

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Desenvolvimento de Novas Funcionalidades Para a Aplicação de Retrabalhos

João Paulo de Oliveira Moutinho Cerqueira

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador da FEUP: Prof. Doutor João Pedro Mendes Moreira

July 21, 2022

Resumo

Atualmente, empresas que fabricam produtos para a indústria automóvel têm demonstrado um maior interesse tanto na diminuição da produção de produtos com falhas, como extinguir peças expedidas que contenham erros.

Sendo uma empresa que produz módulos de controlo e de climatização para veículos rodoviários, tem como obrigação manter uma qualidade elevada nos produtos desenvolvidos. Além disso, melhorias contínuas devem ser constantemente aplicadas, tanto nas máquinas de produção como nas estações de reparação e de qualidade do produto.

Melhorar tanto os processos de montagem como os processos de retrabalhos das peças é benéfico para a redução de produtos não aptos. Desta forma, o objetivo foi auxiliar a estação de retrabalho com o arranjo de peças não conformes de modo a recolher os erros alertados na fase de validação dos requerimentos do produto e disponibilizar ao operador, na fase de reparação das peças, quais os retrabalhos a realizar, diminuindo o erro humano na escolha do retrabalho a implementar.

Com a crescente automatização de processos e de testagem dos produtos produzidos, origina-se uma vasta variedade de erros que necessitam de ser interpretados para poderem ser resolvidos.

O principal objetivo desta dissertação é desenvolver e integrar um sistema de recomendações na aplicação existente de retrabalhos, que conforme o erro do produto, recomenda os retrabalhos mais apropriados, com base em ferramentas de *Data Mining* e de *Machine Learning*. Duas abordagens irão ser implementadas respeitando sempre as condições de ambiente da linha de produção, realizando depois uma comparação objetiva da performance entre ambas, elegendo a melhor e a mais eficiente para o caso em estudo.

Abstract

Currently, companies that manufacture products for the automotive industry have shown a greater interest in both decreasing the production of faulty products, and eliminating parts that contain errors.

As a company that produces control and climatization modules for road vehicles, it has an obligation to maintain a high quality standart in the developed products. In addition, continuous improvements must be constantly applied, both in the production machines and also in the repair and product quality stations.

Improving both the assembly and reworking processes is beneficial for the reduction of unsuitable products. Therefore, the main goal was to assist the rework station with the arrangement of the nonconforming parts, in order to collect the errors alerted in the validation phase of the product requirements and make available to the operator, in the repairing phase, which rework to be performed, reducing human error in the choice of witch rework to implement.

With the increasing automation of processes and product testing, a wide variety of errors originate that need to be interpreted in order to be solved. The main goal of this dissertation is to develop and integrate a recommendation system into the existing rework application, which according to the error, recommends the most appropriate rework based on Data Mining and Machine Learning tools. Two approaches will be implemented, always respecting the environmental conditions of the production line, then performing an objective comparison of performance, choosing the best and most efficient for the case under study.

Agradecimentos

Ao longo da realização desta dissertação, foram várias as pessoas que me apoiaram e ajudaram, e assim gostaria de lhes deixar um agradecimento.

Em primeiro lugar gostaria de agradecer ao meu orientador **Professor Doutor João Pedro Moreira**, por ter contribuído com sua disponibilidade todas as semanas e principalmente pelo o seu conhecimento sobre as matérias aprofundadas.

Gostaria de deixar um agradecimento ao meu supervisor na empresa, **Engenheiro Sílvio Granjo** e a toda a **equipa do departamento Software e EOL**, pela integração e partilha de conhecimento, providenciando uma ajuda preciosa na elaboração desta dissertação. Em particular, queria agradecer ao **Engenheiro Ricardo Sabino** que esteve sempre disponível para auxiliar nas minhas dúvidas e problemas nas linguagens de programação.

Um agradecimento a todos os meus amigos que fizeram parte deste percurso académico, especialmente ao **Bernardo Barbosa, Diogo Silva, Duarte Carvalho, Filipe Sousa e João Sá**, que tanto me ajudaram a descontraír durante períodos difíceis, como também foram os meus colegas das sessões de estudo. Ao meu parceiro árbitro e amigo **Luís Moreira**, que ao fim destes anos como dupla de Árbitros de Andebol Nacional, partilhámos grandes momentos profissionais.

Agradeço à minha **família**, em especial aos **meus pais**, por tudo o que me proporcionaram que me permitiu chegar a este ponto, mas principalmente pelo apoio incondicional que demonstraram ao longo do meu percurso académico.

Gostaria de agradecer à minha **namorada Maria do Mar**, pelo o amor incondicional e paciência nestes últimos meses trabalhosos e difíceis.

João Cerqueira

“Scientists study the world as it is, engineers create the world that never has been.”

Theodore von Karman

Conteúdo

Abreviaturas	xv
1 Introdução	1
1.1 Motivação e Contexto do Projeto	1
1.2 Empresa PREH	2
1.3 Objetivos	2
1.4 Metodologia Implementada	3
1.5 Estrutura da Dissertação	4
2 Funcionamento da Empresa	5
2.1 Produto Desenvolvido	6
2.2 Linha de Produção	8
2.3 Validação das peças	9
2.4 Estação de Retrabalhos	11
2.4.1 Aplicação Rework	11
3 Estado da Arte	13
3.1 Revoluções Industriais Anteriores	13
3.2 Indústria 4.0	14
3.2.1 <i>Internet of Things</i>	15
3.2.2 Cloud Computing	16
3.2.3 Sistemas ERP	17
3.2.4 Sistemas Ciberfísicos	18
3.2.5 <i>Digital Supply Chain</i>	18
3.3 <i>End-Of-Line</i>	19
3.4 <i>Data Analytics</i>	20
3.5 <i>Machine Learning</i>	21
3.6 Algoritmos de Classificação e Regressão	22
3.6.1 <i>Random Forest</i>	23
3.6.2 <i>Decision Tree</i>	23
3.6.3 <i>Naive Bayes</i>	24
3.6.4 <i>Logistic Regression</i>	25
3.6.5 <i>Support Vector Machine</i>	26
4 Método Implementado - Base de Dados	29
4.1 Organização da Base Dados	30
4.1.1 Análise das Tabelas e das Variáveis	30
4.2 Processamento de Dados	33

4.3	<i>Query</i> Desenvolvida	36
4.4	Implementação do Sistema de Recomendações na Aplicação	37
5	Prova de Conceito - Machine learning	41
5.1	Variáveis Categóricas	42
5.2	Treino do Modelo	43
5.3	Algoritmo Desenvolvido	44
5.4	Resultados Gerais	46
5.4.1	Análise Estatística dos Dados	46
5.4.2	Resultado das Recomendações dos Algoritmos	48
5.5	Implementação do Modelo na Aplicação	52
6	Conclusões Finais e Trabalhos Futuros	55
6.1	Conclusão sobre os Métodos	56
6.2	Trabalho Futuro	57
A	Anexo A - Códigos Desenvolvidos	61
A.1	SQL SERVER - Código SQL	62
A.1.1	<i>Query</i> não otimizada desenvolvida, antes de implementar Store Procedure.	62
A.1.2	Store Procedure implementado na aplicação de Produção (Pós otimização)	65
A.2	Machine Learning: Python e C#	67
A.2.1	Machine Learning Colab - Código Python	67
A.2.2	Machine Learning ML.NET - C#	72
B	<i>Flow Charts</i> dos dois métodos implementados	75
C	Tabelas adicionais aos <i>DataSets</i> utilizados	79
C.1	Tabelas Informativas sobre os <i>DataSets</i>	80
C.1.1	Top 30 Erros QWALL mais ocorridos	80
C.1.2	Top 30 Erros EOL mais ocorridos	81
C.1.3	Contagem de Retrabalhos mais Realizados no Dataset QWALL	82
C.1.4	Contagem de Retrabalhos Realizados Para os Erros de EOL	83
C.1.5	Contagem de Retrabalhos Realizados Por Dia	84
C.1.6	Contagem de Retrabalhos Realizados com Auxílio do Sistema de Recomendações	85
	Referências	87

Lista de Figuras

1.1	Diagrama explicativo sobre Diagrama da metodologia Crisp-DM.	3
2.1	Imagens do Módulo Standard Desenvolvido.	6
2.2	Imagens das Versões Premium do Produto.	7
2.3	Diagrama dos processos do EOL.	8
2.4	Diagrama esquemático do funcionamento de uma máquina de EOL.	9
2.5	Exemplo da Aplicação da Estação de Retrabalhos.	12
3.1	Diagrama cronológico das Revoluções Industriais.	14
3.2	Diagrama representativo dos componentes que engloba IoT.	15
3.3	Diagrama representativo de um sistema ERP.	17
3.4	Imagens da Linha EOL da PREH.	19
3.5	Processos para a implementação de Machine Learning.	22
3.6	Diagrama do algoritmo <i>Random Forest</i>	23
3.7	Diagrama do algoritmo <i>Decision Tree</i>	24
3.8	Diagrama do algoritmo NB.	25
3.9	Gráfico representativo do sistema de classificação utilizando um algoritmo regressão logística.	26
3.10	Gráfico representativo do início do processo de classificação do algoritmo SVM .	26
3.11	Gráfico representativo das margens criadas com o algoritmos SVM.	27
4.1	Diagrama UML das relações da base de dados.	33
4.2	Histórico da peça - Erros diferentes, retrabalho considera o último	34
4.3	Histórico da peça - Retrabalhos mal sucedidos (a vermelho)	35
4.4	Histórico da peça - Retrabalhos bem sucedidos sem erro intercalado	35
4.5	Histórico da peça - Exemplo de casos bem sucedidos de erros e retrabalhos . . .	35
4.6	Histórico da Peça - Erro EOL examinado pela estação qualidade	36
4.7	Excerto de código da <i>Query</i> não otimizada	36
4.8	Ambiente Inicial da Aplicação	38
4.9	Página de Recomendação da Aplicação	39
4.10	Ambiente de Recomendações da Aplicação com Instruções	40
5.1	Excerto de código relativo ao processamento de dados	42
5.2	Excerto de código relativo ao processamento de dados	43
5.3	Excerto de código relativo ao algoritmo de pertença	44
5.4	Excerto do código relativo ao algoritmo de ordenação	45
5.5	Excerto do código relativo algoritmo de avaliação	45
5.6	Demonstração da Aplicação de ML	53
5.7	Demonstração da Aplicação de ML	53

