Figura 4.3: Sistema de controlo inteligente.

Capítulo 5

Conclusões

Para finalizar o documento atual serão apresentadas de seguida algumas considerações finais e conclusões que foram extraídas a partir do desenvolvimento do trabalho realizado. Num aspeto mais geral, verifica-se que todos os módulos propostos foram desenvolvidos com sucesso, conseguindo estes interagir uns com os outros de forma eficaz, com um uso máximo do CPU do Raspberry Pi 4 de 56%. A integração das várias aplicações num dispositivo Raspberry Pi 4 pode ser, por isso, considerada um sucesso, permitindo-se a integração do sistema desenvolvido num sistema industrial. No geral, pode-se dizer com confiança que o sistema apresentado contribui de uma forma positiva para o aumento da produtividade do processo de soldadura por transmissão de laser na Iber-Oleff, integrando o conhecimento existente sobre cada peça, auxiliando o operador na sua tomada de decisão e reduzindo os tempos mortos experienciados na produção de peças poliméricas. No entanto, o algoritmo preditivo apresenta um erro nas suas previsões que é bastante elevado. Uma exatidão média de cerca de 93% no conjunto de dados teste significa que para uma força de separação das peças de, por exemplo 800 N, há um erro médio na previsão de cerca de 60 N. Apesar disso, criticamente, o valor de exatidão obtido para o algoritmo é adequado para ser usado num contexto industrial, referindo a Iber-Oleff que uma exatidão superior a 90% é um ótimo valor tendo em conta a complexidade do processo de soldadura por transmissão de laser e a quantidade de ruído associada aos ensaios, a partir dos quais foram extraídos os dados usados para treinar o algoritmo. Considera-se, por isso, que com um conjunto de dados maior e com menos ruído associado, conseguir-se-ia melhorar a exatidão do algoritmo. O uso de 28 registos para o treino e teste de um algoritmo é manifestamente insuficiente, e algo que como já foi referido, a criação de dados sintéticos não consegue corrigir na totalidade. Apesar disto, os resultados obtidos com os dados em questão são encorajadores e pode-se dizer com confiança que com um conjunto de dados maior e mais robusto a metodologia aplicada poderá apresentar resultados excelentes. Podemos concluir, no entanto, que as ANN apresentam-se como o algoritmo que mais se adequa a este tipo de tarefa preditiva, ainda que o algoritmo XGBoost consiga produzir modelos com comportamentos bastante próximos daqueles produzidos pelas ANN. Como processo de otimização de hiperparâmetros, pode-se concluir que o método de Taguchi se apresenta como uma ótima alternativa aos métodos típicos usados para otimizar hiperparâmetros. A sua vantagem principal está em não ter que se treinar o algoritmo com todas as combinações possíveis entre hiperparâmetros para ser conhecida a combinação ótima. Este facto, permite reduzir o tempo que se demora a obter essa combinação, quando comparado com um método como o *Grid Search*. Como métrica usada no método de Taguchi, conclui-se que o uso do erro do algoritmo no conjunto de dados de teste apresenta melhores resultados que o erro médio entre os conjuntos de dados de treino e teste. Verifica-se que a primeira métrica consegue atenuar de forma muito mais profícua o problema de *overfitting*. Este problema também consegue ser combatido através do treino do algoritmo com um conjunto de dados mistos (reais + sintéticos). Ao usar mais dados para treinar o algoritmo verifica-se também um aumento generalizado da performance do algoritmo. No entanto, este aumento é limitado já que os dados gerados sinteticamente, a partir

de uma distribuição normal aproximada dos dados reais, não consegue criar conhecimento que não havia anteriormente. Consegue-se apenas refinar o conhecimento que já existia previamente com o conjunto de dados reais.

No que toca a trabalhos futuros, que possam dar seguimento àquilo que foi desenvolvido neste projeto, sugere-se a realização de mais ensaios que permitam obter mais registos com os quais se possa treinar o modelo preditivo, de forma a conseguir-se obter um sistema com maior exatidão na previsão da força de separação entre duas peças. Para além disso, seria interessante produzir modelos preditivos adicionais que sugerissem valores de parâmetros de produção para que se possa verificar uma certa força de separação entre duas peças. Este tipo de solução permitiria reduzir significativamente a intervenção do operador no processo e, por isso, reduzir ainda mais as perdas de tempo no processo.

Apêndice A

Anexo 1 - Criação da base de dados e tabelas

Para se criar uma base de dados com recurso ao software Microsoft SQL Management Studio, deve-se conseguir identificar a pasta “Databases” no “Object Explorer”. De seguida, deve-se clicar no botão direito do rato por cima da pasta mencionada e selecionar a opção “New Database”, como demonstrado na figura A.1:

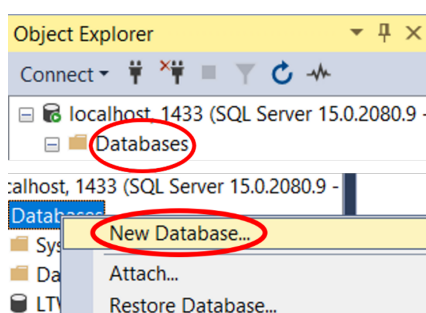


Figura A.1: Criação de uma nova base de dados.

Uma vez concluídos estes passos, será apresentada uma nova página que permite configurar a nova base de dados, como mostrado na figura A.2. Ao ser definido o nome da mesma, são configurados dois ficheiros que permitem armazenar, entre outros, futuras tabelas da base de dados e os seus registos. Nesta altura, também deve ser definido qual o utilizador que “detém” a base de dados. Para se fazer esta seleção, deve-se selecionar o botão com as reticências em frente à caixa de texto em questão. Este botão irá criar uma nova página que permite selecionar esse mesmo proprietário da base de dados. Este poderá ser escolhido tanto pela identificação do seu nome de utilizador na caixa de texto apresentada, tanto através do botão “Browse”. Este apresenta uma nova página que discrimina todos os utilizadores disponíveis para serem proprietários, podendo selecionar-se o proprietário pretendido através de uma caixa de seleção associada ao utilizador em questão. Na imagem seguinte é apresentada a página de configuração da base de dados, com os campos do nome da base de dados e com o botão que permite aceder à página de escolha do proprietário da base de dados realçadas. Esta é seguida por essa mesma página, com o botão “Browse” realçado. Por fim, é apresentada a página que permite escolher o proprietário da base de dados com base em caixas de seleção. Ambas estas páginas podem ser encontradas na figura A.3:

Depois de criada a base de dados, devem ser criadas as relações que permitem armazenar a informação pretendida. Para criar uma tabela, deve-se procurar a pasta “Tables” no “Object Explorer” do SQL Server Management Studio e clicar no botão direito do rato do computador

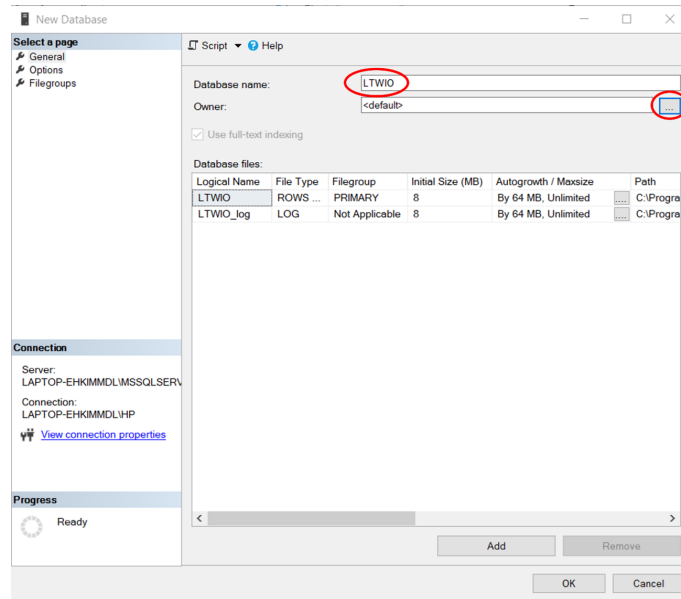


Figura A.2: Página de configuração da nova base de dados.

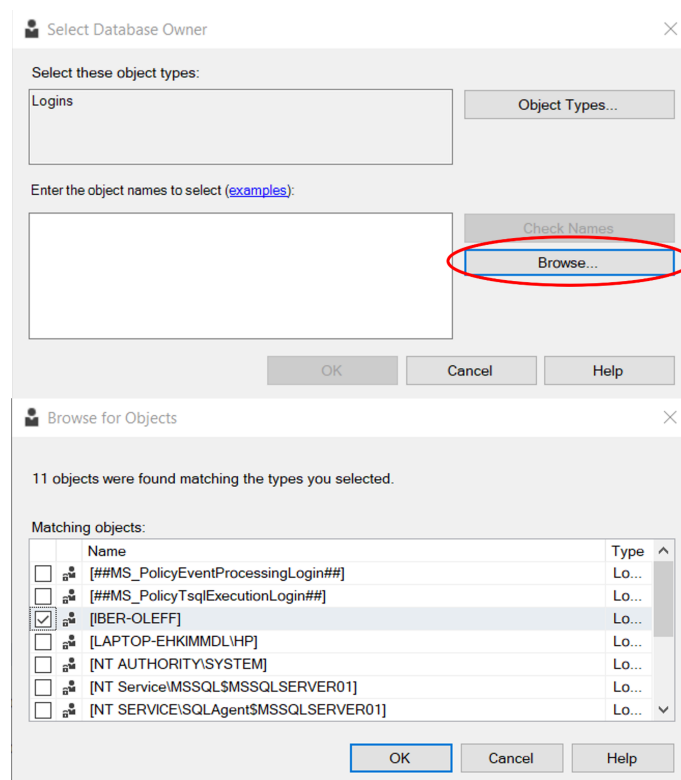


Figura A.3: Definição do utilizador principal da base de dados.

por cima dessa mesma pasta. De seguida deve ser seleccionada a opção “New-Table...”. Uma vez escolhida esta opção é apresentado um ficheiro de configuração da tabela, onde devem ser introduzidos os nomes das colunas da tabela e qual o tipo de dados associados a essa mesma coluna. Uma vez preenchidos estes campos, deve escolher-se uma chave primária para a tabela.

Para esse efeito, basta premir o botão direito do cursor por cima da linha que configura o atributo que se pretende que seja a chave primária e selecionar a opção “*Set Primary Key*”. Antes deste passo, deve-se garantir que o campo “*Allow Nulls*” não se encontra selecionado para esse atributo. Posto isto, e premindo os botões Ctrl+S, é possível atribuir um nome e guardar a tabela criada. De seguida são apresentas imagens da localização da pasta “*Tables*” no “*Object Explorer*”, a localização da opção “*New-Table...*”, o ficheiro de configuração da tabela, a opção de seleção da chave primária e por fim a janela de definição do nome da tabela. A figura A.4 apresenta os vários passos necessários à criação de uma nova tabela na base de dados:

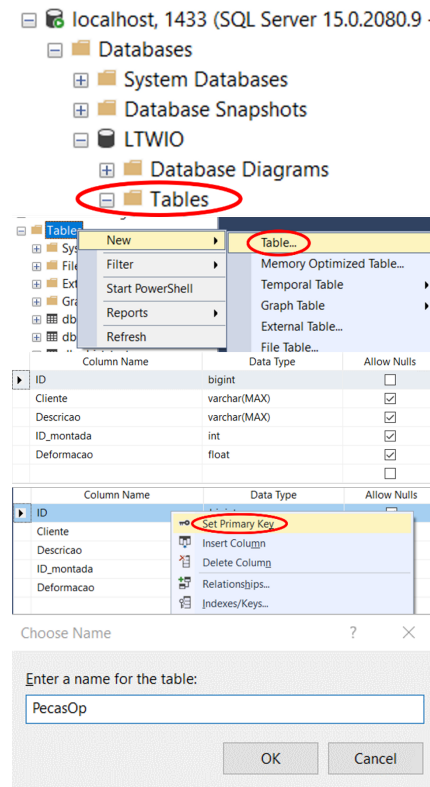


Figura A.4: Criação de uma nova tabela na base de dados.

No que diz respeito ao acesso à base de dados, este pode ser conseguido por duas vias: a “*Windows Authentication*” e a “*SQL Server Authentication*”. A “*Windows Authentication*” permite que um conjunto de computadores presentes num mesmo domínio tenham acesso à base de dados. Neste tipo de autenticação não é necessário introduzir um nome de utilizador nem palavra-passe, já que estes dados encontram-se gravados no “*Active Directory*” do domínio. Para permitir que a base de dados seja acedida por outros dispositivos, não pertencentes ao domínio, pode-se usar o tipo de autenticação “*SQL Server Authentication*”. Neste caso, é necessário sempre que se inicia sessão na base de dados, identificar o nome de utilizador e a palavra-passe. Ambos estes parâmetros devem ser criados para se conseguir aceder à base de dados por esta via. Para este efeito, deve-se utilizar o SQL Server Management Studio e procurar a pasta que diga respeito aos “*Logins*” e aceder à opção que permite criar um novo “*Login*” com o premir do botão direito do cursor, sobre essa pasta. A imagem na figura A.5 demonstra onde esta pode ser encontrada no “*Object Explorer*”:

Depois de realizados estes passos, é mostrada uma página que nos permite fazer a configuração do novo “*Login*”. Deve ser selecionada a opção “*SQL Server Authentication*”, introduzidos o nome de utilizador e palavra-passe e selecionada a base de dados padrão para o “*Login*”. De seguida, é apresentada na figura A.6 essa imagem, bem como identificados os campos que devem ser

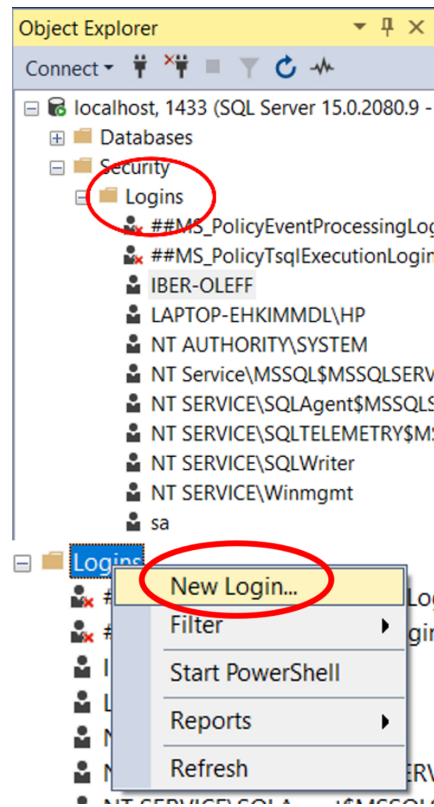


Figura A.5: Criação de um novo "Login".