6.1	Comparação entre o desempenho das duas aplicações - ML vs SQL	57
6.2	Comparação explícita da inconsistência em certas recomendações	58
B.1	Diagrama da Implementação do Modelo de ML na Aplicação	76
B.2	Diagrama da Implementação através de SP na Aplicação	77

Lista de Tabelas

4.1	Formato dos dados após processamento.	34
5.1	Tabela representativa das características dos <i>datasets</i> em estudo	47
5.2	Recolha de alguns erros relativos que demonstra a desigualdade de classes presente nos <i>datasets</i>	47
5.3	Tabela representativa dos resultados dos algoritmos utilizados com os dados dos erros provenientes da estação de qualidade	50
5.4	Tabela representativa dos resultados dos algoritmos utilizados com os dados dos erros provenientes da linha de EOL	50
5.5	Tabela representativa dos tempos de treino de todos os algoritmos utilizados, para cada tipo de erro	51
5.6	Tabela representativa dos resultados dos algoritmos utilizados com recurso ao algoritmos de avaliação desenvolvido	51
5.7	Tabela representativa dos resultados do algoritmos de Pertença perante os algoritmos utilizados	52
C.1	Tabela representativa dos erros mais usuais provenientes da estação de qualidade	80
C.2	Tabela representativa dos erros mais usuais provenientes da linha de EOL	81
C.3	Tabela representativa dos retrabalhos mais realizados dos erros provenientes da estação de qualidade	82
C.4	Tabela representativa dos retrabalhos mais realizados dos erros provenientes da estação de qualidade	83
C.5	Número de retrabalhos realizados diariamente	84
C.6	Número de recomendações realizados diariamente	85

Abreviaturas

API	Application Programming Interface
BD	Base de Dados
CSV	Comma-separated values
CPS	Cyber-Physical System
CRISP-DM	Cross Industry Standard Process for Data Mining
DM	Data Mining
EOL	End of Line
ERP	Enterprise Resource Planning
GBM	Gradient Boosting Machine
GM	General Motors
HMI	Human Machine Interface
LBFGS	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
MART	Multiple Additive Regression Trees
ML	Machine Learning
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
NB	Naive Bayes
OVA	One Versus All
QWALL	Muro de Qualidade ou Estação de Qualidade
RF	Random Forest
SDCA	Stochastic Dual Coordinate Ascent
SP	Store Procedures
SQL	Structured Query Language
SVM	Support Vector Machine
UTF 8	Unicode Transformation Format 8
WPF	Windows Presentation Foundation

Capítulo 1

Introdução

Este estudo é derivado de um projeto de dissertação curricular, realizado em ambiente empresarial, cuja proposta consiste no melhoramento da linha de produção, mais concretamente na diminuição da variedade do processo de retrabalho de peças defeituosas.

Neste primeiro capítulo, o projeto alvo é apresentado e contextualizado, descrevendo o sector e a empresa, clarificando os objetivos e as áreas de investigação aprofundadas. Por fim, a metodologia adotada, bem como a estrutura escolhida para apresentar esta dissertação, serão devidamente abordados ao longo do documento.

1.1 Motivação e Contexto do Projeto

As tecnologias presentes nas empresas de manufaturação, especialmente na área da automação, cresceram exponencialmente nas últimas duas décadas, uma vez que muitos segmentos da indústria automóvel adotaram métodos destinados a reduzir os custos de mão-de-obra e materiais, aumentando simultaneamente a produtividade, a eficiência e a qualidade dos componentes.

De um ponto de vista estratégico, tornou-se relevante analisar a forma como a empresa pode continuar a crescer de forma sustentável, face ao crescimento exponencial das tecnologias emergentes da Indústria 4.0 e à continua progressão quanto à redução do erro Humano nas decisões importantes na linha de produção.

Cada vez mais as empresas têm consciência que erros de origem humana têm muita margem de melhoria, podendo ser mitigados mudando basicamente os procedimentos e guiando os trabalhadores durante os seus procedimentos de trabalho. Métodos como formações mais frequentes e a existência de um manual de instruções de trabalho claro e informativo são cruciais. Porém, existem outros fatores relacionados com o funcionário que não são possíveis controlar, tais como o cansaço, o foco no trabalho e o trabalho baseado na intuição.

Com base no que foi referido, este trabalho focar-se-á na indústria automóvel, que sendo um mercado muito competitivo e dispendioso, não se compadece com qualquer falha nas diversas cadeias de produção ou com atrasos que possam provocar o adiamento da entrega do produto final aos clientes.

1.2 Empresa PREH

Como fornecedor mundial da indústria automóvel, o grupo PREH tem cerca de sete mil colaboradores espalhados pela Europa, América do Norte e Ásia, com um volume de negócios anual superior a mil milhões de euros.

A PREH foi criada em 1919 em Bad Neustadt a. d. Saale, Alemanha, e faz parte do Grupo Joyson desde 2011. Atualmente, a Joyson é um dos trinta maiores fornecedores mundiais do sector automóvel. Dentro do Grupo, Joyson Ningbo na China, a PREH representa a Divisão *Automotive Electronics*. As capacidades de desenvolvimento e fabrico da PREH incluem sistemas de Interface Homem Máquina (HMI) para automóveis e veículos comerciais, bem como unidades de controlo de E-Mobilidade.

A inovação técnica e a flexibilidade são algumas das qualidades que caracterizam a PREH desde os seus anos de fundação. A empresa conta consistentemente com uma elevada integração vertical interna, e como resultado, os mesmos padrões são garantidos em todas as fábricas do mundo e a produção pode ser feita de forma flexível numa rede de produção internacional.

A PREH desenvolve e fabrica, por exemplo, unidades de controlo climático, controlos centrais, interruptores multi-funcionais, módulos com *feedback* tátil ativo para ecrãs tácteis e unidades de controlo para gestão de baterias. Deste modo, a PREH trabalha com os maiores fabricantes do mercado, como por exemplo: Audi, BMW, Ford, GM, Daimler, Mercedes-Benz, Porsche, Rolls-Royce e Volkswagen.

1.3 Objetivos

Esta dissertação tem como objetivo final realizar alterações à aplicação de retrabalhos, previamente desenvolvida pela PREH, que permitam, de acordo com o tipo de falha no produto, orientar o operador para o retrabalho que deve ser efetuado, diminuindo a interpretação subjetiva feita pelo operador. Este novo Sistema de Recomendações embutido na aplicação, deverá recomendar os três retrabalhos mais adequados para o erro em questão. Esta proposta foi realizada com o intuito de ir de encontro aos seguintes objetivos:

- Diminuir a variedade de retrabalhos utilizados para corrigir a peça;
- Diminuir a quantidade de peças que, após serem retrabalhadas, reprovam novamente devido a uma reparação previamente errada;
- Devido à falta de documentação dos erros originados como também dos retrabalhos efetuados, concentrando toda a sabedoria nos operadores dessa estação, era necessário criar um sistema de dupla validação para evitar retrabalhos desapropriados e ao mesmo tempo agilizar o processo;

Neste contexto, decidiu-se recorrer a dois métodos para alcançar os objetivos propostos, sendo diferenciados apenas pela forma como tratam os dados recolhidos. No caso do método que ficou estabelecido nas linhas de produção foi *queries* e *store procedures*, onde as recomendações provêm diretamente da base de dados e dependem de todo o histórico de peças retrabalhadas com erro.

O segundo método foi através de um modelo criado de *Machine Learning* onde as previsões do modelo são treinadas conforme o histórico de peças retrabalhadas. A escolha do método que ficaria funcional na linha de produção foi feita devido às limitações que as próprias empresas de manufatura enfrentam, tais como limitações de acesso à internet em todos os computadores da linha de produção, existindo só acesso aos servidores internos por razões de cibersegurança, com diferentes versões de sistemas operativos e de *frameworks*, equipamentos desgastados e antigos e, por último, a necessidade da aplicação ser correr no mesmo ambiente operacional sem que haja aplicações adicionais que permitam alcançar o mesmo fim.

Deste modo, foram desenvolvidas ambas as aplicações, em que uma está integrada nos computadores da estação de retrabalhos e a outra foi desenvolvida para prova de conceito e de comparação entre as recomendações de ambas as aplicações.

1.4 Metodologia Implementada

Cross Industry Standard Process for Data Mining CRISP - DM [1] é uma metodologia prática, flexível e útil na resolução de questões de inúmeros e variados dados. Esta metodologia, ou processo de *Data Mining*, visa fornecer um plano para orientar um projeto de DM, melhorando o processo de recolha de informação proveniente da base de dados [2]. Estas fases são mostradas na figura 1.1 e serão descritas a seguir:

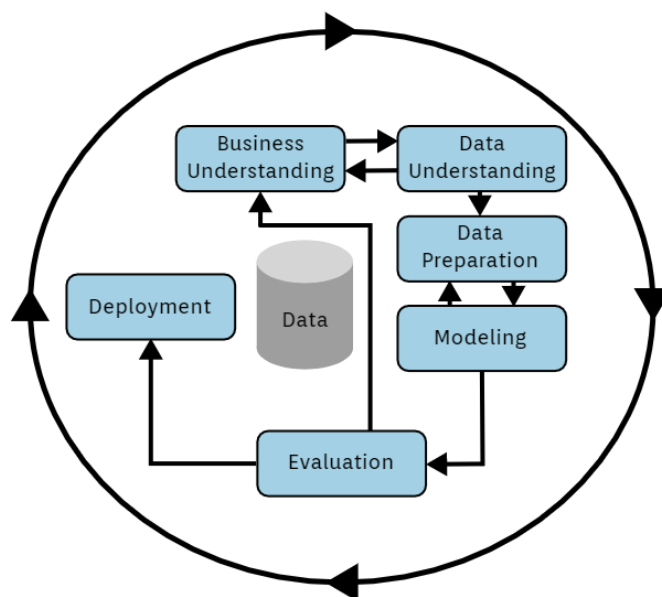


Figura 1.1: Diagrama explicativo sobre Diagrama da metodologia Crisp-DM.

- **Business Understanding:** Nesta fase, os objetivos e os requisitos do projeto são definidos de modo a que o problema possa ser formulado como um problema de DM. Além disso, o plano de trabalho é também determinado;
- **Data Understanding:** Corresponde à recolha inicial de dados, o analista prossegue com atividades para se familiarizar com os dados, identificar problemas de qualidade dos mesmos

e descobrir os primeiros conhecimentos sobre os dados. Nesta fase, o analista pode também detetar subconjuntos interessantes para formular hipóteses de informação desconhecida até à data;

- **Data Preparation:** Esta fase engloba várias ações que, após terem sido realizadas, resultam num conjunto final de dados, onde as variáveis relevantes são selecionadas ou retiradas e os dados errados são determinados. Simultaneamente, os formatos são definidos, novas características poderão ser adicionadas e os dados de várias fontes podem ser numa só tabela;
- **Modeling:** A fase de modelação de dados consiste na seleção da técnica de modelação, construção do caso de teste e a obtenção do modelo. Aqui, todas as técnicas de DM podem ser utilizadas. De forma generalizada, a escolha das técnicas depende do problema do negócio e dos próprios dados, sendo o mais importante o modo de como se explica todas escolhas para se obter o modelo. Para avaliação do modelo resultante, é apropriado avalia-lo em relação aos critérios de avaliação e selecionar os que produzem melhores desempenhos;
- **Evaluation:** Na fase de validação, os resultados são verificados em relação aos objetivos empresariais definidos. O analista testa-os para garantir que podem generalizar o modelo em relação aos dados não vistos. Subsequentemente, o analista também valida se o modelo cobre com êxito todas as questões-chave do negócio;
- **Deployment:** Nesta última fase, é realizado uma representação de código do modelo originado num programa ou num sistema operativo, incluindo também os mecanismos para pontuar ou categorizar novos dados que ainda não foram trabalhados, à medida que estes surgem. A representação de código deve incluir todas as etapas de preparação dos dados que conduzem até à criação do modelo. Este processo assegura que o modelo tratará novos dados (nunca trabalhados), da mesma forma que durante o desenvolvimento do modelo, sendo assim um processo automatizado.

1.5 Estrutura da Dissertação

Esta dissertação é desenvolvida ao longo de seis capítulos, que são concebidos e preparados para fornecer uma explicação detalhada de todo o contexto e da metodologia utilizada para o projeto.

Este documento começa por introduzir a empresa e o seu funcionamento, principalmente nas áreas onde o projeto incide. Na revisão do estado da arte, explora as principais áreas de investigação relacionadas com o tema da dissertação: inovações da Indústria 4.0, *Data Analytics* e *Machine Learning*. Após as componentes teóricas, serão desenvolvidos os dois métodos concebidos. No capítulo quatro, será apresentada a abordagem ao problema a partir da base de dados e de *queries*. No capítulo seguinte, será abordado o segundo método desenvolvido à base de algoritmos de *Machine Learning*, que permanece como prova de conceito e de comparação de resultados. No capítulo seis, serão descritas as conclusões gerais sobre os métodos implementados, bem como o trabalho futuro a ser desenvolvido.

Capítulo 2

Funcionamento da Empresa

A PREH é um produtor e distribuidor de peças e componentes para automóveis. Os seus produtos incluem sistemas de baterias, unidades centrais de controlo, interruptores inteligentes, sistemas de informação e entretenimento e consolas de comutação, entre outros.

Dada a sua reputação, a PREH possui fortes regimes de controlo e de garantia de qualidade para que os seus clientes fiquem satisfeitos com os produtos recebidos. Uma vez que as peças produzidas têm como destino o setor automóvel, estas não devem apresentar quaisquer erros de fabrico. Sendo assim, é importante e fundamental garantir que todas as 10 mil peças expedidas diariamente não apresentem uma única falha.

A empresa a nível tecnológico é obrigada a adotar tecnologias da indústria 4.0, e ao dia de hoje, todas as máquinas das linhas de montagem estão automatizadas, comunicando todas entre si através dos servidores internos da PREH, sendo o operador a confirmar e a guiar os procedimentos de montagem de cada peça.

Todo o software e aplicações instaladas nas máquinas são desenvolvidos pela PREH, permitindo que operador controle e perceba se os procedimentos de montagem estão a ser cumpridos. Este sistema permite também que todas as informações e valores de parâmetros sejam registados na base de dados correspondente ao *Trace Number* da peça que é basicamente um número de série único de catorze dígitos. Deste modo, a PREH consegue conhecer e guardar o historial de todas as peças uma vez que fica devidamente gravado, desde o processo de montagem final e EOL¹, como também o histórico de todos os sub-módulos da peça.

Estas tecnologias de tracking da empresa, além de permitirem monitorizar todas as peças, também permitem que todas as máquinas confirmem se as peças que vão trabalhar passaram previamente por todas as máquinas anteriores, impedindo assim que as peças passem, por engano, por processos de montagem.

Outro fator de prevenção inserido nos processos de cada máquina é a realização de uma validação das transformações realizadas no processo anterior, a fim de detetar mais cedo, erros fulcrais de produção dos produtos.

¹Explicado em detalhe ao longo do documento

2.1 Produto Desenvolvido

Desde 1992, a empresa produz componentes eletrônicos para a indústria automóvel e, entre os seus clientes, encontram-se os maiores fabricantes automóveis a nível mundial. Como exemplo, a BMW que representa um quarto do total de vendas do grupo, seguido pela Volkswagen, Audi e Seat.

Esta dissertação incidiu na linha da BMW BZM, que é uma das maiores linhas de produção atualmente na PREH Portugal. O produto desenvolvido é um *Center Stack* para os modelos da BMW Series, para o X5 e para o G20. Esta produção é exclusiva à PREH a nível global, ou seja, é responsável pelo fabrico de todas as versões deste módulo para todos os carros, inclusive a versão inglesa do mesmo. Nas imagens a seguir, apresenta-se a versão standard do produto 2.1



(a) Módulo Standard Vista de Cima.



(b) Módulo Standard Vista Perfil.

Figura 2.1: Imagens do Módulo Standard Desenvolvido.

Este modelo apresenta diferentes versões, onde a maioria das diferenças são cosméticas. Além da versão standard, existe uma versão M (figura 2.2a) onde a peça apresenta cores diferentes e um revestimento em couro Knob² e, por último, a versão Swarovski (figura 2.2b) onde o Knob é constituído por uma peça de cristal.

²O Knob, nos carros elétricos, é peça responsável por alternar os diferentes estados de movimentos do carro.



(a) Módulo Versão M Vista de Cima.



(b) Módulo Versão Swarovski Vista de Cima.

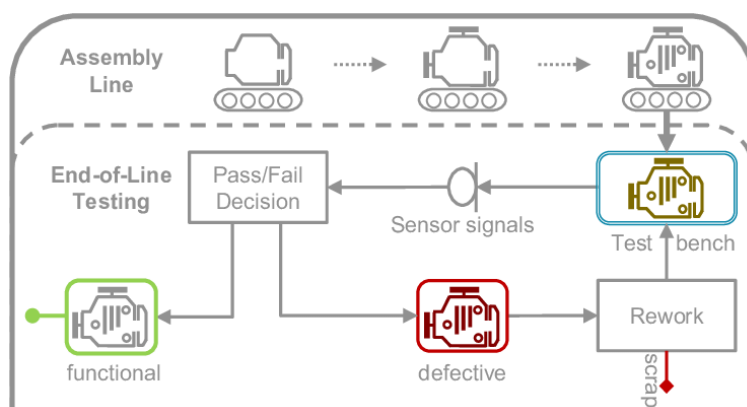
Figura 2.2: Imagens das Versões Premium do Produto.