selecionados ou preenchidos:

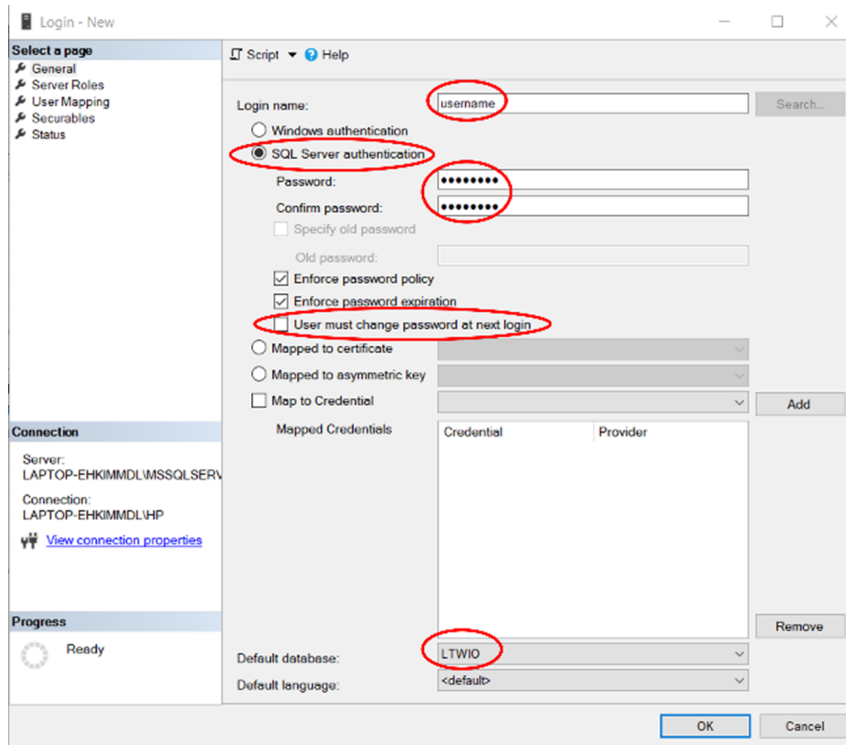


Figura A.6: Configuração do novo "Login".

Este tipo de autenticação, ainda que menos segura, permitiu realizar o acesso à base de dados a partir de um Raspberry Pi 4, que de outra forma (com "Windows Authentication") não teria sido possível.

Esta página foi intencionalmente deixada em branco.

Apêndice B

Anexo 2 - Ensaios de otimização de hiperparâmetros para o algoritmo preditivo

B.1 Ensaios redes neuronais artificiais

B.1.1 Conjunto de dados reais

Média entre o RMSE nos conjuntos de dados de treino e teste como métrica

No que diz respeito à rede neuronal associada aos 28 registos que foram extraídos do projeto PolyWeld foram realizados os ensaios de Taguchi, utilizando como métrica para calcular o rácio S/N, a média entre o erro quadrático médio nos conjuntos de dados de treino e teste que se encontram na tabela B.1. A partir do design experimental presente na tabela 3.2 foi possível desenvolver-se o cálculo do rácio S/N associado a cada experiência. Cada uma destas foi repetida 10 vezes, uma vez para cada conjunto de dados de treino e teste. Depois de efetuado este cálculo para cada experiência, realizou-se um cálculo da média do valor de S/N para cada nível de cada variável em particular. Estes valores permitem montar uma experiência adicional com os níveis das variáveis que maximizam o valor de S/N. Uma vez descoberto o valor de S/N para essa experiência é avaliado qual o maior S/N, construindo-se os níveis dos seguintes ensaios de Taguchi a partir dos níveis do ensaio em questão, de modo a conseguir-se uma convergência dos valores de S/N e obter-se o conjunto ótimo de hiperparâmetros para o algoritmo. Na tabela B.2 são apresentados os valores de S/N para cada ensaio de Taguchi. Os valores máximos de S/N encontram-se evidenciados a negrito. Tal como referido anteriormente, calculou-se a média do valor de S/N para cada nível de cada parâmetro a controlar. Os resultados desse processo são apresentados na tabela B.3. Tal como na tabela B.2, os valores máximos de S/N, neste caso, para cada variável em cada ensaio serão apresentados a negrito. Uma vez determinados os máximos, apresentados na tabela B.3, estes foram combinados e aplicados ao algoritmo de forma a perceber se conseguiam apresentar erros menores que os ensaios previamente efetuados. Como se pode verificar pela tabela B.4, nenhum destes conjuntos resulta em valores de S/N superiores aos valores máximos apresentados nos ensaios L27. Deste modo, para o segundo ensaio L27, foram utilizados como base para a definição dos níveis dos hiperparâmetros, o conjunto apresentado na 10^a experiência do primeiro ensaio L27, enquanto que para os terceiro, quarto e quinto ensaio foram utilizados os conjuntos da 9^a experiência do segundo ensaio L27, da 10^a experiência do terceiro ensaio L27 e da 12^a experiência do quarto ensaio L27, respetivamente. Não foram realizados mais ensaios, uma vez que o valor máximo de S/N no quinto ensaio foi menor que o valor apresentado no quarto ensaio, pelo que se considerou que o conjunto de hiperparâmetros que resultou no máximo valor de S/N nesse mesmo ensaio representa a configuração ótima do

algoritmo. Trata-se esta da configuração da 12^a experiência do quarto ensaio. Por fim na tabela B.5, são apresentados os valores de erro quadrático médio e exatidão para cada conjunto de dados de treino e teste, bem como os seus valores médios e desvios padrão e também, nas figuras B.1 e B.2 os gráficos que permitem visualizar o comportamento do algoritmo durante o processo de aprendizagem (evolução da função de perda considerada e da exatidão do algoritmo). O processo explicitado para este conjunto de dados é idêntico para todos os restantes conjuntos de dados e métricas considerados, tanto para o caso das ANN, como para o caso do XGBoost (neste caso, para visualização gráfica apenas serão disponibilizadas as previsões dos algoritmos quando comparadas com os valores reais das variáveis em estudo). Deste modo, nas próximas secções serão apenas apresentadas as tabelas e gráficos relevantes que apresentam o mesmo tipo de informação que as tabelas e imagens que esta secção contém.

Tabela B.1: Ensaios conduzidos com o conjunto de dados reais e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso das ANN.

	Parâmetros	Níveis		
		1	2	3
Ensaio 1	n_hidden	75	100	125
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	2	3	4
	loss	MSE	MAE	HUBER
	lr	0.01	0.001	0.0001
	m	0.8	0.9	1
	epoch	75	100	125
	p	0	0.25	0.5
Ensaio 2	n_hidden	85	100	115
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	2	3	4
	loss	MSE	MAE	HUBER
	lr	0.015	0.01	0.005
	m	0.85	0.9	0.95
	epoch	100	125	150
	p	0	0.1	0.25
Ensaio 3	n_hidden	80	85	90
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	3	4	5
	loss	MSE	MAE	HUBER
	lr	0.008	0.005	0.003
	m	0.92	0.95	0.98
	epoch	125	150	175
	p	0.05	0.1	0.15
Ensaio 3	n_hidden	82	85	88
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	3	4	5
	loss	MSE	MAE	HUBER
	lr	0.009	0.008	0.007
	m	0.93	0.95	0.97
	epoch	150	175	200
	p	0.025	0.05	0.075
Ensaio 4	n_hidden	84	85	86

Continua na página seguinte

Tabela B.1 – continuação da página anterior

Parâmetros	Níveis		
	1	2	3
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	3	4	5
loss	MSE	MAE	HUBER
lr	0.008	0.007	0.005
m	0.91	0.93	0.95
epoch	165	175	185
p	0.05	0.075	0.1

Tabela B.2: Valores de S/N associados aos ensaios com os conjuntos de dados reais e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para as ANN.

	Taguchi					
	1	2	3	4	5	
Experiências	1	14.50474	14.54148	15.55813	15.96618	16.05286
	2	13.45336	15.56565	16.03863	16.32075	16.15237
	3	6.94747	15.54272	15.67121	15.91062	16.08144
	4	12.66061	10.01353	11.44826	10.10289	11.27337
	5	13.62647	8.685949	11.79273	11.52516	12.11795
	6	10.79583	12.36668	11.92081	9.465664	13.12302
	7	12.40253	14.87797	15.40703	15.92063	15.57226
	8	12.80587	15.21315	16.06618	16.401	16.09454
	9	10.80921	15.96095	15.3081	16.26494	16.11331
	10	14.68972	13.84589	16.41673	16.00327	15.92801
	11	9.24151	14.3948	13.38403	15.32947	15.15234
	12	6.563702	13.94303	15.20996	16.69531	15.97213
	13	13.35111	13.93484	14.53257	15.47575	15.22903
	14	5.360298	6.420988	11.37399	13.73501	15.01147
	15	6.101915	14.1888	15.64091	15.33318	15.56813
	16	12.15676	10.77557	13.49264	13.87245	14.19939
	17	8.818052	12.82525	11.10966	13.47319	13.86817
	18	7.813037	14.41108	15.08542	14.24285	15.1845
	19	8.135894	13.11384	10.7335	12.36218	14.26931
20	13.17053	15.14407	15.40998	14.72993	14.66001	
Experiências	21	7.052021	14.55054	15.45081	13.58175	14.78699
	22	-1.46302	14.01308	13.17246	15.32556	15.65672
	23	9.464966	14.65599	14.87887	15.19013	15.27346
	24	6.280421	13.47489	15.21104	14.90479	15.09642
	25	-12.2538	11.27189	12.5166	15.12322	15.65605
	26	13.20082	14.42715	16.21471	15.88412	16.16885
	27	11.39926	15.04735	15.55798	15.92532	15.9723

Tabela B.3: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.

	Nível	Hiperparâmetro							
		n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p
Ensaio 1	1	12.00068	10.41766	8.663976	8.007244	8.242729	10.65365	9.746606	10.59259
	2	9.344011	8.464288	8.880461	10.46991	11.01576	11.64611	6.087048	10.0484
	3	6.109679	8.572418	9.90993	8.977211	8.195875	5.154606	11.62071	6.813381
Ensaio 2	1	13.6409	14.51578	13.97841	13.43787	12.93201	14.02257	12.9052	13.77746
	2	12.74892	11.97275	12.66625	12.43183	13.037	13.45487	13.3785	14.27615
	3	13.96653	13.86782	13.71168	14.48664	14.38734	12.87891	14.07265	12.30274
Ensaio 3	1	14.35679	14.87477	14.46867	14.78941	13.69755	14.9837	13.83344	14.31907
	2	14.02732	13.33018	13.82909	12.9382	14.02986	14.95103	14.02849	14.43749
	3	14.34955	14.5287	14.4359	15.00604	15.00625	12.79893	14.87173	13.97709
Ensaio 4	1	14.20865	15.21105	15.02295	15.51935	14.46135	14.89614	14.32658	14.42151
	2	14.90672	13.4509	14.00605	12.59512	14.73208	14.89003	14.81436	14.85749
	3	14.78078	15.23419	14.86715	15.78168	14.70271	14.10998	14.75521	14.61714
Ensaio 5	1	14.73124	15.45061	15.28504	15.76583	14.87078	15.08062	14.75402	15.12992
	2	15.12368	14.26106	14.596	13.7203	14.94435	15.06411	15.18574	15.04377
	3	15.28224	15.42549	15.25612	15.65102	15.32203	14.99242	15.19739	14.96346

Tabela B.4: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.

		Hiperparâmetros								S/N
		n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p	
Níveis	Ensaio 1	1	1	3	2	2	2	3	1	12.41076
	Ensaio 2	3	1	1	3	3	1	3	2	15.45938
	Ensaio 3	1	1	1	3	3	1	3	2	15.52792
	Ensaio 4	2	3	1	3	2	2	2	2	15.75572
	Ensaio 5	3	1	1	1	3	1	3	1	16.06528

Tabela B.5: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.

	Conjuntos de treino e teste										Médias	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.958	0.976	0.955	0.981	0.956	0.979	0.968	0.968	0.975	0.963	0.9679	0.0092
Exatidão teste	0.827	0.892	0.878	0.861	0.893	0.871	0.906	0.901	0.915	0.888	0.8832	0.0242
RMSE treino	0.087	0.053	0.098	0.04	0.088	0.051	0.063	0.068	0.065	0.075	0.0688	0.0174
RMSE teste	0.324	0.196	0.239	0.288	0.184	0.25	0.163	0.172	0.171	0.199	0.2186	0.0519
RMSE médio	0.2055	0.1245	0.1685	0.164	0.136	0.1505	0.113	0.12	0.118	0.137	0.1437	0.0274

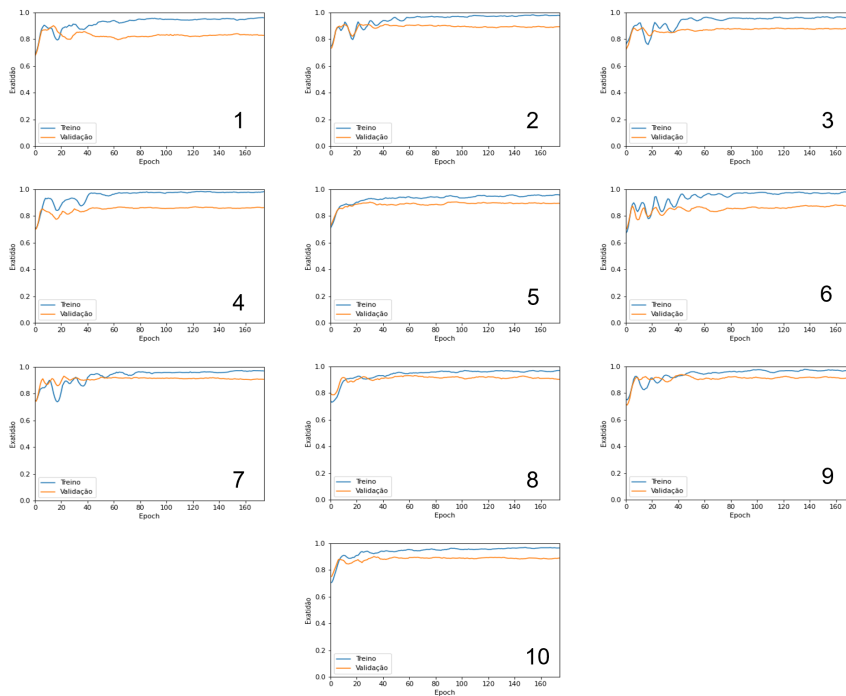


Figura B.1: Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.