Além do Knob, apresenta mais funcionalidades como Start/Stop, alternar entre os modos de desempenho do carro, botões de controlo do sistema de infoentretenimento e funcionalidades de ajuste de altura do carro.

Quanto aos módulos presentes neste produto, apresentam botões digitais com resposta de retorno ao input dado. Estes botões incluem dois tipos de *feedback* háptico: tátil e cinestésico. O *feedback* tátil aborda a percepção tátil a partir da pele, como as vibrações. O *feedback* cinestésico aborda a percepção cinestésica do próprio esforço muscular. Os sentidos visuais e auditivos distinguem assim as duas funcionalidades, uma vez que a maioria dos dispositivos de estimulação tátil requer um contacto físico com os utilizadores, sem que estes necessitem de confirmar a concretização da ação realizada. Deste modo, permite de uma forma eficaz a redução da carga visual para confirmar assertividade da ação desejada, diminuindo as distrações durante a condução.

2.2 Linha de Produção

Todas as linhas da PREH apresentam quase o mesmo layout. Este layout é com base no produto desenvolvido. Esse produto é então repartido em módulos funcionais, em que cada um tem a sua linha de produção. No final, existe uma linha de junção de todos os módulos e antes de ser expedido para o cliente, o produto passa por uma linha de verificação e validação de todos os componentes funcionais e cosméticos do produto. Estas linhas EOL são totalmente automatizadas uma vez que devido às normas impostas pelos clientes das grandes marcas que operam no setor automóvel, todos os produtos têm que passar por testes rigorosos que são submetidos a todos os componentes, que são, por sua vez, automatizados para evitar desvios nos parâmetros de avaliação das peças produzidas. Os produtos que estejam aptos são expedidos para o cliente. A figura 2.3 demonstra os processos da linha EOL.



Fonte da imagem:[3]

Figura 2.3: Diagrama dos processos do EOL.

Por outro lado, os produtos não aptos são redirecionados diretamente para a estação de retrabalhos da linha correspondente, ou dependendo do erro da fase de EOL, são enviados primeiro para a estação de qualidade (muro de qualidade) e o produto é revisto por operadores especializados, que conforme o erro originado, enviam para o início da linha de EOL ou para a estação de retrabalhos. Esta estação é importante pois permite serializar possíveis falsos erros que as máquinas da linha de EOL, erradamente estejam a detetar. Com o sistema de rastreamento, os operadores da estação de toda a linha de montagem conseguem diagnosticar as peças com falhas de componentes ou riscos e tomar decisões sem ter que conhecer as pessoas que as trabalharam previamente, criando um processo preventivo de falhas cruciais ao longo da linha, evitando sobrecarregamento do EOL.

Devido ao elevado rigos das normas e dos requerimentos das peças, os intervalos de aceitação dos parâmetros dos sensores na linha EOL por vezes são muito curtos o que permite que falsos erros possam aparecer. Por sua vez, estes falsos erros podem ser derivar de uma má colocação da peça no suporte onde incide o teste da máquina ou de uma ligeira descalibração dos sensores responsáveis.

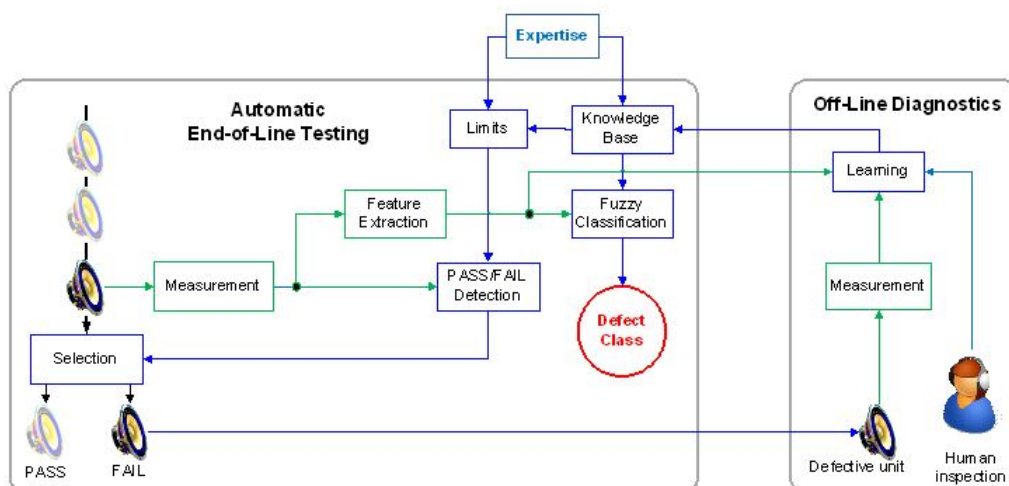
As peças que forem encaminhadas para a estação de retrabalhos são retrabalhadas de acordo com o erro e o histórico da peça. Após o retrabalho, as peças são enviadas novamente para a estação de EOL.

Todo este processo é repetido até que a peça seja expedida. Quando após várias reparações, uma peça reprova constantemente na linha de EOL, esta é descartada para reaproveitamento de peças dos conjunto de atuadores funcionares ou módulos do produto final. Esta decisão é tomada porque a reparação da peça já não é vantajosa para empresa devido à extensa alocação de recursos na reparação da mesma.

2.3 Validação das peças

Todas as peças desenvolvidas na PREH são validadas de acordo com todos os requerimentos do produto, sendo uma condição exigida tanto pelos clientes como também pela PREH. Esta validação pode ser dividida em dois tipos: funcional e cosmético. Na fase EOL, as peças finais sofrem inúmeros testes, tais como:

- **Pressão das Teclas:** as teclas devem ter uma certa resistência ao clique do botão mas estes não podem estar nem folgados nem muito justos ao encaixilhamento do próprio botão
- **Som:** os botões devem produzir um som de confirmação de ação do condutor, no entanto, este não pode estar acima do decibéis estabelecidos;
- **Visão:** as peças são examinadas quanto a problemas íntimos de cosmética. Além disso, também é testado o funcionamento correto das luzes que estes produtos exteriorizam, testando a conformidade e a intensidade correta de todas as luzes.



Fonte da imagem:[4]

Figura 2.4: Diagrama esquemático do funcionamento de uma máquina de EOL.

Neste processo de validação (figura 2.4), a automatização total das máquinas de EOL é chave para se alcançar conformidade, fiabilidade e confiança de que todas as peças expedidas estão

aptas para compor o produto final, neste caso, um veículo rodoviário. Em todas as peças que reprovem na linha EOL, é produzido um erro composto por dois componentes, designados por Teste Característico e Teste do Objeto. Estes erros são compostos somente por dígitos, no máximo pode conter quatro.

No primeiro segmento de erro é identificado o tipo de teste em que a peça reprovou, enquanto que no outro segmento é assinalado a localização do teste realizado. A necessidade do erro estar repartido em tipo e localização do teste é necessário visto que existe diferentes componentes onde o teste realizado é igual, como é o caso de certos botões, onde os parâmetros de teste são os mesmos. Além dos erros produzidos que são guardados na base de dados, as máquinas também produzem um texto de erro onde expõem em que parâmetros a peça reprovou.

Como já foi referido anteriormente, as peças que reprovam passam na estação de qualidade antes de ser efetuado qualquer intervenção à peça. Nesta estação, podem tomar as seguintes decisões consoante o tipo de erro em questão com a peça:

- Encaminhar automaticamente para a estação de retrabalho para reparação;
- Voltar a retestar as peças, colocando no início da linha EOL, para mitigar a reparação de peças sem possíveis erros;
- Realizar uma verificação manual extra dos diferentes componentes reprovados. Após verificação, estes podem ser redirecionados para a estação de retrabalhos ou para o início da linha EOL.

Nesta sub-estação da muralha de qualidade, existe uma pequena diferença quanto à alteração da estrutura do modelo do erro. Quando as peças são submetidas a esta verificação humana, conforme o terceiro ponto anteriormente mencionado, o operador realiza a verificação dos componentes reprovados e redireciona para a estação retrabalhos. Porém neste caso, nesta sub-estação o código de erro é alterado, independentemente do erro da peça, para 6023-603 que significa muralha de qualidade e a informação consoante o erro é descrito por uma frase standard respetivo ao teste manual reprovado, separado num setor extra de informação da peça. Toda esta informação fica registada no histórico do mesmo.

2.4 Estação de Retrabalhos

A estação de retrabalhos, da linha onde a dissertação teve ênfase, é composta por quatro estações. Por dia são realizados entre quatrocentos a setecentos retrabalhos, divididos por três turnos, em equipas de quatro a cinco técnicos.

Todos os técnicos da estação de retrabalhos estão devidamente qualificados e têm um bom conhecimento do produto desenvolvido, desde as falhas originadas às abordagens corretas de reparação consoante o erro. A responsabilidade e a sabedoria necessária para a decisão do tipo de reparação a realizar é única e exclusivamente dos próprios técnicos, não existindo nenhum documento com normas de reparações para todos os erros originados. Para contrariar a inexistência dessa documentação, a PREH investe na formação destas pessoas, permitindo que as mesmas consigam diagnosticar na integra o tipo de abordagem a implementar, de modo a permitir a resolução eficaz dos problemas.

Apesar da total automatização das máquinas na linha EOL, os inúmeros testes exercidos a cada peça geram uma enorme diversificação de erros. Numa análise superficial, a PREH desde 2020 nesta linha de produção em estudo, obteve mais de três mil erros diferentes, em que todos os dias a ocorrência de um novo erro é uma possibilidade, dificultando o trabalho destes técnicos quanto à correta reparação e acarretando muitas dificuldades quanto à uniformidade entre os técnicos dos diferentes três turnos para a utilização das reparações mais indicadas.

As peças que chegam à estação de retrabalho podem originar de três secções distintas:

- Linha de EOL;
- Estação de qualidade;
- Linhas de produção, com falhas significativas.

O procedimento dos técnicos quanto à reparação das peças é semelhante, independentemente do erro. Antes de executar a reparação, o técnico necessita de aceder ao histórico da peça para ter conhecimento em que parte da linha a peça reprovou e o seu erro originado. Após a reparação e dependendo do tipo de reparação realizada, as peças são enviadas para diferentes posições da linha de produção. Caso o técnico necessite de realizar uma desmontagem total do produto ou realizar uma troca do componente, este é enviado para o local respetivo da linha de produção onde se realizam estas montagens e todo o percurso da peça é novamente repetido.

2.4.1 Aplicação Rework

Na estação de retrabalhos, além da aplicação que permite consultar o histórico da peça, os técnicos dispõem de uma aplicação específica que utilizam para descrever na base de dados do tipo de reparação efetuada para o erro em questão. Na figura 2.5 apresenta-se a aplicação utilizada originalmente na empresa.

Durante este processo, os técnicos são obrigados a identificar o tipo de reparação (cosmético ou funcional), o sub-módulo da peça final que sofreu a reparação, um diagnóstico sobre o motivo da falha e, por fim, o retrabalho utilizado para o seu conserto.

Rework Station

FICHEIRO IDIOMA SETUP CONFIGURAÇÕES AJUDA

TraceNr: 00300079435032

Servidor: Preh Portugal

Projeto: BMW BZM Auto

Utilizador: João Cerqueira - 4279

Info da Peça

Ref: 90170-054/0011

Descrição: BZM-M AT HI X LL M PS

Grupo: FINAL M

REWORK

Diagnóstico

Grupo: Bottom Plate

Tipo de Defeito: Functional

Diagnóstico: Motor com Ruído

Rework Steps

Ação: Desmontar Produto Final - Linha 2

Rework Steps

1. Desmontar Componente 'BeMiko' and release to station 'OP260 - Carrier & BeMiko'
2. Desmontar Componente 'Bottom Plate' and release to station 'OP265.2 - Motor Screwing'

Aplicar

Figura 2.5: Exemplo da Aplicação da Estação de Retrabalhos.

Todos estes campos de informação listados já contém todas as possibilidades pré-definidas para todas secções, ou seja, o técnico não tem de descrever o que fez mas sim selecionar a informação previamente apresentada. Porém, há uma dificuldade em manter a consistência nos inputs selecionados quando existe cerca de três mil erros diferentes registados, e tal tarefa pode resultar em incongruências na informação inserida na aplicação, sendo frequente a realização dos retrabalhos corretos e, porventura, inserir erradamente o retrabalho efetuado.

No contexto deste produto, existem cerca de cinquenta e um possíveis retrabalhos diferentes que o técnico pode implementar. No caso dos diagnósticos existem cerca de cem possibilidades diferentes e quanto aos sub-módulos, existem cerca de 10 grupos.

Além das ferramentas de reparação, dos componentes novos de substituição e dos computadores de trabalho, os trabalhadores possuem um scanner em todas estações que permite o input automático através da leitura de um QR code.

Esta aplicação foi desenvolvida em C#, em ambiente .NET [5] usando o *framework* [6] 4.5.2 em WPF (*Windows Presentation Foundation*).

Capítulo 3

Estado da Arte

Este estudo insere-se num projeto de dissertação desenvolvido em ambiente empresarial que visa o melhoramento de uma aplicação de retrabalhos, inserida no final da linha de produção. Esta estação é responsável pelo retrabalho das peças que reprovaram na fase de EOL.

Neste capítulo, o projeto alvo é apresentado e contextualizado, descrevendo as tecnologias em estudo, bem como a clarificação dos objetivos e das áreas de investigação que se destinam a ser respondidos. Finalmente, a metodologia bem como a estrutura adotada para apresentar esta dissertação será explicada ao longo do documento.

3.1 Revoluções Industriais Anteriores

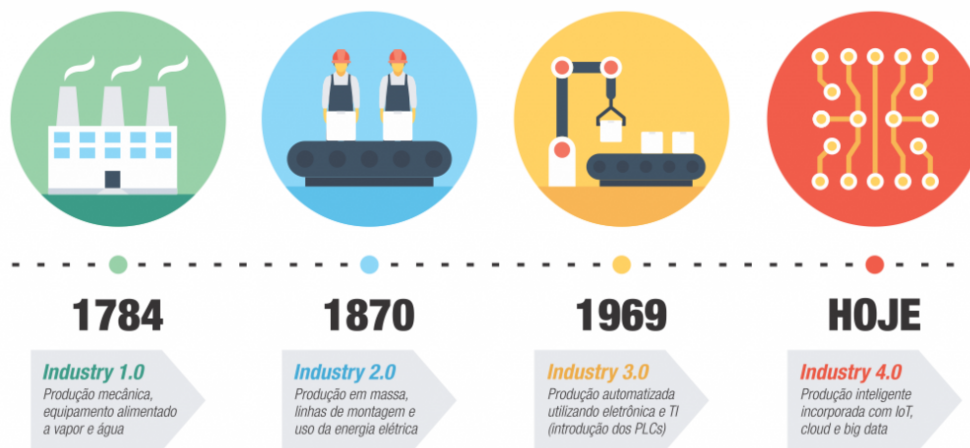
Antes da Revolução Industrial, as pessoas viviam num ambiente em que os bens e materiais eram feitos à mão. No final do século XVIII, houve a primeira Revolução Industrial, caracterizada pela transição para novos processos de fabrico, otimizando objetivos tradicionais, como custo, qualidade e inovação.

Por volta de 1740, dá-se uma transição de um fabrico doméstico artesanal em pequena escala para uma tendência de produção em massa, pela industrialização da indústria têxtil e por um desenvolvimento significativo de mineração para encontrar carvão. Por sua vez, o seu uso impulsionou a produção de aço e proporcionou o crescimento da indústria de construção civil, com o aumento das populações dos centros industriais.

A Segunda Revolução Industrial foi uma época em que se começou a introduzir fontes de energia mais vantajosas, tais como o petróleo [7]. Esta revolução industrial, que começou na Grã Bretanha, trouxe grandes avanços na agricultura, nas empresas de manufatura e no transporte, originando uma melhoria na qualidade de vida, ao mesmo tempo que provocou um aumento do desemprego devido às máquinas que substituíam os operadores nos postos de trabalho. Porém, os custos de produção e os preços caíram drasticamente e houve um rápido crescimento da produtividade industrial.

A Terceira Revolução Industrial, também conhecida como a Revolução Digital, começou nos anos 70 do século XX através da automatização parcial utilizando controlos e computadores programáveis com memória. O ponto central desta geração é a produção em massa e o uso generali-

zado da lógica digital, transístores MOS, chips de circuitos integrados e suas tecnologias derivadas incluindo computadores, microprocessadores, *smartphones* e a Internet. Estas inovações tecnológicas transformaram os métodos tradicionais de produção e de negócios. Houve revolução digital, onde tecnologia analógica é convertida para formatos digitais. É também importante mencionar que a Indústria 3.0 ainda está presente, estando muitas das fábricas atualmente a este nível de evolução. A figura 3.1 ilustra cronologicamente as diferentes Revoluções Industriais.



Fonte da imagem:[8]

Figura 3.1: Diagrama cronológico das Revoluções Industriais.

3.2 Indústria 4.0

A indústria 4.0 representa uma nova etapa na organização e controlo da cadeia de valor industrial. Os sistemas ciberfísicos formam a base da Indústria 4.0 [9] (máquinas inteligentes). Utilizam sistemas de controlo modernos através de softwares incorporados e dispendo de um endereço Internet para se ligarem e serem abordados através dos equipamentos IoT (Internet of Things). O objetivo é permitir os processos de tomada de decisão autónomos, que permitem monitorizar todos os equipamentos eletrónicos e processos em tempo real, originando a criação de valor em toda a rede interligada. Uma das mais recentes evoluções tecnológicas são os sistemas reconfiguráveis, que tornam possível adaptar os componentes de hardware e software de acordo com os requisitos de mercado. Isto significa que as máquinas não se desvalorizam com o tempo e a qualidade de produção será garantida.

Assim, a Indústria 4.0 é uma visão mais ampla com estruturas claras e arquiteturas de referência, caracterizada principalmente pela ponte entre os bens industriais físicos e as tecnologias digitais nos chamados sistemas ciberfísicos, representando a quarta revolução industrial no caminho para uma cadeia de valor de ponta a ponta da empresa, utilizando equipamentos IoT e a inteligência descentralizada na fabricação.[10]

3.2.1 Internet of Things

Nas últimas duas épocas, a Internet cresceu exponencialmente, fornecendo a qualquer hora, em qualquer lugar, conectividade independentemente da pessoa ou do material. Podendo ser alargada a objetos de rotina, os serviços da Internet expandiram-se no que é designado por *Internet of Things* (IoT) que consiste essencialmente na integração de pessoas, de processos e de tecnologia com dispositivos e sensores conectáveis para permitir uma monitorização remota, visualização ou manipulação do estado do sistema e avaliação das tendências de tais dispositivos [11]. No fundo, pode ser vista como a interação entre o mundo físico e o digital através de sensores e atuadores. IoT pode ser definido como o paradigma em que as capacidades computacionais e de rede estão incorporadas praticamente em qualquer objeto e utilizado para consultar e alterar o estado do mesmo.