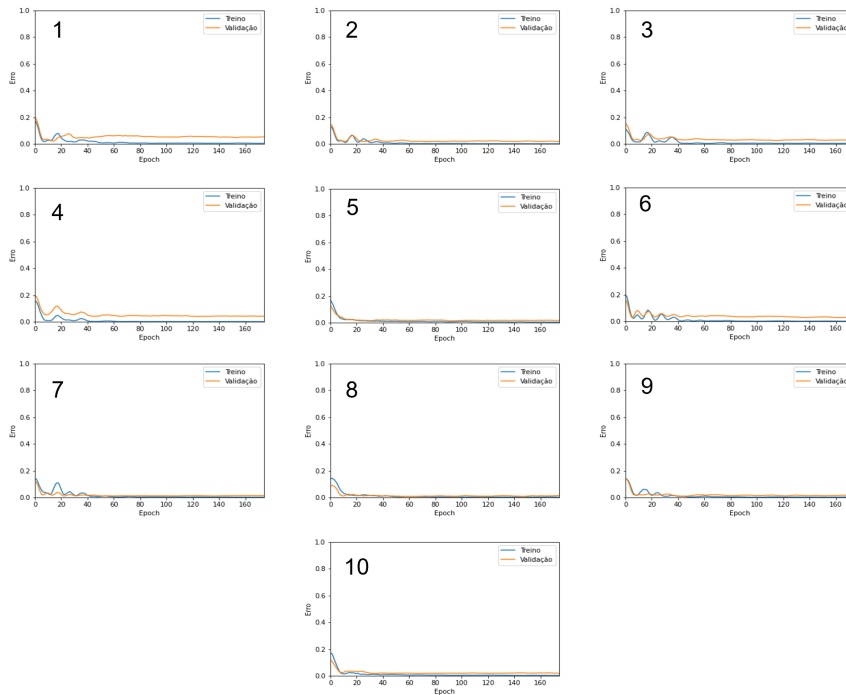


Figura B.2: Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.

RMSE nos conjuntos de dados de teste como métrica

Tabela B.6: Ensaios conduzidos com o conjunto de dados reais e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso das ANN.

Parâmetros	Níveis		
	1	2	3
n_hidden	75	100	125
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	2	3	4
loss	MSE	MAE	HUBER
lr	0.01	0.001	0.0001
m	0.8	0.9	1
epoch	75	100	125
p	0	0.25	0.5
n_hidden	60	75	90
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	2	3	4
loss	MSE	MAE	HUBER
lr	0.005	0.001	0.0005
m	0.85	0.9	0.95
epoch	85	100	115
p	0.25	0.5	0.75
n_hidden	50	60	70

Continua na página seguinte

Tabela B.6 – continuação da página anterior

Parâmetros	Níveis		
	1	2	3
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	2	3	4
loss	MSE	MAE	HUBER
lr	0.0075	0.005	0.0025
m	0.8	0.85	0.9
epoch	70	85	100
p	0.35	0.5	0.65

Tabela B.7: Valores de S/N associados aos ensaios com os conjuntos de dados reais e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para as ANN.

	Taguchi		
	1	2	3
1	11.53667	12.3395	12.74023
2	11.41376	11.7144	12.59407
3	7.033098	10.44162	12.14701
4	11.60251	13.89844	13.58593
5	12.97771	12.17548	13.32063
6	10.42097	12.92432	13.12762
7	12.06769	10.97529	11.5162
8	10.52671	11.96373	12.24133
9	10.10367	10.80332	11.42192
10	12.36434	13.06956	12.3245
11	9.482106	11.37556	11.93248
12	6.612928	7.905994	9.697318
13	11.98724	13.04679	12.70544
14	4.706767	11.70465	13.0714
15	5.63518	10.39467	13.21321
16	11.59973	13.23385	13.24203
17	8.272949	11.54124	12.96364
18	7.703537	10.95376	11.73092
19	9.523216	12.53216	13.42775
20	11.8734	11.67296	12.44969
21	7.079586	8.846731	10.62076
22	-1.91467	12.66	13.31168
23	9.050441	10.10899	13.64272
24	6.12398	8.413207	12.25571
25	-10.0082	11.29199	12.83479
26	10.6944	11.9799	12.48123
27	10.74973	10.46427	11.90557

Tabela B.10: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.

	Conjuntos de treino e teste										Médias	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.91	0.91	0.91	0.93	0.88	0.91	0.88	0.9	0.9	0.91	0.904	0.0143
Exatidão teste	0.87	0.91	0.88	0.89	0.93	0.89	0.94	0.86	0.92	0.9	0.899	0.0247

RMSE treino	0.172	0.202	0.176	0.144	0.226	0.199	0.224	0.198	0.206	0.181	0.1928	0.0237
RMSE teste	0.278	0.165	0.262	0.232	0.133	0.18	0.11	0.234	0.151	0.202	0.1947	0.0533
RMSE médio	0.225	0.1835	0.219	0.188	0.1795	0.1895	0.167	0.216	0.1785	0.1915	0.1938	0.0185

Tabela B.8: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.

Nível	Hiperparâmetro								
	n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p	
Ensaio 1	1	10.85365	9.657679	7.868833	7.083182	7.639835	9.641863	9.069111	9.45538
	2	8.707197	7.843348	8.321832	10.11707	9.888695	10.53587	5.830018	9.222366
	3	5.907987	7.967802	9.278163	8.268577	7.940298	5.2911	10.5697	6.791083
Ensaio 2	1	11.91512	11.09983	11.2674	11.48642	12.56084	11.13661	11.0621	11.68425
	2	11.46956	11.70295	11.67617	11.97544	11.58188	11.43645	11.28802	11.84328
	3	10.88558	11.46749	11.3267	10.80841	10.12754	11.69721	11.92015	10.74273
Ensaio 3	1	12.52166	11.99264	12.73645	12.63255	12.85428	12.33972	12.28799	12.7528
	2	12.3201	13.13715	12.35667	12.71877	12.74413	12.35667	12.48575	12.40419
	3	12.54777	12.25974	12.29641	12.03821	11.79111	12.69314	12.6158	12.23254

Tabela B.9: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.

Níveis	Hiperparâmetros									S/N
	n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p		
Ensaio 1	1	1	3	2	2	2	3	1	11.89305	
Ensaio 2	1	2	2	2	1	3	3	2	10.90355	
Ensaio 3	3	2	1	2	1	3	3	1	10.25613	

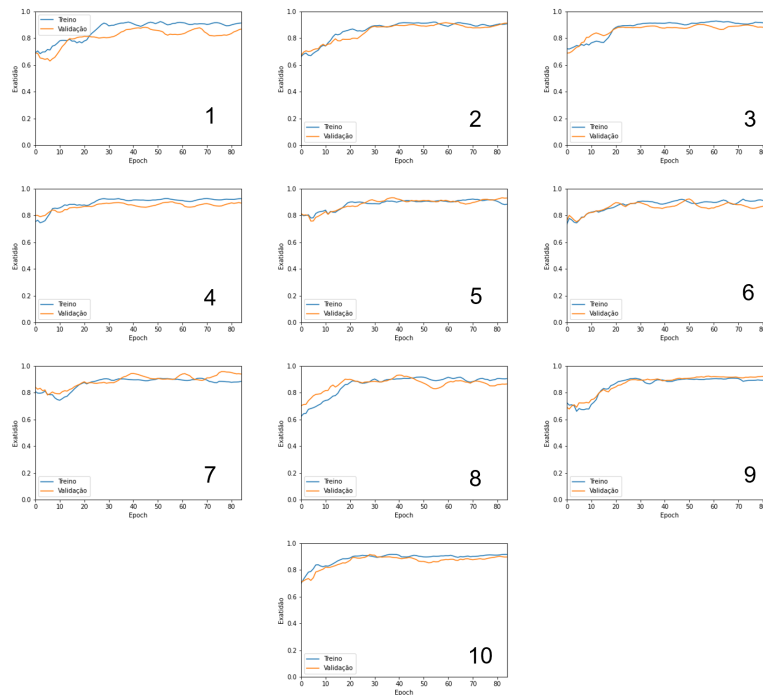


Figura B.3: Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.

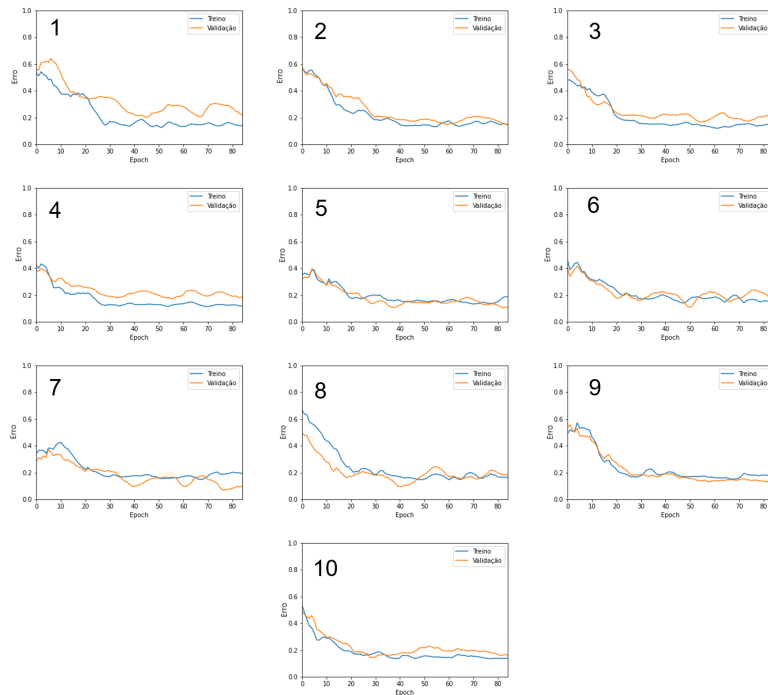


Figura B.4: Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.

B.1.2 Conjunto de dados mistos

Média entre o RMSE nos conjuntos de dados de treino e teste como métrica

Tabela B.11: Ensaios conduzidos com o conjunto de dados mistos e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso das ANN.

Parâmetros		Níveis		
		1	2	3
Ensaio 1	n_hidden	75	100	125
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	2	3	4
	loss	MSE	MAE	HUBER
	lr	0.01	0.001	0.0001
	m	0.8	0.9	1
	epoch	75	100	125
	p	0	0.25	0.5
Ensaio 2	n_hidden	85	100	115
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	2	3	4
	loss	MSE	MAE	HUBER
	lr	0.015	0.01	0.005
	epoch	100	125	150

Continua na página seguinte

Tabela B.11 – continuação da página anterior

Parâmetros	Níveis		
	1	2	3
p	0	0.1	0.25
n_hidden	80	85	90
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	3	4	5
loss	MSE	MAE	HUBER
lr	0.012	0.01	0.008
m	0.88	0.9	0.92
epoch	115	125	135
p	0	0.05	0.1
n_hidden	77	80	83
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	4	5	6
loss	MSE	MAE	HUBER
lr	0.011	0.01	0.009
m	0.89	0.9	0.91
epoch	120	125	130
p	0	0.025	0.05
n_hidden	80	83	85
fa_hidden	RELU	SELU	LEAKY RELU
hidden_layer	5	6	7
loss	MSE	MAE	HUBER
lr	0.01	0.01	0.01
m	0.88	0.89	0.9
epoch	120	125	130
p	0.01	0.025	0.04

Tabela B.12: Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para as ANN.

	Taguchi				
	1	2	3	4	5
1	17.24011	17.81567	18.09106	18.06102	18.70098
2	15.56417	17.7867	18.02666	18.4542	18.72322
3	6.652509	17.62323	17.64997	18.21117	18.48274
4	17.11776	17.16053	17.11499	17.30667	17.01541
5	16.86217	17.06292	16.73906	17.20123	16.94122
6	10.36471	17.38793	16.68872	16.06048	17.01288
7	13.99395	16.58407	17.32429	17.72357	17.73312
8	16.08869	18.15707	18.24784	18.21408	18.28293
9	11.56572	17.52371	17.70117	18.03523	18.11687
10	17.641	18.09144	17.63945	16.89701	18.20725
11	8.723577	17.31508	17.80101	18.12619	17.77905
12	8.275391	15.87764	17.33914	17.86192	18.05054
13	17.21963	17.42861	17.55421	17.71688	17.74086
14	8.590712	17.06672	17.37703	17.48542	17.5618
15	7.268849	17.77716	17.90976	17.9724	18.08675

16	17.0824	16.96189	17.3068	17.59937	17.75148
17	7.745729	16.97101	17.7884	17.66774	18.1493
18	10.54334	17.37747	17.64698	17.90656	18.06324
19	9.591648	16.42727	18.13248	17.87296	18.86977
20	14.54506	17.84575	18.41526	18.64254	18.74481
21	9.047765	16.84277	17.56442	18.30227	18.28713
22	8.224113	17.47045	17.61639	17.74264	17.73709
23	11.78979	17.44159	17.4862	17.52592	17.60603
24	6.922051	17.41975	17.64968	17.76588	17.93912
25	10.54622	17.36724	17.63363	18.0535	18.04271
26	16.71458	18.10465	18.2721	18.43705	18.50873
27	11.93297	17.40307	17.7508	18.06194	18.30157

Tabela B.13: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.

Nível	Hiperparâmetro								
	n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p	
Ensaio 1	1	13.93886	11.92014	11.30713	12.41442	14.2952	13.05431	11.25718	12.17526
	2	11.45451	11.59553	13.13093	12.54451	12.95828	14.26232	11.44051	12.82626
	3	11.03491	12.91262	11.99023	11.46936	9.174813	9.111661	13.7306	11.42677
Ensaio 2	1	17.45576	17.29173	17.42975	17.59701	17.25635	17.33162	17.17541	17.57244
	2	17.20745	17.3573	17.30783	17.11528	17.52794	17.46158	17.25599	17.47904
	3	17.36917	17.38335	17.29479	17.32009	17.24808	17.23918	17.60098	16.9809
Ensaio 3	1	17.50931	17.85105	17.69579	17.80725	17.60148	17.73331	17.60685	17.82439
	2	17.59586	17.34845	17.4421	17.48857	17.79484	17.60877	17.6991	17.73639
	3	17.83566	17.74133	17.80294	17.64502	17.54451	17.59876	17.63487	17.38006
Ensaio 4	1	17.69641	18.0477	17.88161	18.0504	17.66373	17.93752	17.83341	17.66096
	2	17.69261	17.41972	17.55622	17.61776	17.97271	17.80143	17.91994	17.99921
	3	18.04497	17.96656	17.99615	17.76583	17.79754	17.69504	17.68063	17.77382
Ensaio 5	1	17.88993	18.42728	18.12813	18.23882	17.97763	18.05662	17.94083	18.1953
	2	17.93225	17.51568	17.76215	17.87058	18.03301	18.01942	18.08861	18.02468
	3	18.22633	18.10555	18.15823	17.93911	18.03787	17.97247	18.01907	17.82853

Tabela B.14: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.