Fonte da imagem:[12]

Figura 3.2: Diagrama representativo dos componentes que engloba IoT.

Este conceito pode ser aplicado em diversas áreas de coisas como dispositivos móveis, nas redes de transporte e em inúmeras linhas de produção [13]. O aumento da utilização de aparelhos IoT, permite que hajam vários pontos onde a cibersegurança e a privacidade de partilha de dados não sejam conseguidas. Por outro lado, com esta tecnologia, as empresas enfrentam um aumento considerável na quantidade de dados que estes dispositivos produzem, sendo importante descobrir formas de os poder armazenar, localizar, analisar e dar sentido a esta vasta quantidade de dados gerados.

3.2.2 Cloud Computing

As soluções baseadas na nuvem oferecem aos fabricantes uma vasta gama de benefícios, entre os quais os mais significativos: escalabilidade, eficiência operacional, integração em aplicações, armazenamento, gestão, análise de dados e uma maior segurança. Em particular, a computação em nuvem facilita a investigação, conceção e desenvolvimento de novos produtos, o que potencia a inovação, reduz os custos de desenvolvimento de produtos e acelera o tempo de colocação no mercado dos respetivos produtos[14]. Esta solução ajuda os fabricantes a gerir os seus negócios com maior inteligência, tornando-se possível através da utilização alargada dos dados que são analisados por ferramentas de DM [15]. Na prática, a nuvem está rapidamente a tornar-se o local central para armazenamento de dados, análise e inteligência para a maioria dos fabricantes.

Nos dias de hoje, existem nuvens empresariais que se limitam a uma única organização, nuvens públicas disponíveis para muitas organizações ou mesmo nuvens híbridas que é uma combinação de ambas. A disponibilidade de redes de alta capacidade, computadores e dispositivos de armazenamento de baixo custo, bem como a adoção generalizada de hardware de virtualização, criação de arquiteturas orientadas para os serviços de computação autónoma e utilitária levou ao crescimento da computação em nuvem.

A gama de computação em nuvem disponível é vasta, mas a maioria enquadra-se numa das seguintes categorias:

Platform as a Service (SaaS): é um método de entrega de software que permite o acesso a dados a partir de qualquer dispositivo com uma ligação à Internet e um navegador da Web. Neste modelo baseado na web, os fornecedores de software fornecem e mantêm os servidores, as bases de dados e o código que compõem as diversas aplicações da empresa. As implementações SaaS não requerem hardware adicional ou complexo, o que permite aos compradores atribuir a maior parte das responsabilidades da empresa que forneça estes serviços de nuvem. Os sistemas SaaS são tipicamente pagos dentro de um modelo de subscrição, enquanto que o software local é normalmente adquirido através de uma licença perpétua, paga adiantadamente.

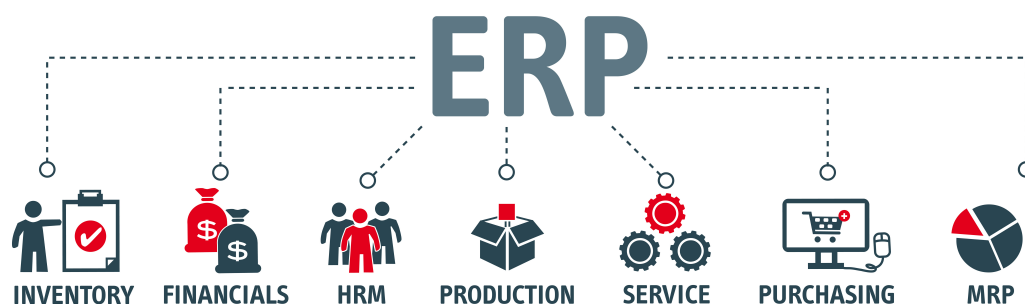
Infrastructure as a Service (IaaS): Fornece armazenamento e serviços computacionais através de um pagamento de um plano por utilização do serviço. No entanto, a gama completa de serviços oferecidos por todos os grandes fornecedores de nuvens públicas é bastante alargada e variada como: bases de dados dinâmicas ou escaláveis, ferramentas de desenvolvimento e de análises de dados, redes virtuais privadas, implementação de ML e monitorização de aplicações;

Platform as a Service (PaaS): fornece serviços e fluxos de trabalho direcionados para programadores, que utilizam ferramentas, processos e APIs partilhados para desenvolver, testar e implementar em aplicações. PaaS assegura que os criadores tenham acesso aos recursos, sigam os processos e utilizem apenas serviços específicos;

Functions as a Service (FaaS): É um tipo de serviço de cloud computing que permite executar código em resposta a eventos sem a complexa infra-estrutura tipicamente associada à construção e lançamento de aplicações. O alojamento de uma aplicação de software na Internet requer tipicamente o aprovisionamento e a gestão de um servidor virtual ou físico, gestão do próprio sistema operativo e o processamento de processos de alojamento de servidores Web. Com FaaS, o hardware físico, o sistema operativo das máquinas virtuais e a gestão de software de servidor web são todos tratados automaticamente pelo fornecedor de serviços da nuvem.

3.2.3 Sistemas ERP

O planeamento de recursos empresariais (ERP) é um processo utilizado pelas empresas para gerir e integrar as diversas partes importantes do seu negócio. Muitas aplicações de software ERP são importantes para as empresas porque as ajudam a implementar o planeamento de recursos, integrando todos os processos necessários para gerir a sua empresa através de um único sistema [16]. Um sistema de software ERP também pode integrar planeamento, inventário de compras, vendas, marketing, finanças, recursos humanos, entre outros.



Fonte da imagem:[17]

Figura 3.3: Diagrama representativo de um sistema ERP.

A implementação de um sistema ERP implica geralmente mudanças significativas nos processos e práticas de trabalho de todo o pessoal responsável [18], geralmente com a ajuda de três tipos de serviços: consultoria, personalização e suporte. Geralmente, a má compreensão dos processos necessários nas empresas antes da transformação para um sistema ERP é a principal razão para uma potencial fracasso no projeto. Por conseguinte, é crucial que as organizações analisem minuciosamente os processos antes de implementarem um software ERP, sendo mais difícil de implementar em organizações descentralizadas, uma vez que estas têm frequentemente diferentes processos, semântica de dados diferentes, hierarquias de autorização ou privilégios e centros de decisão em diferentes localizações da sua empresa.

A maior vantagem de um sistema ERP é que a integração de múltiplos processos empresariais poupa tempo e despesas. As equipas que intervêm nos processos podem tomar decisões mais

rapidamente e com menos erros. Os dados tornam-se visíveis em toda a organização, o que beneficia múltiplas tarefas, tais como, previsão de vendas, elaboração de um inventário mais otimizado, seguimento de encomendas desde a aceitação até ao cumprimento e centralização dos dados da Empresa.

3.2.4 Sistemas Ciberfísicos

Os sistemas Ciberfísicos (CPS) são construídos a partir da integração perfeita de algoritmos computacionais e componentes físicos. Estes sistemas combinam: dispositivos digitais e analógicos, sensores e atuadores, redes de comunicação e computadores. Tal como a Internet transformou a forma como as pessoas interagem com a informação, os sistemas ciberfísicos estão a transformar a forma como as pessoas interagem com o mundo físico.

Ao contrário dos sistemas incorporados mais tradicionais, um CPS [19] totalmente desenvolvido é concebido como uma rede de elementos que interagem com a entrada e saída física, em vez de dispositivos isolados, e são estritamente ligados a conceitos de robótica e redes de sensores com mecanismos de inteligência apropriados para inteligência computacional. A ligação entre elementos computacionais e físicos aumenta a adaptabilidade, eficiência, funcionalidade, fiabilidade, segurança e usabilidade do sistema.

Ao mesmo tempo, a escala e a heterogeneidade inerente a estes sistemas colocam tremendos desafios de engenharia [20]. Com isto, são necessárias novas abordagens tecnológicas para formalizar a sua conceção, geri-los e controlá-los de uma forma escalável, eficiente e segura, assegurando a sua usabilidade ao longo do tempo.

3.2.5 Digital Supply Chain

A cadeia de fornecimento digital pode ser definida como o desenvolvimento de sistemas de informação e a adoção de tecnologias inovadoras reforçando a integração e a agilidade da cadeia de fornecimento, melhorando assim o serviço ao cliente e o desempenho sustentável da própria organização [21].

A transformação digital utiliza tecnologias digitais para criar novos processos empresariais e oportunidades para enfrentar mudanças cada vez maiores. Muitas organizações utilizam tecnologias de transformação digital (Indústria 4.0) para melhorar a integração de todos os sistemas [22]. Os gestores da cadeia de abastecimento tentam reduzir os custos e desenvolver de forma mais ágil, sistemas flexíveis e conectados para aumentar o valor de mercado para satisfazer o consumidor em rápida mudança de comportamentos e volatilidade do mesmo.

A transformação digital ajuda as organizações a encurtar desenvolvimento de produtos, a aumentar a flexibilidade e a gerar inovação, ajudando as pessoas a permanecerem competitivas na era da Internet [23]. A transformação digital utiliza novas tecnologias na indústria para potenciar a indústria 4.0, conduzindo à sincronização da produção, evitando a distorção da informação, diminuindo prazos de entrega e aumentando a produtividade dos processos.

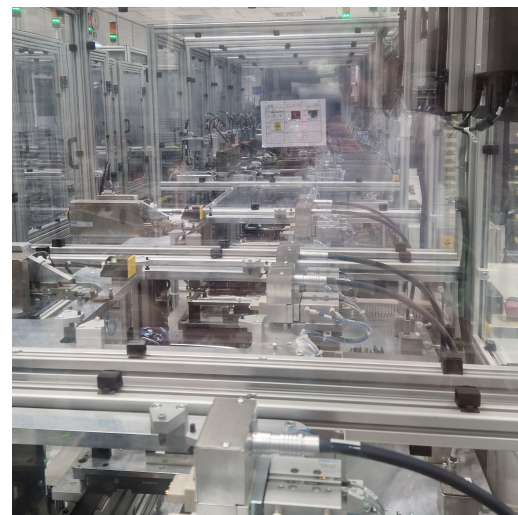
3.3 End-Of-Line

Testar uma unidade fabricada no final da linha de montagem é uma etapa crítica no processo de produção. Os produtos defeituosos ou mesmo aqueles que não correspondem aos limites de especificação devem ser separados das unidades funcionais, que serão expedidas para o cliente. Os testes de fim de linha avaliam não só a qualidade do produto, mas também a estabilidade e o rendimento do processo de produção. A detecção fiável de unidades não funcionais é o objetivo principal do teste, mas reduzir a taxa de rejeição e maximizar a produção é o objetivo final.

Os testes automáticos substituem cada vez mais testes subjetivos por operadores, de modo a encurtar o ciclo de produção e melhorar a reprodutibilidade e comparabilidade dos resultados. Contudo, as medições objetivas devem fornecer uma avaliação abrangente tão sensível como um testador humano utilizando os seus sentidos visuais e auditivos. Para competir plenamente com um operador experiente, os instrumentos de medição devem também ter capacidades de aprendizagem para acumular conhecimentos sobre as causas físicas da falha. Além disso, deve ser capaz de ser integrado em linhas automatizadas que sejam capazes de suportar ambientes complexos e ruidosos. A figura 3.4, apresenta fotografias da linha da EOL da empresa.



(a) Linha de EOL da PREH, vista fora.



(b) Linha de EOL da PREH, vista dentro.

Figura 3.4: Imagens da Linha EOL da PREH.

A variação dos parâmetros lineares e não lineares fornecidos pelos modelos físicos pode ser utilizada para a deteção de unidades defeituosas. Por exemplo, um aumento da massa em movimento (por exemplo, demasiada cola) ou uma diminuição do fator de força (por exemplo, magnetização insuficiente) reduz a saída de pressão sonora dos altifalantes, a frequências mais elevadas. A relação entre causas e efeitos torna-se mais complexa quando ocorrem falhas iniciais e consecutivas.

Os fabricantes devem recordar que encontrar as peças defeituosas não é o único objetivo final. O foco deve estar em como otimizar o processo de fabrico de modo a que as taxas de produção sem erros sejam tão altas quanto possível e o oposto para os custos de sucata e de retrabalho.

Confiar apenas em testes de fim de linha pode garantir que as peças expedidas não contenham erros, porém, não pode ser a única forma de detecção de erros, de modo a maximizar a eficácia produtiva.

3.4 *Data Analytics*

Hoje em dia, embora a tecnologia tenha evoluído consideravelmente, os analistas lidam com uma maior quantidade de dados e com bases de dados mais complexas e extensas, ou seja, apesar dos computadores atuais serem totalmente capazes de recuperar os dados com facilidade, a tarefa de os analisar tornou-se mais complicada, e este aumento de complexidade implica que seja necessário a existência de especialistas da área a lidar com os dados [24].

A análise de dados é o processo de analisar os dados no seu estado "bruto" a fim de extrair informações significativas. Esta sabedoria gerada é então utilizada para informar e conduzir a decisões inteligentes no mundo empresarial [25]. Deste modo, um analista extrai dados em formato bruto, organiza e analisa, transformando-os em informação coerente e inteligível. Tendo interpretado os dados, o analista de dados transmitirá as suas conclusões sob a forma de sugestões ou recomendações.

Com o intuito de encontrar padrões num conjunto de dados que possam transmitir utilidade e relevância sobre uma determinada área do negócio, a análise de dados ajuda a realizar uma aprendizagem sobre o passado e prever tendências e comportamentos futuros. Ao invés de basear as suas decisões e estratégias através do processo de tentativa erro, as decisões são implementadas com base em informação disponibilizada através dos dados passados.

O processo de análise de dados pode ser decomposto em quatro tipos:

- **Descritivo:** A análise descritiva é um tipo de análise superficial, que analisa o histórico dos acontecimentos. As duas principais técnicas utilizadas na análise descritiva são a agregação e a mineração dos dados. Assim, o analista reúne primeiro os dados e apresenta-os num formato resumido (que envolve a secção de agregação) e depois filtra-os para permitir descobrir padrões. É importante perceber que a análise descritiva não tenta explicar os dados passados ou estabelecer relações de causa e consequência.
- **Diagnóstico:** A fase de diagnóstico preocupa-se em saber o porquê da informação providenciada. Começa-se por procurar as anomalias inerentes nos dados, ou seja, qualquer acontecimento que não possa ser explicado pelos mesmos. Desta forma, os analistas iniciam a fase de descoberta, onde identificam todas as fontes de dados adicionais que lhes possam dizer mais sobre o porquê da ocorrência de anomalias terem surgido. Normalmente, o analista recorre a métodos probabilísticos, à análise regressiva e temporal dos dados e filtragem.

- **Preditivo:** a análise preditiva prevê o que é provável acontecer no futuro. Estima a probabilidade de um resultado futuro, com base nos dados passados, e também através das demonstrações probabilísticas. Embora nunca se possa garantir a assertividade nos resultados, pelo menos elimina grande parte da subjetividade quanto às principais decisões empresariais.
- **Prescritivo:** a análise prescritiva aconselha sobre as ações e decisões que devem ser tomadas. A análise prescritiva mostra como se pode tirar partido dos resultados previstos. Ao realizar uma análise prescritiva, os analistas consideram uma série de cenários possíveis e avaliam as diferentes ações que a empresa poderá tomar. Este tipo análise é um dos tipos mais complexos e pode envolver trabalho com algoritmos, ML e procedimentos de modelação computacional.

Data Mining (DM) [26] é o processo de exploração e análise de grandes blocos de informação envolvendo algoritmos de ML, métodos estatísticos e sistemas de armazenamento de dados, tendo como objetivo transformar os dados brutos numa estrutura compreensível e padronizada para uma utilização posterior e para facilitar a interpretação dos padrões subjacentes [24].

3.5 Machine Learning

Data Mining e *Machine Learning* são ambos subcampos da *Data Science*. Os dois conceitos são semelhantes, uma vez que são aplicados para resolver problemas complexos, com o uso somente dos dados recolhidos [27]. Utilizando ambos, o programador é capaz de extrair informação relevante de grandes conjuntos de dados. Esta informação é geralmente recolhida sob a forma de um padrão e disponibilizada graficamente ou em tabelas, que permite ao ser humano adquirir conhecimentos que, caso contrário, seria impossível de o obter, devido à dimensão dos mesmos. *Machine Learning* (ML) é um dos subgrupos da Inteligência Artificial que se foca na aprendizagem e na melhoria através dos dados fornecidos, ou seja, tem a capacidade de aprender e de melhorar a sua previsão de respostas através do input imposto, sem que seja programado especificamente para atingir ou alcançar uma solução.

O processo de aprendizagem começa com observações ou recolha de dados para procurar padrões nos dados e tomar as melhores decisões no futuro, com base nos exemplos recolhidos anteriormente. O objetivo principal é permitir aos computadores que aprendam automaticamente sem a intervenção humana.

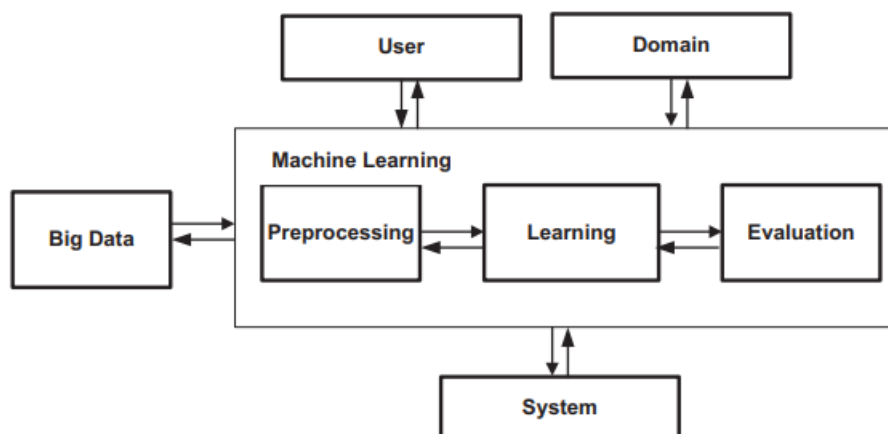


Figura 3.5: Processos para a implementação de Machine Learning.