Níveis	Hiperparâmetro									S/N
	n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p		
Ensaio 1	0	2	1	1	0	1	2	1	13.97292	
Ensaio 2	2	2	0	0	1	1	2	0	17.84084	
Ensaio 3	2	0	2	0	1	0	1	0	17.45609	
Ensaio 4	2	0	2	0	1	0	1	1	18.97936	
Ensaio 5	2	0	2	0	1	0	1	0	18.58273	

Tabela B.15: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para as ANN.

	Conjuntos de treino e teste										Média	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.97	0.97	0.96	0.97	0.97	0.97	0.97	0.97	0.96	0.96	0.967	0.0046
Exatidão teste	0.92	0.91	0.92	0.92	0.92	0.92	0.91	0.92	0.92	0.93	0.919	0.0054
RMSE treino	0.064	0.064	0.071	0.067	0.056	0.067	0.062	0.065	0.072	0.078	0.0666	0.0057
RMSE teste	0.151	0.151	0.155	0.176	0.156	0.151	0.169	0.164	0.168	0.14	0.1581	0.0103
RMSE médio	0.1075	0.1075	0.113	0.1215	0.106	0.109	0.1155	0.1145	0.12	0.109	0.1124	0.0052

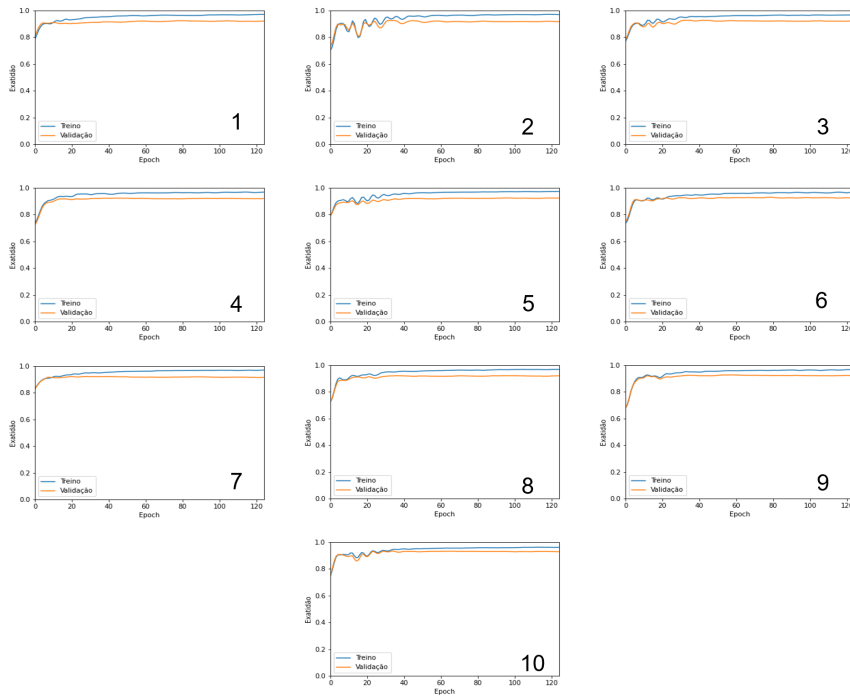


Figura B.5: Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.

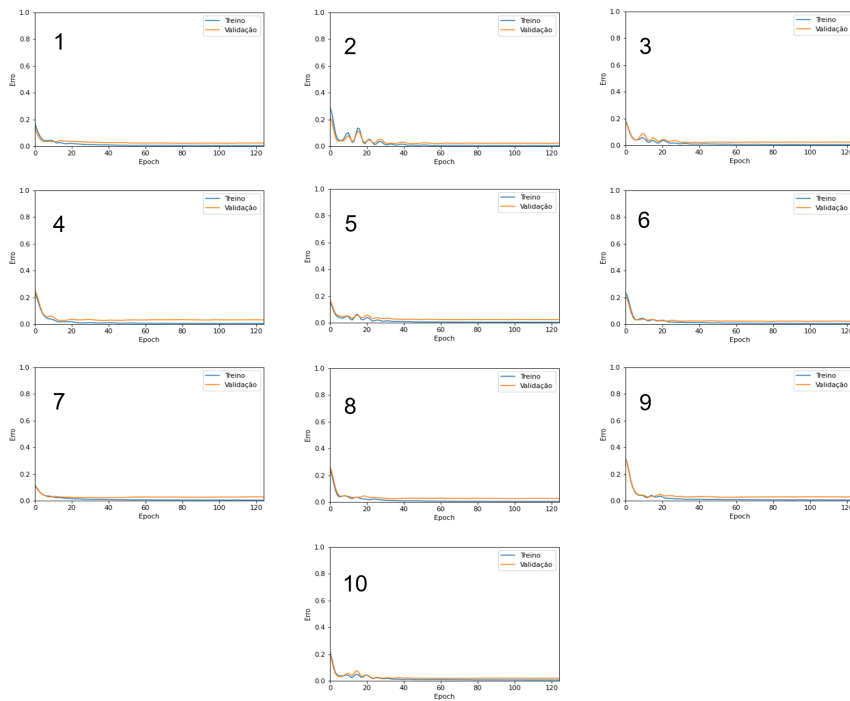


Figura B.6: Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.

RMSE nos conjuntos de dados de teste como métrica**Tabela B.16:** Ensaios conduzidos com o conjunto de dados mistos e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso das ANN.

	Parâmetros	Níveis		
		1	2	3
Ensaio 1	n_hidden	75	100	125
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	2	3	4
	loss	MSE	MAE	HUBER
	lr	0.01	0.001	0.0001
	m	0.8	0.9	1
	epoch	75	100	125
	p	0	0.25	0.5
Ensaio 2	n_hidden	85	100	115
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	3	4	5
	loss	MSE	MAE	HUBER
	lr	0.015	0.01	0.0075
	m	0.85	0.9	0.95
	epoch	100	125	150
	p	0.15	0.25	0.35
Ensaio 3	n_hidden	100	115	130
	fa_hidden	RELU	SELU	LEAKY RELU
	hidden_layer	4	5	6
	loss	MSE	MAE	HUBER
	lr	0.01	0.0075	0.005
	m	0.875	0.9	0.925
	epoch	90	100	110
	p	0.1	0.15	0.2

Tabela B.17: Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para as ANN.

	Taguchi		
	1	2	3
1	15.77282	16.9224	16.58473
2	15.51091	15.34057	16.57738
3	6.752948	17.01209	16.46945
4	16.9415	16.25734	16.72341
5	16.69315	17.17839	16.80189
6	10.26638	16.91589	16.84717
7	14.01194	16.74318	16.1239
8	15.00661	15.6642	16.60088

Continua na página seguinte

Tabela B.17 – continuação da página anterior

	Taguchi		
	1	2	3
9	11.41958	16.72399	16.4518
10	14.70544	16.98534	16.54252
11	8.670447	15.14427	16.40222
12	8.382349	16.35669	15.50468
13	17.0096	17.11436	16.90562
14	8.399491	17.23235	16.96964
15	7.333294	17.16767	16.96909
16	16.95816	16.63874	16.74739
17	8.179482	15.942	16.42055
18	10.50593	16.7158	16.69162
19	10.203	16.26829	16.02528
20	14.5174	15.7972	16.40129
21	9.072944	16.67434	16.35502
22	8.498832	17.22731	17.06907
23	11.94856	15.65414	17.15037
24	7.039512	17.25111	17.06697
25	10.62339	16.89203	17.00637
26	15.44891	16.6051	16.55971
27	12.01561	17.04886	16.74162

Tabela B.18: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.

	Nível	Hiperparâmetro							
		n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p
Ensaio 1	1	13.59732	11.50981	11.2408	12.09633	13.8583	12.76252	11.12264	11.5506
	2	11.12713	11.57004	12.63858	12.59311	12.70833	13.77911	11.4175	12.78776
	3	11.04091	12.68551	11.88599	11.07592	9.198727	9.223728	13.22522	11.42699
Ensaio 2	1	16.52867	16.27791	16.52268	16.81505	16.78322	16.46884	16.57954	16.63578
	2	16.58858	16.88873	16.59821	16.48755	16.06202	16.6551	16.53455	16.37441
	3	16.60204	16.55266	16.5984	16.41669	16.87405	16.59536	16.60521	16.70911
Ensaio 3	1	16.57562	16.31806	16.75306	16.75373	16.63648	16.5232	16.59867	16.6241
	2	16.57259	16.9448	16.56995	16.55707	16.65377	16.70437	16.58292	16.66267
	3	16.70841	16.59376	16.53361	16.54582	16.56638	16.62906	16.67504	16.56986

Tabela B.19: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.

	Hiperparâmetro									S/N
	n_hidden	fa_hidden	hidden_layer	loss	lr	m	epoch	p		
Níveis	Ensaio 1	1	3	2	2	1	2	3	2	16.76517
	Ensaio 2	3	2	3	1	3	2	3	3	17.17093
	Ensaio 3	3	2	1	1	2	2	3	2	17.12198

Tabela B.20: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para as ANN.

	Conjuntos de treino e teste										Médias	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.9338	0.9364	0.9333	0.9344	0.9354	0.9345	0.9348	0.9344	0.9352	0.9327	0.9345	0.001
Exatidão teste	0.9297	0.9205	0.933	0.9283	0.9268	0.93	0.9278	0.93	0.9298	0.9329	0.9288	0.0033
RMSE treino	0.1309	0.1289	0.1305	0.1271	0.1272	0.1288	0.1287	0.1312	0.1292	0.1339	0.1297	0.0019
RMSE teste	0.1303	0.141	0.1353	0.1479	0.1412	0.1341	0.136	0.1348	0.1396	0.1311	0.1371	0.0051
RMSE médio	0.1306	0.135	0.1329	0.1375	0.1342	0.1314	0.1324	0.133	0.1344	0.1325	0.1334	0.0019

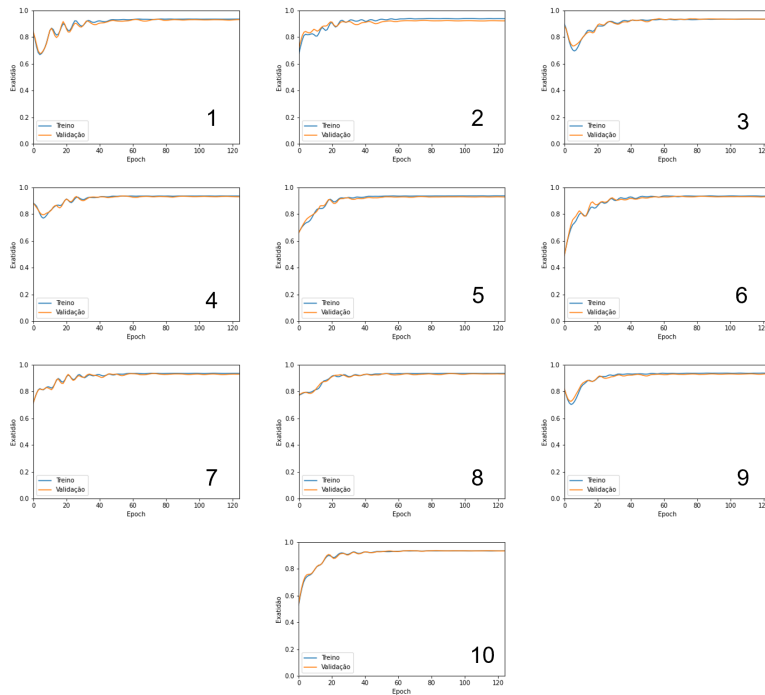


Figura B.7: Evolução da exatidão das previsões do algoritmo, com o desenrolar do processo de treino.

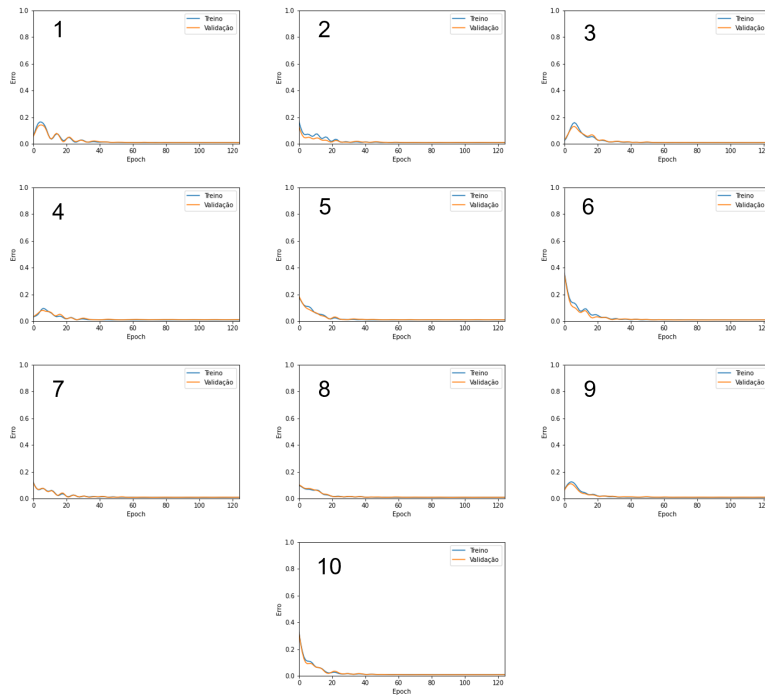


Figura B.8: Evolução do erro das previsões do algoritmo, com o desenrolar do processo de treino.

B.2 Ensaios XGBoost

B.2.1 Conjunto de dados reais

Média entre o RMSE nos conjuntos de dados de treino e teste como métrica

Tabela B.21: Ensaios conduzidos com o conjunto de dados reais e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso do XGBoost.

Parâmetros	Níveis		
	1	2	3
n_dt	75	100	125
prof_max	2	4	6
folha_max	2	4	6
lr	0.1	0.3	0.5
div_min	0	2	4
loss	squared error	squared log error	logistic
peso_min	0	1	4
lambda	0	1	3
Ensaio 2	50	75	85
prof_max	5	6	7
folha_max	3	4	5
lr	0.2	0.3	0.4
Ensaio 2	Continua na página seguinte		

Tabela B.21 – continuação da página anterior

Parâmetros	Níveis		
	1	2	3
div_min	1	2	3
loss	squared error	squared log error	logistic
peso_min	0	1	2
lambda	0.5	1	2

Tabela B.22: Valores de S/N associados aos ensaios com os conjuntos de dados reais e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para o XGBoost.

	Taguchi		
	1	2	
	1	16.0892	12.19562
	2	12.17375	12.17362
	3	12.19673	12.19569
	4	17.31245	12.19562
	5	12.17361	12.17361
	6	12.19562	12.19562
	7	17.29131	12.19562
	8	12.17361	12.17361
	9	12.19562	12.19562
	10	13.07328	12.17361
	11	12.25628	12.23645
	12	12.19562	12.19562
	13	12.77942	12.17361
Experiências	14	12.19585	12.19562
	15	12.19562	12.19562
	16	12.97174	12.17361
	17	12.56933	12.39119
	18	12.19562	12.19562
	19	14.62818	13.09163
	20	12.19562	12.19562
	21	12.17361	12.17361
	22	14.78611	12.86875
	23	12.19562	12.19562
	24	12.17361	12.17361
	25	14.37413	12.47178
	26	12.19562	12.19562
	27	12.17361	12.17361

Tabela B.23: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.

Níveis	Hiperparâmetro								
	n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda	
Ensaio 1	1	13.75577	12.99803	13.03908	12.93044	14.81176	13.76297	13.80392	13.03268
	2	12.49253	13.11199	13.10558	13.15731	12.23659	12.42958	12.98847	13.11872
	3	12.98846	13.12673	13.09209	13.14901	12.18841	13.04421	12.44436	13.08536
Ensaio 2	1	12.18829	12.29238	12.28481	12.21898	12.39332	12.19562	12.21455	12.30957
	2	12.21455	12.26308	12.2235	12.30957	12.21455	12.17361	12.39332	12.26761
	3	12.39332	12.2407	12.28784	12.26761	12.18829	12.42693	12.18829	12.21898

Tabela B.24: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.

Níveis		Hiperparâmetro								S/N
		n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda	
Ensaio 1	Ensaio 1	0	2	1	1	0	0	0	1	17.44336
	Ensaio 2	2	0	2	1	0	2	1	0	12.17528

Tabela B.25: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, média entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.

	Conjuntos de treino e teste										Média	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	1	1	1	1	1	1	1	1	1	1	1	0
Exatidão teste	0.84	0.87	0.85	0.89	0.86	0.87	0.87	0.82	0.9	0.85	0.862	0.0223
RMSE treino	0.0012	0.001	0.0011	0.0008	0.001	0.001	0.001	0.0013	0.0013	0.001	0.0011	0.0001
RMSE teste	0.351	0.247	0.2825	0.2288	0.2653	0.262	0.238	0.288	0.2011	0.282	0.2646	0.0387
RMSE médio	0.1761	0.124	0.1418	0.1148	0.1332	0.1315	0.1195	0.1447	0.1012	0.1415	0.1328	0.0194

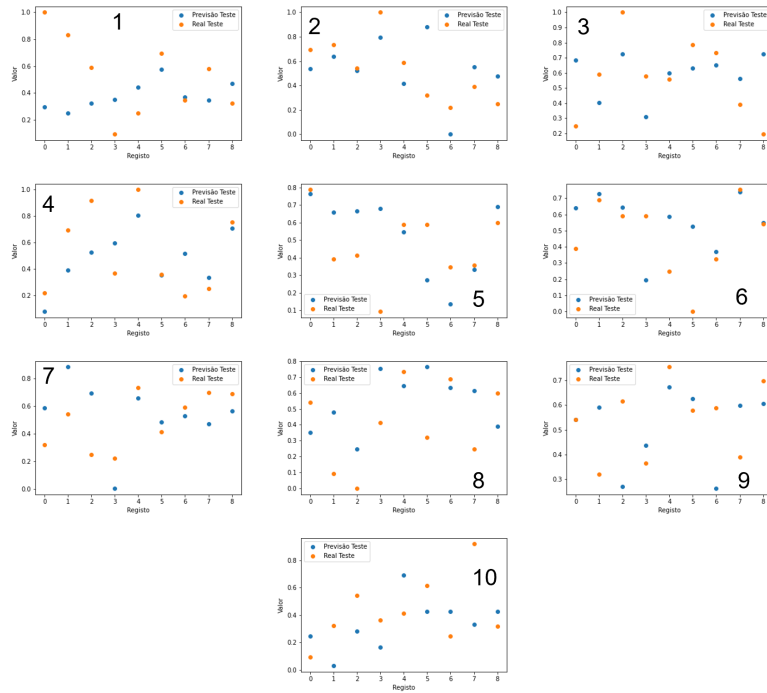


Figura B.9: Comparação das previsões e valores reais para os conjuntos de dados de teste.

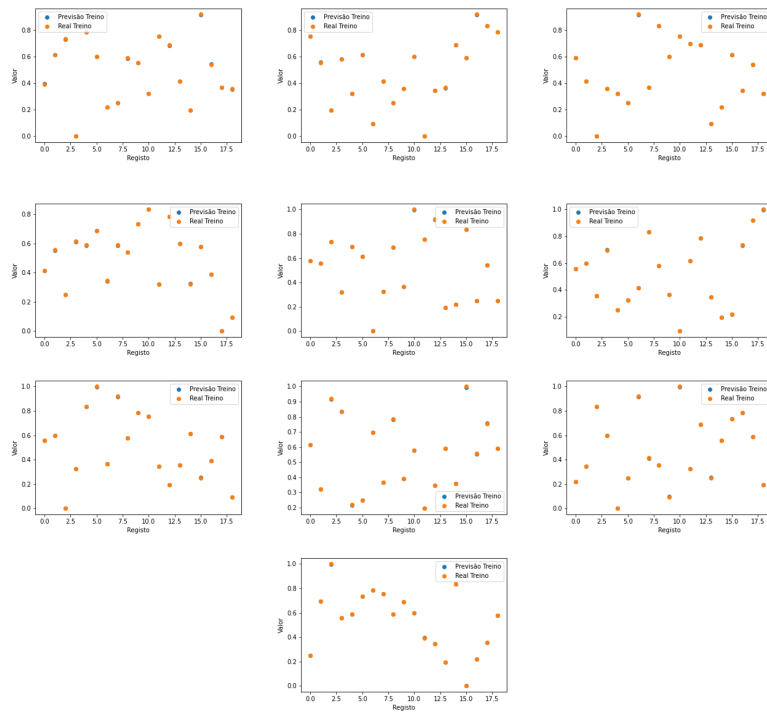


Figura B.10: Comparação das previsões e valores reais para os conjuntos de dados de treino.

RMSE nos conjuntos de dados de teste como métrica

Tabela B.26: Ensaios conduzidos com o conjunto de dados reais e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso do XGBoost.

	Parâmetros	Níveis			
		1	2	3	
Ensaio 1	n_dt	75	100	125	
	prof_max	2	4	6	
	folha_max	2	4	6	
	lr	0.1	0.3	0.5	
	div_min	0	2	4	
	loss	squared error	squared log error	logistic	
	peso_min	0	1	4	
	lambda	0	1	3	
	Ensaio 2	n_dt	85	100	115
		prof_max	2	3	4
folha_max		3	4	5	
lr		0.4	0.5	0.6	
div_min		0	1	2	
loss		squared error	squared log error	logistic	
peso_min		2	4	6	
lambda		0	0.5	1	
Ensaio 3		n_dt	95	100	105
		prof_max	3	4	5
	folha_max	2	3	4	
	lr	0.45	0.5	0.55	
	div_min	0	0.5	1	
	loss	squared error	squared log error	logistic	
	peso_min	1	2	3	
	lambda	0	0.25	0.5	
	Ensaio 4	n_dt	100	105	110
		prof_max	3	4	5
folha_max		2	3	4	
lr		0.475	0.5	0.525	
div_min		0	0.25	0.5	
loss		squared error	squared log error	logistic	
peso_min		1.5	2	2.5	
lambda		0.125	0.25	0.375	
Ensaio 5		n_dt	110	115	120
		prof_max	4	5	6
	folha_max	2	3	4	
	lr	0.49	0.5	0.51	
	div_min	0	0.125	0.25	
	loss	squared error	squared log error	logistic	
	peso_min	1.75	2	2.25	
	lambda	0.2	0.25	0.3	
	Ensaio 5	n_dt	115	120	125
		prof_max	4	5	6
folha_max		2	3	4	
Ensaio 6	Continua na página seguinte				

Tabela B.26 – continuação da página anterior

Parâmetros	Níveis		
	1	2	3
lr	0.48	0.49	0.5
div_min	0.1	0.125	0.15
loss	squared error	squared log error	logistic
peso_min	1.9	2	2.1
lambda	0.225	0.25	0.275

Tabela B.27: Valores de S/N associados aos ensaios com os conjuntos de dados reais e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para o XGBoost.

	Taguchi						
	1	2	3	4	5	6	
1	11.72422	10.34621	10.94777	11.12739	11.05682	12.53788	
2	12.26603	12.26578	12.26578	12.32671	13.34845	13.33817	
3	12.27366	12.49483	12.27139	12.27139	13.03772	13.25126	
4	11.34078	11.2485	11.19019	11.09826	11.06805	12.57527	
5	12.26578	12.26578	12.26578	12.32213	13.26756	13.34526	
6	12.27139	12.49483	12.27139	12.27139	13.07022	13.25227	
7	11.31384	10.9414	10.95475	11.08303	11.04206	12.60711	
8	12.26578	12.26578	12.26578	12.29833	13.28512	13.27519	
9	12.27139	12.49483	12.27139	12.27139	13.07044	13.23806	
10	12.98571	12.26578	11.712	10.77916	10.27114	13.36122	
11	12.28872	13.03809	12.98209	13.16017	13.34976	13.26652	
12	12.27139	12.27139	12.27139	12.34946	12.68574	12.44187	
13	12.8092	12.26578	11.72554	10.80726	10.40978	13.33817	
Experiências	14	12.27186	12.69333	12.99217	13.11102	13.34459	13.25677
	15	12.27139	12.27139	12.27139	12.41202	12.81287	12.56777
	16	12.96917	12.26578	11.84722	10.8835	10.39711	13.34526
	17	12.42286	13.19432	12.74645	13.19621	13.34932	13.26788
	18	12.27139	12.27139	12.27139	12.40642	12.81295	12.45244
	19	10.88874	12.27139	13.29593	13.2847	13.2868	13.41475
	20	12.27139	12.27139	12.40642	12.81295	12.63069	12.66885
	21	12.26578	12.26578	12.26578	12.26578	12.47857	12.88142
	22	11.12467	12.27139	13.29747	13.2848	13.28679	13.43847
	23	12.27139	12.27139	12.34244	12.60677	12.7294	12.63422
	24	12.26578	12.26578	12.26578	12.26578	12.47463	12.88549
	25	12.33316	12.27139	13.28889	13.30565	13.29382	13.46124
	26	12.27139	12.27139	12.41697	12.81233	12.66171	12.67901
	27	12.26579	12.26578	12.26578	12.26578	12.48068	12.87946

Tabela B.28: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.

	Níveis	Hiperparâmetro							
		n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda
Ensaio 1	1	11.99921	12.13729	12.17657	12.2763	11.94328	12.0008	12.01774	12.15192
	2	12.50685	12.09914	12.2549	12.10748	12.28836	12.48434	11.99537	12.10104
	3	11.99534	12.26497	12.06993	12.11763	12.26977	12.01627	12.4883	12.24845
Ensaio 2	1	11.86866	12.16563	12.18298	12.12732	11.79418	11.79605	12.0288	12.18298
	2	12.50414	12.22757	12.26588	12.28324	12.50414	12.26578	12.26952	12.26588
	3	12.26952	12.24912	12.19345	12.23176	12.344	12.58049	12.344	12.19345
Ensaio 3	1	11.85602	12.26873	12.25063	12.27174	12.02886	11.89697	12.06786	12.24372
	2	12.31329	12.29135	12.29605	12.28451	12.52043	12.09772	12.61042	12.29734
	3	12.6495	12.25873	12.27213	12.26257	12.26952	12.82413	12.14053	12.27776
Ensaio 4	1	11.89667	12.26419	12.26322	12.27106	11.73931	12.07874	12.17483	12.27192
	2	12.1228	12.24216	12.2627	12.28237	12.73851	11.80161	12.66558	12.27042
	3	12.76717	12.28029	12.26072	12.23321	12.30882	12.9063	11.94624	12.2443
Ensaio 5	1	12.47183	12.46063	12.49924	12.49405	11.56804	12.1667	12.29383	12.47429
	2	12.15925	12.49599	12.46096	12.48459	13.1074	12.04589	13.12001	12.49529
	3	12.81368	12.48813	12.48455	12.46612	12.76931	13.23216	12.03091	12.47517
Ensaio 6	1	13.04672	13.01799	13.01678	13.03442	13.11993	12.57382	12.90642	13.02683
	2	13.0331	13.03263	13.02912	13.0226	13.08132	13.18329	13.08168	13.02171
	3	12.99366	13.02285	13.02757	13.01646	12.87223	13.31636	13.08537	13.02493

Tabela B.29: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.

	Hiperparâmetro									S/N
	n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda		
Ensaio 1	2	3	2	1	2	2	3	3	12.26283	
Ensaio 2	2	3	2	2	2	3	3	2	12.48728	
Ensaio 3	3	2	2	2	2	3	2	2	13.15896	
Ensaio 4	3	3	1	2	2	3	2	1	13.42206	
Ensaio 5	3	2	1	1	2	3	2	2	13.45061	
Ensaio 6	1	2	2	1	1	3	3	1	13.31427	

Tabela B.30: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados reais, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.

	Conjuntos de treino e teste										Médias	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.9047	0.9077	0.8689	0.9104	0.8829	0.9049	0.8775	0.9026	0.9159	0.9008	0.8976	0.0148
Exatidão teste	0.8872	0.901	0.8731	0.8966	0.9027	0.8918	0.9131	0.8679	0.9049	0.9011	0.8939	0.0135
RMSE treino	0.1972	0.2018	0.239	0.1733	0.2251	0.2015	0.2368	0.1963	0.1957	0.1846	0.2051	0.0206
RMSE teste	0.2174	0.1935	0.2653	0.2524	0.1785	0.1911	0.1502	0.2238	0.1959	0.2292	0.2097	0.033
RMSE médio	0.2073	0.1977	0.2522	0.2129	0.2018	0.1963	0.1935	0.21	0.1958	0.2069	0.2074	0.0162

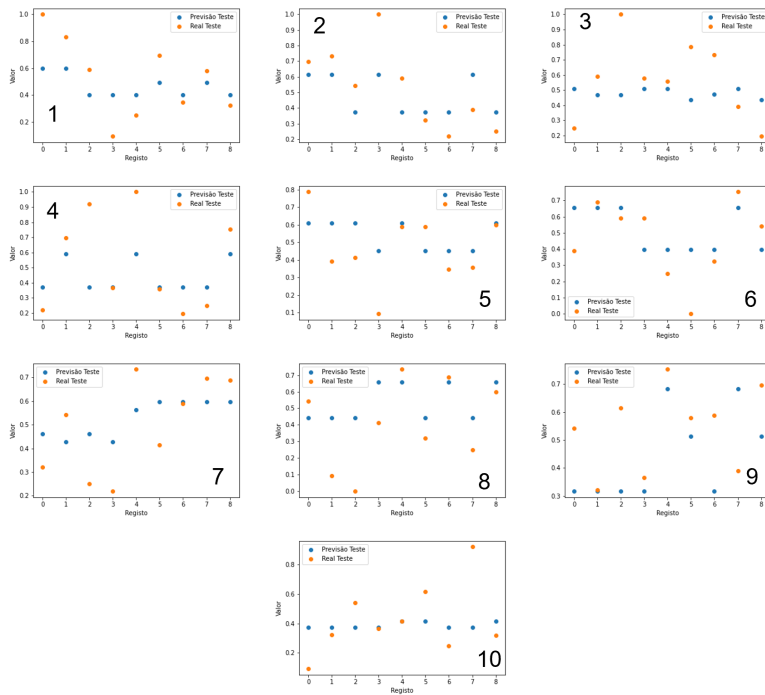


Figura B.11: Comparação das previsões e valores reais para os conjuntos de dados de teste.

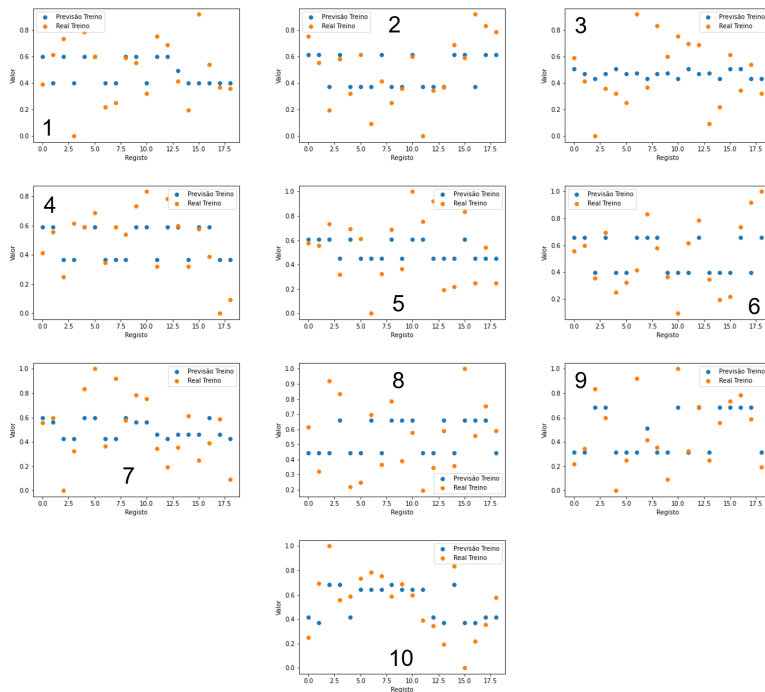


Figura B.12: Comparação das previsões e valores reais para os conjuntos de dados de treino.

B.2.2 Conjunto de dados mistos

Média entre o RMSE nos conjuntos de dados de treino e teste como métrica

Tabela B.31: Ensaios conduzidos com o conjunto de dados mistos e com a média entre os RMSE nos conjuntos de dados de treino e teste como métrica para cálculo do S/N, no caso do XGBoost.

	Parâmetros	Níveis		
		1	2	3
Ensaio 1	n_dt	75	100	125
	prof_max	2	4	6
	folha_max	2	4	6
	lr	0.1	0.3	0.5
	div_min	0	2	4
	loss	squared error	squared log error	logistic
	peso_min	0	1	4
	lambda	0	1	3
	Ensaio 2	n_dt	65	75
prof_max		4	6	8
folha_max		4	6	8
lr		0.4	0.5	0.6
div_min		0	0.5	1
loss		squared error	squared log error	logistic
peso_min		0	0.5	1
lambda		2	3	4
Ensaio 3		n_dt	60	65
	prof_max	6	8	10
	folha_max	6	8	10
	lr	0.5	0.6	0.7
	div_min	0	0.25	0.5
	loss	squared error	squared log error	logistic
	peso_min	0	0.25	0.5
	lambda	3	4	5

Tabela B.32: Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à média entre os RMSE nos conjuntos de dados de treino e teste como métrica para o seu cálculo, para o XGBoost.

	Taguchi			
	1	2	3	
Experiências	1	18.4409	21.27216	21.62981
	2	14.03098	15.54804	16.28809
	3	15.11415	16.66736	17.08335
	4	21.27645	21.62981	21.71244
	5	14.03093	15.47543	16.23658
	6	15.65419	16.82499	17.27022
	7	21.62981	21.71244	21.59153

Continua na página seguinte

Tabela B.32 – continuação da página anterior

	Taguchi		
	1	2	3
8	14.03093	15.55093	16.18731
9	15.76557	16.82603	17.23037
10	19.14702	21.28549	21.29786
11	16.37611	17.22763	17.51649
12	14.02696	15.75169	16.38682
13	19.10547	21.38075	21.59331
14	15.99128	17.10172	17.48764
15	14.02695	15.7328	16.34858
16	20.49792	21.45142	21.49549
17	16.48352	17.25532	17.63756
18	14.02696	15.78858	16.51823
19	19.43831	20.63152	21.22023
20	14.57022	16.34858	16.88148
21	14.03093	15.02445	15.48038
22	20.73584	21.15145	21.23235
23	14.82199	16.51823	16.79484
24	14.03093	15.36038	15.65185
25	19.72873	21.0721	20.98258
26	14.27787	16.44616	16.85161
27	14.03093	14.88401	15.47543

Tabela B.33: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.

Níveis	Hiperparâmetro								
	n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda	
Ensaio 1	1	16.66377	16.13062	16.4648	16.08303	20.00005	16.34423	16.92121	16.17007
	2	16.63135	16.63045	16.50547	16.66771	14.95709	15.88178	16.00851	16.65761
	3	16.18509	16.71914	16.50994	16.72946	14.52306	17.25419	16.55049	16.65252
Ensaio 2	1	17.94524	17.75077	17.89033	17.78946	21.28746	17.91116	17.94088	17.81775
	2	18.10838	17.9084	17.84415	17.82557	16.38578	17.32899	17.41139	17.86499
	3	17.49299	17.88744	17.81214	17.93159	15.87337	18.30646	18.19434	17.86387
Ensaio 3	1	18.35886	18.19828	18.25907	18.19338	21.41729	18.3017	18.24257	18.23278
	2	18.47578	18.25864	18.19222	18.27251	16.87573	17.74514	17.93342	18.27202
	3	17.84119	18.2189	18.22454	18.20994	16.3828	18.62898	18.49984	18.17102

Tabela B.34: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.

	Hiperparâmetro								S/N
	n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda	
Ensaio 1	1	3	3	3	1	3	1	2	21.01767
Ensaio 2	2	2	1	3	1	3	3	2	20.88837
Ensaio 3	2	2	1	2	1	3	3	2	21.12357

Tabela B.35: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, médio entre o RMSE nos conjuntos de treino e teste como métrica para cálculo do S/N e para o XGBoost.

	Conjuntos de treino e teste										Médias	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.9997	0.9997	0.9997	0.9997	0.9997	0.9996	0.9997	0.9996	0.9996	0.9997	0.9997	0
Exatidão teste	0.9165	0.918	0.9148	0.9161	0.9143	0.9132	0.9151	0.9232	0.9118	0.917	0.916	0.0029
RMSE treino	0.0008	0.0009	0.0008	0.0008	0.0008	0.0009	0.0008	0.0009	0.0009	0.0008	0.0008	0
RMSE teste	0.1586	0.1532	0.1645	0.1736	0.1661	0.165	0.1636	0.1573	0.1729	0.1575	0.1632	0.0064
RMSE médio	0.0797	0.0771	0.0826	0.0872	0.0834	0.083	0.0822	0.0791	0.0869	0.0792	0.082	0.0032

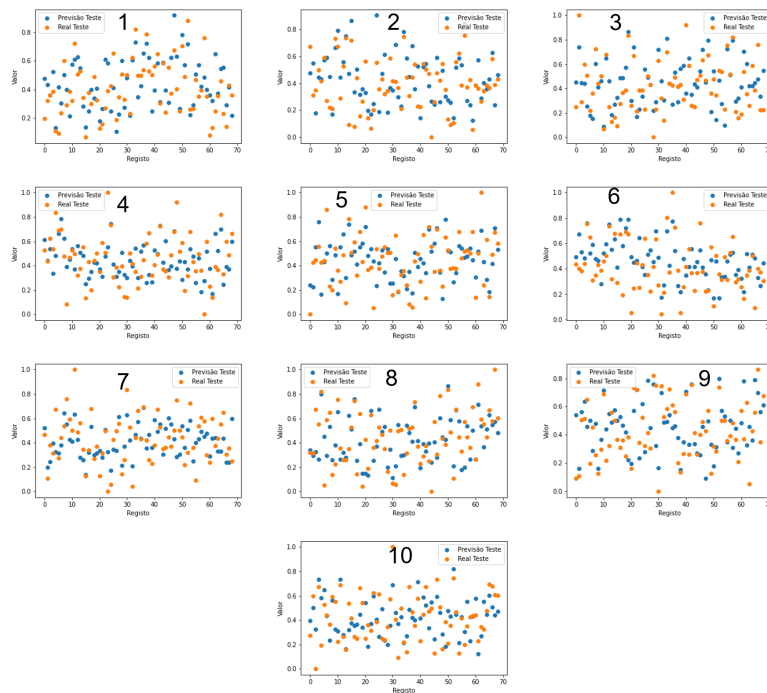


Figura B.13: Comparação das previsões e valores reais para os conjuntos de dados de teste.

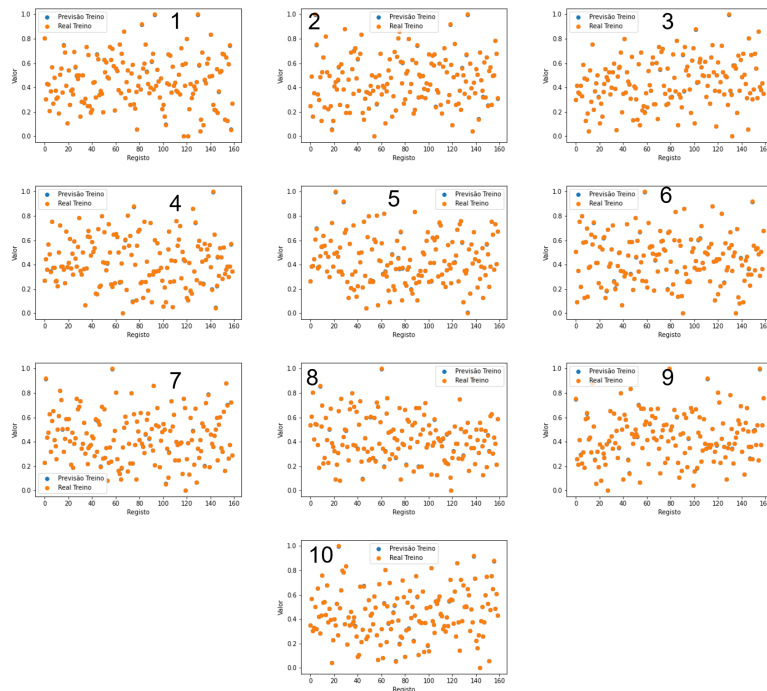


Figura B.14: Comparação das previsões e valores reais para os conjuntos de dados de treino.

RMSE nos conjuntos de dados de teste como métrica

Tabela B.36: Ensaios conduzidos com o conjunto de dados mistos e com o RMSE nos conjuntos de dados de teste como métrica para cálculo do S/N, no caso do XGBoost.

	Parâmetros	Níveis		
		1	2	3
Ensaio 1	n_dt	75	100	125
	prof_max	2	4	6
	folha_max	2	4	6
	lr	0.1	0.3	0.5
	div_min	0	2	4
	loss	squared error	squared log error	logistic
	peso_min	0	1	4
	lambda	0	1	3
Ensaio 2	n_dt	85	100	115
	prof_max	2	3	4
	folha_max	4	6	8
	lr	0.05	0.1	0.2
	div_min	0	1	2
Ensaio 2	loss	squared error	squared log error	logistic
	peso_min	0	0.5	1
	lambda	0	0.5	1

Tabela B.37: Valores de S/N associados aos ensaios com os conjuntos de dados mistos e à ao RMSE nos conjuntos de dados de teste como métrica para o seu cálculo, para o XGBoost.

	Taguchi	
	1	2
	1	16.67258
2	13.87394	14.19796
3	14.84868	15.78183
4	15.67768	16.2559
5	13.87394	14.23591
6	15.35693	15.89373
7	15.66811	15.89806
8	13.87394	14.41771
9	15.40734	15.90692
10	15.43822	16.15859
11	16.01342	16.41231
12	13.8484	14.16966
13	16.30886	16.63782
Experiências 14	15.67655	16.30335
15	13.8484	14.03708
16	15.54368	16.19318
17	16.15627	16.42405
18	13.8484	14.06258
19	15.76497	16.48677
20	14.30545	15.1009
21	13.87394	13.87394
22	15.19665	15.76207
23	14.49807	15.22322
24	13.87394	13.87394
25	16.17853	16.44902
26	14.06415	15.04594
27	13.87394	13.87395

Tabela B.38: Médias dos valores de S/N para cada nível de cada hiperparâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.

Níveis	Hiperparâmetro								
	n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda	
Ensaio 1	1	15.02813	14.95996	14.9458	15.0384	15.8277	14.71458	15.27627	15.00549
	2	15.18691	14.92345	14.92503	14.93348	14.70397	14.50382	14.47858	14.94508
	3	14.62552	14.95715	14.96973	14.86868	14.30889	15.62215	15.08571	14.88999
Ensaio 2	1	15.50205	15.45694	15.38325	15.47305	16.30799	15.19153	15.53844	15.47425
	2	15.59985	15.35811	15.38834	15.39188	15.26237	14.82922	14.86875	15.35671
	3	15.07664	15.36349	15.40695	15.31361	14.60818	16.15778	15.77135	15.34758

Tabela B.39: Valores de S/N dos ensaios conduzidos com os níveis associados ao máximo S/N para cada parâmetro, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.

	Hiperparâmetro									S/N
	n_dt	prof_max	folha_max	lr	div_min	loss	peso_min	lambda		
Ensaio 1	2	1	3	1	1	3	1	1	17.01363	
Ensaio 2	2	1	3	1	1	3	3	1	16.95038	

Tabela B.40: Valores de exatidão e RMSE do melhor algoritmo, com o conjunto de dados mistos, RMSE nos conjuntos de teste como métrica para cálculo do S/N e para o XGBoost.

	Conjuntos de treino e teste										Média	STD
	1	2	3	4	5	6	7	8	9	10		
Exatidão treino	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.94	0.95	0.94	0.941	0.003
Exatidão teste	0.93	0.93	0.93	0.93	0.93	0.92	0.92	0.93	0.92	0.93	0.927	0.0046
RMSE treino	0.114	0.112	0.113	0.108	0.11	0.112	0.112	0.117	0.107	0.117	0.1122	0.0032
RMSE teste	0.135	0.137	0.136	0.151	0.143	0.146	0.147	0.132	0.146	0.136	0.1409	0.0061
RMSE médio	0.1245	0.1245	0.1245	0.1295	0.1265	0.129	0.1295	0.1245	0.1265	0.1265	0.1266	0.002

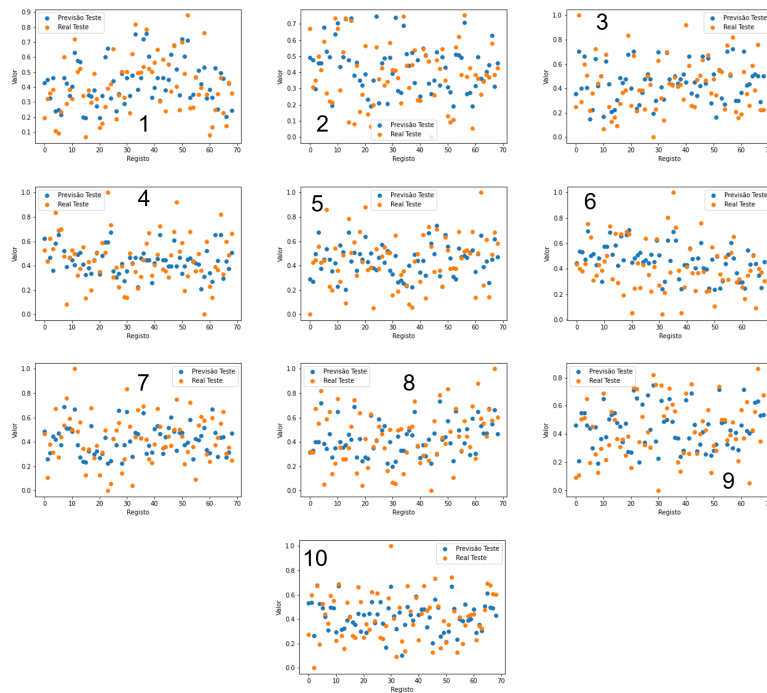


Figura B.15: Comparação das previsões e valores reais para os conjuntos de dados de teste.

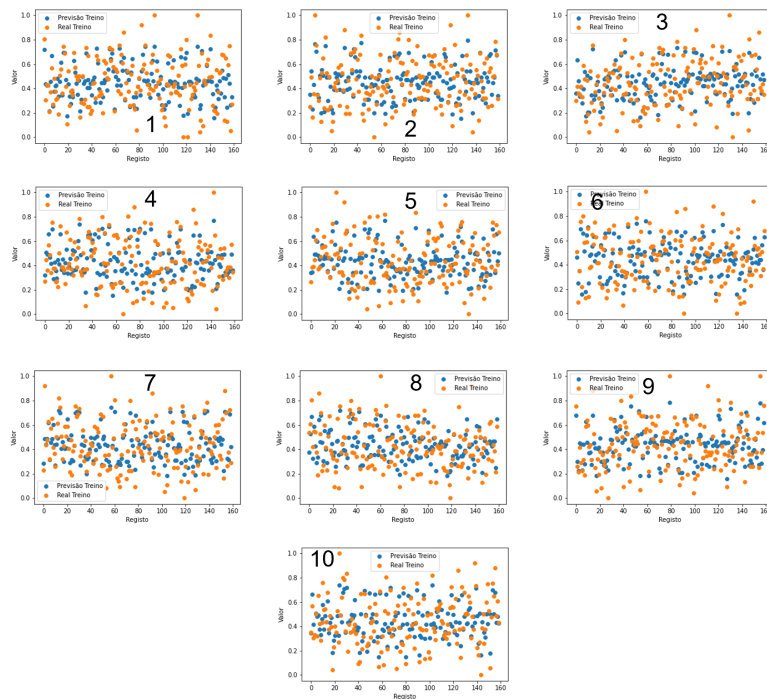


Figura B.16: Comparação das previsões e valores reais para os conjuntos de dados de treino.

Apêndice C

Anexo 3 - Tabela de dados mistos

Tabela C.1: Tabela de dados mistos normalizados.

Registro	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
0	0	0	0	0.166667	0.833729
1	0	0.5	0.5	0.583333	0.752969
2	0	1	1	1	0.733967
3	0.5	0.5	0.5	1	0.541568
4	0.5	1	1	0.166667	0.615202
5	0.5	0	0	0.583333	0.194774
6	1	0	1	0.583333	0.092637
7	1	0.5	0	1	0
8	1	1	0.5	0.166667	0.356295
9	0	1	0.5	0.583333	0.786223
10	0	0	1	1	0.598575
11	0	0.5	0	0.166667	1
12	0.5	1	0	1	0.365796
13	0.5	0	0.5	0.166667	0.589074
14	0.5	0.5	1	0.583333	0.91924
15	1	0.5	1	0.166667	0.344418
16	1	1	0	0.583333	0.32304
17	1	0	0.5	1	0.218527
18	0	0.25	0.5	0.333333	0.688836
19	0	0.5	0.75	0.166667	0.555819
20	0	0.75	1	0	0.589074
21	0.25	0.25	0.75	0	0.579572
22	0.25	0.5	1	0.333333	0.695962
23	0.25	0.75	0.5	0.166667	0.413302
24	0.5	0.25	1	0.166667	0.320665
25	0.5	0.5	0.5	0	0.249406
26	0.5	0.75	1	0.333333	0.249406
27	0	0.25	1	0.333333	0.389549
28	0	140	1550	2910	650
29	0.291075	0.22268	0.39228	0.3361	0.524996
30	0.549341	0.489412	0.657182	0.467879	0.345795
31	0.072392	0.343445	0.357198	0.144139	0.673228

Continua na página seguinte

Tabela C.1 – continuação da página anterior

Registro	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
32	0	0.455692	0.714215	0.716164	0.565442
33	0.654231	0.338369	0.489745	0.384496	0.185559
34	0.530807	0.382137	0.288619	0.89967	0.408582
35	0.451646	0.773387	0.24546	0.692075	0.664357
36	0.393994	0.417509	0.153709	0.807291	0.596005
37	0.399864	0.578017	0.582261	0.290877	0.502806
38	0.585225	0.570409	0.214842	0.381812	0.483268
39	0.515788	0.56615	0.46388	0.599207	0.495127
40	0.464552	0.719123	0.338725	0.552574	0.383513
41	0.638701	0.247894	0.443323	0.72238	0.264117
42	0.318169	0.335541	0.82401	0.170488	0.677855
43	0.55249	0.710307	0.456557	0.603983	0.504174
44	0.094696	0.353641	0.567372	0.295253	0.878849
45	0.899115	0.203237	0.758872	0.532938	0.066448
46	0.47953	0.345303	0.333428	0.703334	0.288794
47	0.698204	0.508332	0.190826	0.855354	0.266674
48	0.558999	0.107994	0.49071	0.369084	0.131188
49	0.35956	0.521637	0.593903	0.567809	0.47496
50	0.712797	0.904426	0.959244	0.3817	0.375735
51	0.507068	0.515743	0.644367	0.667885	0.190918
52	0.427111	0.571193	0.317428	0.531634	0.487396
53	1	0.368427	0.432337	0.401821	0.128767
54	0.396381	0.404586	0.584783	0.529126	0.493345
55	0.454023	0.386917	0.446449	0.742334	0.090447
56	0.399279	0.897669	0.370696	0.502682	0.349153
57	0.505092	0.214394	0.858851	0.715612	0.385168
58	0.356517	0.350916	0.45525	0.605373	0.375732
59	0.262433	0.768743	0.569012	0.087889	0.371704
60	0.42122	0.750845	0.353696	0.342984	0.583754
61	0.413266	0.613203	0.077957	0.731623	0.443251
62	0.462108	0.577747	0.289214	0.492045	0.228898
63	0.636338	0.65728	0.544256	0.853886	0.435693
64	0.185108	0.304666	0.634886	0.47986	0.803157
65	0.562001	0.588949	0.562982	0.563282	0.415903
66	0.554528	0.569523	0.704729	0.747488	0.079148
67	0.574881	0.457501	0.380314	0.321146	0.263524
68	0.412701	0.168899	0.234979	0.666536	0.19472
69	0.345485	0.113064	0.646797	0.758874	0.605195
70	0.160183	0.713466	0.676667	0.468761	0.689654
71	0.198498	0.501652	0.433884	0.442694	0.752245
72	0.38965	0.4575	0.130158	0.154725	0.248004
73	0.834417	0.53396	0.053995	0.852693	0
74	0.352108	0.470787	0.527535	0.221448	0.551939
75	0.707795	0.352657	0.844972	0.380074	0.042194
76	0.607498	0.606988	0.552138	0.867723	0.563534
77	0.764129	0.845802	0.286632	0.82194	0.538053
78	0.534115	0.265657	0.28522	0.499824	0.220229
79	0.667676	0.622788	0.533983	0.595262	0.28596

Continua na página seguinte

Tabela C.1 – continuação da página anterior

Registro	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
80	0.075362	0.477822	0.250282	0.42376	0.701205
81	0.31528	0.753505	0.902556	0.100939	0.514517
82	0.38225	0.839579	0.622441	0.367376	0.460112
83	0.034478	0.667731	0.192592	0.570223	0.819801
84	0.251968	0.715553	0.634935	0.319217	0.558483
85	0.518827	0.228381	0.855683	0.275696	0.421783
86	0.323007	0.397454	0.459258	0.305992	0.569719
87	0.435233	0.641881	0.477276	0.550974	0.383981
88	0.062653	0.706621	0.678421	0.243599	0.719141
89	0.332641	0.039911	0.352353	0.474928	0.543585
90	0.420017	0.875978	0.495802	0.325969	0.584933
91	0.816649	0.341886	0.157187	0.532459	0.342866
92	0.307607	0.898091	0.374354	0.802792	0.546578
93	0.455099	0.90756	0.60082	0.428093	0.34762
94	0.327967	0.422132	0.616517	0.554374	0.609125
95	0.444324	0.701806	0.851009	0.526943	0.308237
96	0.432511	0.272171	0.404147	0.475564	0.393943
97	0.472761	0.694135	0.651094	0.407617	0.627576
98	0.559906	0.448978	0.528023	0.458225	0.13718
99	0.425965	0.536527	0.403367	0.498883	0.670343
100	0.704324	0.429311	0.707217	0.354942	0.226418
101	0.402244	0.562041	0.388941	0.604942	0.503669
102	0.484402	0.597046	0.144294	0.686328	0.498287
103	0.47262	0.263424	0.320753	0.457008	0.507969
104	0.295125	0.675142	0.625379	0.517394	0.671794
105	0.688388	0.660713	0.38544	0.735273	0.357777
106	0.041051	0.252976	0.612387	0.818617	0.535511
107	0.304627	0.539429	0.836464	0.642923	0.597982
108	0.548208	0.495683	0.697773	0.494099	0.257085
109	0.545393	0.673791	0.382201	0.596033	0.52913
110	0.545956	1	0.375435	0.523448	0.46373
111	0.485667	0.674816	1	0.173	0.403146
112	0.202757	0.340802	0.666348	0.50425	0.632822
113	0.385458	0.54046	0.638099	0.429581	0.417107
114	0.456913	0.356833	0.425934	0.68455	0.425736
115	0.648831	0.535086	0.104248	0.755058	0.329277
116	0.49177	0.490708	0.715045	0.477443	0.305102
117	0.322691	0.670641	0.718443	0.755976	0.477551
118	0.349395	0.848524	0.103354	0.811901	0.760348
119	0.685221	0.539512	0.456795	0.400708	0.092617
120	0.941419	0.544937	0.586867	0.510729	0.162164
121	0.360636	0.23187	0.309244	0.539688	0.672274
122	0.444903	0.574346	0.610866	0.681765	0.433963
123	0.29551	0.812589	0.826363	0.718235	0.524991
124	0.654721	0.522635	0.387037	0.494539	0.320833
125	0.598707	0.425063	0.828432	0.287146	0.311871
126	0.56744	0.273394	0.593755	0.639696	0.386965
127	0.367387	0.772886	0.613641	0.439031	0.558288

Continua na página seguinte

Tabela C.1 – continuação da página anterior

Registro	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
128	0.425394	0.515338	0.251083	0.689275	0.621008
129	0.878274	0.576672	0.54571	0.988558	0.107417
130	0.687493	0.3742	0.353018	0.48159	0.359271
131	0.593377	0.640197	0.455652	0.432264	0.7494
132	0.513793	0.528776	0.585191	0.472629	0.488602
133	0.643761	0.325583	0.388051	0.92447	0.210266
134	0.663702	0.25369	0.61524	0.486652	0.107871
135	0.622032	0	0.445343	0.604685	0.268506
136	0.670448	0.392706	0.341846	0.434217	0.211954
137	0.343223	0.418952	0.503976	0.621717	0.379124
138	0.418639	0.520406	0.645462	0.35889	0.528025
139	0.648465	0.639123	0.344929	0.905589	0.053043
140	0.452399	0.794032	0.526517	0.614326	0.501465
141	0.459168	0.820576	0.135366	0.444238	0.313448
142	0.53901	0.232249	0.860813	0.68957	0.228206
143	0.570986	0.39357	0.71699	0.128914	0.303617
144	0.653752	0.463526	0.36297	0.900796	0.25783
145	0.714033	0.585173	0.177012	0.598498	0.364102
146	0.262134	0.746419	0.530799	0.676511	0.861618
147	0.580692	0.463306	0.627753	0.437532	0.248402
148	0.433977	0.691346	0.274682	0.660091	0.648996
149	0.542297	0.744347	0.378114	0.525622	0.368262
150	0.474693	0.089023	0.704457	0.24446	0.3214
151	0.713712	0.219848	0.332696	0.499985	0.126296
152	0.413921	0.396224	0.377099	0.652245	0.65378
153	0.520965	0.585721	0.444814	0.521931	0.372601
154	0.316397	0.82218	0.644328	0.516602	0.677613
155	0.479571	0.458135	0.393816	0.420735	0.428358
156	0.615425	0.126872	0.667424	0.555249	0.357598
157	0.552711	0.381709	0.592416	0.605539	0.156448
158	0.562481	0.726817	0.519173	0.309573	0.35267
159	0.40234	0.251292	0.814898	0.696708	0.431614
160	0.669365	0.574798	0.526897	0.454288	0.13678
161	0.261553	0.38803	0.522794	0.715654	0.680681
162	0.657871	0.690736	0	0.607678	0.470733
163	0.597932	0.658477	0.620098	0.701882	0.352482
164	0.786686	0.592742	0.210496	0.458014	0.367392
165	0.383388	0.48902	0.718951	0.546324	0.46531
166	0.313546	0.599567	0.374064	0.477683	0.266961
167	0.779401	0.475845	0.358431	0.509822	0.372585
168	0.332877	0.698673	0.648195	0.472242	0.445946
169	0.747696	0.724813	0.82427	0.279191	0.189538
170	0.815308	0.744249	0.39245	0.440977	0.222708
171	0.425759	0.000642	0.241935	0.739024	0.571658
172	0.512088	0.504418	0.655839	0.533292	0.488458
173	0.354807	0.342024	0.338318	0.215551	0.320926
174	0.49106	0.54443	0.437001	0.206131	0.414247
175	0.551862	0.448282	0.0781	0.453207	0.259957

Continua na página seguinte

Tabela C.1 – continuação da página anterior

Registro	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
176	0.700678	0.385358	0.489231	0.232373	0.162644
177	0.5065	0.398998	0.352804	0.273123	0.502617
178	0.634606	0.097603	0.133578	0.523095	0.385335
179	0.663979	0.10645	0.87816	0.464015	0.054754
180	0.458828	0.385042	0.082462	0.654325	0.537007
181	0.639518	0.458196	0.668738	0.425485	0.343656
182	0.740883	0.5184	0.710568	0.749201	0.425873
183	0.704018	0.666469	0.593092	0.490975	0.260579
184	0.198578	0.100248	0.314035	0.671552	0.627486
185	0.352557	0.290425	0.562082	0.620381	0.52625
186	0.771486	0.656503	0.598962	0.482768	0.37828
187	0.423593	0.737359	0.468308	0.301413	0.221994
188	0.464542	0.440937	0.535364	0.602387	0.309591
189	0.321724	0.797079	0.582499	0.344203	0.504102
190	0.288576	0.337821	0.723877	0.263165	0.445785
191	0.613932	0.625725	0.51127	0.754007	0.341674
192	0.613908	0.36287	0.324011	0.586428	0.34886
193	0.750182	0.617441	0.521961	0.723623	0.349459
194	0.456003	0.790773	0.349456	0.84181	0.60152
195	0.041241	0.618466	0.471516	0.327953	0.670425
196	0.721223	0.453556	0.315828	0.454308	0.376264
197	0.481621	0.442015	0.505831	0.475574	0.415863
198	0.449819	0.425112	0.094826	0.66016	0.428274
199	0.641236	0.47871	0.38318	0.5181	0.453194
200	0.046617	0.812568	0.605113	0.184055	1
201	0.178341	0.378283	0.21779	0.307796	0.736171
202	0.737033	0.624465	0.753678	0.632932	0.399407
203	0.118272	0.54538	0.79106	0.53256	0.799845
204	0.3666	0.472677	0.239711	0.281976	0.271796
205	0.493242	0.81298	0.286123	0.297459	0.321156
206	0.411615	0.509306	0.838495	0.472056	0.647955
207	0.692549	0.906822	0.696857	0.555921	0.296991
208	0.294892	0.543787	0.511205	0.846898	0.490202
209	0.553685	0.545013	0.772987	0.470878	0.240884
210	0.559006	0.750806	0.243481	0.576436	0.438422
211	0.703941	0.362059	0.743907	0	0.212507
212	0.210561	0.198255	0.522586	0.560594	0.722062
213	0.345725	0.18662	0.566601	0.372043	0.360742
214	0.571607	0.511168	0.517085	0.432702	0.385309
215	0.410745	0.783081	0.315859	0.516969	0.59303
216	0.48148	0.4401	0.64332	0.744728	0.550536
217	0.473357	0.549133	0.572816	0.526059	0.636818
218	0.35469	0.686218	0.731861	0.502611	0.502142
219	0.765212	0.431405	0.521446	1	0.236636
220	0.391618	0.278593	0.370302	0.72814	0.440945
221	0.489066	0.5294	0.228971	0.489387	0.428202
222	0.455674	0.353617	0.562838	0.924875	0.259245
223	0.487561	0.454097	0.567439	0.647568	0.499908

Continua na página seguinte

Tabela C.1 – continuação da página anterior

Registo	Deformação (mm)	Intensidade (W)	Força de compactação (N)	Velocidade (mm/s)	Força de arranque (N)
224	0.549449	0.806105	0.387275	0.713854	0.49529
225	0.040916	0.315695	0.725578	0.318858	0.724798
226	0.120599	0.725208	0.692003	0.492014	0.745935
227	0.509093	0.338105	0.614396	0.431604	0.199415
228	0.808124	0.411059	0.747091	0.51019	0.141312

Referências

- [1] B. Acherjee, “Laser transmission welding of polymers – a review on process fundamentals, material attributes, weldability, and welding techniques,” *Journal of Manufacturing Processes*, vol. 60, pp. 227–246, 12 2020.
- [2] J. P. Coelho, M. A. Abreu, and M. C. Pires, “High-speed laser welding of plastic films,” *Optics and Lasers in Engineering*, vol. 34, pp. 385–395, 10 2000.
- [3] A. Dey, “Machine learning algorithms: A review,”
- [4] V. Sharma, S. Rai, and A. Dev, “A comprehensive study of artificial neural networks,” *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, p. 2277, 2012.
- [5] S. Sharma, S. Sharma, and A. Athaiya, “Activation functions in neural networks,” *International Journal of Engineering Applied Sciences and Technology*, vol. 4, pp. 310–316, 2020.
- [6] B. Mills and J. A. Grant-Jacob, “Lasers that learn: The interface of laser machining and machine learning,” *IET Optoelectronics*, vol. 15, pp. 207–224, 10 2021.
- [7] R. T. Kneusel, *Math for Deep Learning*. No Starch Press, 2021.
- [8] J. A. S. Sá, A. C. Almeida, B. R. P. Rocha, M. A. S. Mota, J. R. S. Souza, and L. M. Dentel, “Lightning forecast using data mining techniques on hourly evolution of the convective available potential energy,” pp. 1–5, 6 2016.
- [9] R. E. Schapire, “A brief introduction to boosting,” 1999.
- [10] M. Mehrpouya, A. Gisario, A. Rahimzadeh, and M. Barletta, “An artificial neural network model for laser transmission welding of biodegradable polyethylene terephthalate/polyethylene vinyl acetate (pet/peva) blends,” *International Journal of Advanced Manufacturing Technology*, vol. 102, pp. 1497–1507, 6 2019.
- [11] B. Acherjee, S. Mondal, B. Tudu, and D. Misra, “Application of artificial neural network for predicting weld quality in laser transmission welding of thermoplastics,” *Applied Soft Computing*, vol. 11, pp. 2548–2555, 3 2011.
- [12] M. R. Nakhaei, N. B. M. Arab, and F. Kordestani, “Modeling of weld lap-shear strength for laser transmission welding of thermoplastic using artificial neural network,” *Advanced Materials Research*, vol. 445, pp. 454–459, 2012.
- [13] H. P. Fu, C. Y. Chen, H. P. Yeh, W. C. Yeh, and J. S. Lu, “Combining bpn and tm to build a prediction model of process parameters: a case study,” *Total Quality Management and Business Excellence*, vol. 27, pp. 666–680, 5 2016.
- [14] P. Prates, M. Dias, and A. Pontes, “Controlo inteligente do processo de injeção,” 2021.
- [15] T. D. Hoang, Q. V. Nguyen, V. C. Nguyen, and N. H. Tran, “Self-adjusting on-line cutting condition for high-speed milling process,” *Journal of Mechanical Science and Technology*, vol. 34, pp. 3335–3343, 8 2020.

- [16] Docker, “Use containers to build, share and run your applications.” <https://www.docker.com/resources/what-container>. Accessed: 2022-03-12.
- [17] F. Dave, M. M. Ali, R. Sherlock, A. Kandasami, and D. Tormey, “Laser transmission welding of semi-crystalline polymers and their composites: A critical review,” *Polymers*, vol. 13, pp. 1–52, 3 2021.
- [18] L. F. Gonçalves, F. M. Duarte, C. I. Martins, and M. C. Paiva, “Laser welding of thermoplastics: An overview on lasers, materials, processes and quality,” *Infrared Physics & Technology*, vol. 119, p. 103931, 12 2021.
- [19] B. Acherjee, “Laser transmission welding of polymers – a review on welding parameters, quality attributes, process monitoring, and applications,” *Journal of Manufacturing Processes*, vol. 64, pp. 421–443, 4 2021.
- [20] R. Arboretti, R. Ceccato, L. Pegoraro, and L. Salmaso, “Design of experiments and machine learning for product innovation: A systematic literature review,” 2021.
- [21] W. L. Hosch, “machine learning.” <https://www.britannica.com/technology/machine-learning>. Accessed: 2021-12-18.
- [22] A. V. den Bosch, *Corpus Linguistics: An International Handbook*. Walter de Gruyter, 1 2009.
- [23] B. Fulkerson, D. Michie, D. J. Spiegelhalter, and C. C. Taylor, “Machine learning, neural and statistical classification,” *Technometrics*, vol. 37, p. 459, 11 1995.
- [24] L. Torgo and J. Gama, “Regression using classification algorithms,”
- [25] T. S. Chang, “A comparative study of artificial neural networks, and decision trees for digital game content stocks price prediction,” *Expert Systems with Applications*, vol. 38, pp. 14846–14851, 11 2011.
- [26] S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas, “Machine learning: a review of classification and combining techniques,” *Artificial Intelligence Review 2007 26:3*, vol. 26, pp. 159–190, 11 2007.
- [27] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, “State-of-the-art in artificial neural network applications: A survey,” *Heliyon*, vol. 4, p. 938, 2018.
- [28] W. Cai, J. Z. Wang, P. Jiang, L. C. Cao, G. Y. Mi, and Q. Zhou, “Application of sensing techniques and artificial intelligence-based methods to laser welding real-time monitoring: A critical review of recent literature,” *Journal of Manufacturing Systems*, vol. 57, pp. 1–18, 10 2020.
- [29] R. Hecht-Nielsen, “Theory of the backpropagation neural network,”
- [30] N. Srebro, A. Tewari, K. L. Clarkson, H. Israel, D. P. Woodruff, A. Cotter, S. Sra, S. Nowozin, and S. J. Wright, *Optimization for Machine Learning*, vol. 2. 2012.
- [31] J. Brownlee, *XGBoost With Python: Gradient Boosted Trees with XGBoost and scikit-learn - Jason Brownlee - Google Livros*. Machine Learning Mastery, v1.15 ed., 2021.
- [32] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- [33] Q. Ren, H. Cheng, and H. Han, “Research on machine learning frameworkbased on random forest algorithm,” vol. 1820, 3 2017.
- [34] T. Agrawal, *Hyperparameter Optimization in Machine Learning*. Apress, 2021.

- [35] S. Fraley, M. Oom, B. Terrien, J. Zalewski, R. Bredeweg, J. Morga, R. Sekol, and R. Wong, “4.1: Design of experiments via taguchi methods-orthogonal arrays.” <https://eng.libretexts.org/@go/page/22674>. Accessed: 2022-03-25.
- [36] J. Wu, S. Lu, H. J. Wang, Y. Wang, F. B. Xia, and Jin-Wang, “A review on laser transmission welding of thermoplastics,” *International Journal of Advanced Manufacturing Technology*, vol. 116, pp. 2093–2109, 10 2021.
- [37] N. Kumar, R. Rudrapati, and P. K. Pal, “Multi-objective optimization in through laser transmission welding of thermoplastics using grey-based taguchi method,” *Procedia Materials Science*, vol. 5, pp. 2178–2187, 1 2014.
- [38] R. G. Kumar, H. N. N. Murthy, B. Anand, and S. Jabiulla, “Parametric study of laser welding on polyamide using design of experiments,” *Journal of Physics: Conference Series*, vol. 1950, 8 2021.
- [39] A. G. Olabi, G. Casalino, K. Y. Benyounis, and M. S. Hashmi, “An ann and taguchi algorithms integrated approach to the optimization of co2 laser welding,” *Advances in Engineering Software*, vol. 37, pp. 643–648, 2006.
- [40] P. Moreira, “Sistema de apoio à gestão de componentes de manutenção na ria stone,” 2021.
- [41] A. Trujillo-Ortiz, R. Hernandez-Walls, and K. Barba-Rojo, “Henze-zirkler’s multivariate normality test.,” 12 2007.
- [42] M. Panik, *Advanced Statistics from an Elementary Point of View*. Elsevier, 2005.
- [43] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2 2015.
- [44] Docker, “Dockerfile reference.” <https://docs.docker.com/engine/reference/builder/#usage>. Accessed: 2022-03-15.
- [45] Docker, “Overview of docker compose.” <https://docs.docker.com/compose/>. Accessed: 2022-03-15.
- [46] Balena, “Build your iot project with balena..” <https://www.balena.io>. Accessed: 2022-03-15.