Os algoritmos de ML [28] tipicamente costumam passar por três etapas: métodos de pré-processamento dos dados, fase de aprendizagem do algoritmo implementado e fase de avaliação dos resultados obtidos, como é demonstrado na 3.5. O pré-processamento dos dados é basicamente a transformação dos dados originais recolhidos em dados filtrados, em que a informação proveniente seja adequada para a interpretação do algoritmo. Os dados previamente recolhidos poderão apresentar uma estrutura não conveniente à interpretação, como ruído, dados inconsistentes e incompletos. Na fase de aprendizagem, procede-se à escolha do algoritmo e consequentes ajustes no mesmo, de modo a permitir a correta leitura e interpretação dos dados recolhidos para que se possa prever mais precisamente os resultados desejáveis. Por fim, na fase de avaliação verifica-se a performance do algoritmo no contexto, considerando a estimação do número de erros com base em testes estatísticos.

3.6 Algoritmos de Classificação e Regressão

Algoritmos de ML com aprendizagem supervisionada pode ser definidos em algoritmos de classificação e de regressão [?] que, respetivamente, identificam a que categoria pertence um objeto e prevêm uma saída contínua. Por vezes, existe uma linha ambígua entre algoritmos de classificação e algoritmos de regressão pois alguns algoritmos podem ser utilizados tanto para classificação como para regressão, ao mesmo tempo que a classificação é apenas um modelo de regressão com um limite aplicado quando o valor é superior ao limite definido, é classificado como verdadeiro, enquanto que um número inferior é classificado como falso.

3.6.1 Random Forest

O algoritmo *Random Forest* (RF), é um algoritmo eficiente e muito utilizado, baseado num conjunto de árvores de decisão, figura 3.6, geralmente treinadas com o método de *bagging* ("agrupar"), sendo aplicado tanto para problemas de classificação como de regressão, introduzido por Breiman em 2001 [29].

Este algoritmo começa por desenhar N árvores inicializadas com X amostras da base de dados e em cada árvore criada não são realizadas técnicas de *prunning*. Desta forma, durante a construção da árvore, as variáveis de entrada dessa amostra são escolhidas aleatoriamente e as melhores divisões são calculadas. Por último, em problemas de classificação, a previsão de novos inputs é realizada por métodos de votação que elegem a previsão mais frequente.

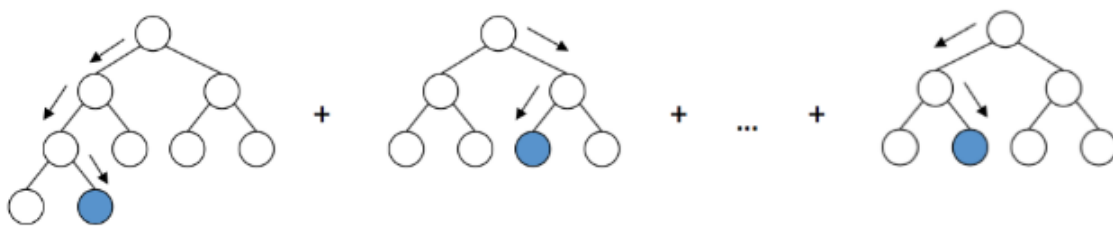


Figura 3.6: Diagrama do algoritmo *Random Forest*.

O autor [30] admite que o bom desempenho do algoritmo se deve à boa qualidade de construção de cada árvore e à existência de uma baixa correlação entre elas. A estimação da correlação entre as árvores é conseguida através das amostras não utilizadas na fase de construção das mesmas. Estas amostras são assim utilizadas para estimar o erro de previsão da solução de cada árvore e também para avaliar a importância que cada variável tem no conjunto dos dados [31].

3.6.2 Decision Tree

Este algoritmo tem como objetivo a construção de uma árvore a partir da base de dados de forma a maximizar a sua precisão. Esta classificação é realizada dividindo a árvore em conjuntos (*training*), onde cada nó representa um teste a um determinado atributo, as arestas indicam os valores ou intervalos de um nó e as folhas (nós com zero de altura ou os últimos nós originado das ramificações da árvore) são as classes e os caminhos que são originados da raiz até uma folha corresponde a uma regra de classificação [32].

Quanto à construção destas árvores, é realizada a partir de uma abordagem de "Dividir e Conquistar" (*Top-Down*). Assim sendo, este modelo funciona separando os dados do *training set* com base nos parâmetros que maximizam o ganho de informação. Cada subgrupo definido pela primeira ramificação, será então dividido consecutivamente e este processo é repetido até que os subgrupos formados não consigam ser mais divididos [33], como se pode verificar na figura 3.7.

Após esta ramificação de classes, procede-se à criação das regras com o objetivo de evitar o *overfitting*, onde cada caminho distinto da raiz até as folhas representa uma regra única. Os

antecedentes são desenvolvidos pelos resultados da função teste dado pela classe respetiva à folha em questão.

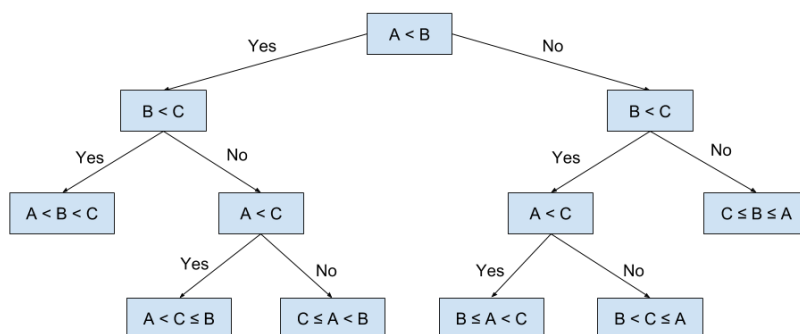


Figura 3.7: Diagrama do algoritmo *Decision Tree*.

Para cada conjunto de regras, é determinada uma taxa de erro e em seguida serão removidas, individualmente por ordem de utilidade, as regras que sejam menos úteis na resolução do problema. Averigua-se novamente a taxa de erro e verifica-se se houve alguma alteração.

Desta forma, este modelo reconstrói as regras com mais utilidade, de forma a introduzir informação da regra previamente eliminada. Assim, a árvore criada terá menos ramificações, sendo mais simples, compacta e perceptível pelas pessoas.

3.6.3 Naive Bayes

Naive Bayes é um algoritmo de classificação para problemas de classificação binária (duas classes) e também de multiclasse, utilizado para simplificar o processo de aprendizagem, assumindo que os atributos são independentes uns dos outros.

Ao fazer essa suposição, em que as classes não estão correlacionadas ou dependentes entre si, o algoritmo imposto requer menos tempo de treino. Assim, como os métodos de regressão logística, o modelo NB lida com probabilidades. O cálculo das probabilidades para cada hipótese é simplificado para o tornar mais simples, ou seja, em vez de tentar calcular os valores de cada atributo $P(d1, d2, d3)$, assume-se que são condicionalmente independentes dado o valor alvo e calculados como $P(d1|H) * P(d2|H) * P(d3|H)$ e assim sucessivamente.

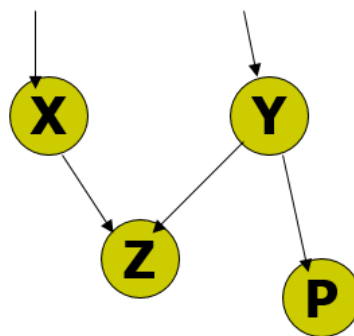


Figura 3.8: Diagrama do algoritmo NB.

Na dissertação de Mehdi Salehi [34], a inferência de *Bayes* foi aplicada como abordagem de modelação preditiva em vários métodos diferentes. Este algoritmo é considerado como tendo um bom desempenho neste ambiente onde não há elevadas incertezas relativamente ao processo de fabricação, como se pode confirmar na figura 3.8.

Noutro trabalho, I. Rish [35] propõe um estudo no qual o desempenho do algoritmo NB é analisado em vários cenários diferentes, demonstrando que geralmente este algoritmo é bem sucedido na tarefa de previsão, mesmo quando as probabilidades não são calculadas com precisão. Adicionalmente, o autor também demonstra que o algoritmo apresenta um bom desempenho ao lidar com recursos independentes.

De notar que, este algoritmo é bastante utilizado e, por isso, considera-se interessante verificar os resultados obtidos no contexto desta dissertação.

3.6.4 Logistic Regression

A regressão logística é uma técnica de classificação utilizada em ML e o seu funcionamento consiste em técnicas probabilísticas e de estatística. No fundo, esta regressão é um método estatístico para analisar um conjunto de dados no qual existem uma ou mais variáveis independentes que determinam o resultado previsto [36], através ao uso da equação 3.1 de regressão logística. A intenção por detrás da utilização da regressão logística é encontrar o modelo mais adequado para descrever a relação entre a variável dependente e a independente.

A regressão logística, apesar do seu nome, é um modelo de classificação e não um modelo de regressão, sendo um método simples e mais eficiente para problemas de classificação binária e linear, como representa o gráfico 3.9. Considerado um modelo de fácil implementação, atinge um desempenho muito bom com classes linearmente independentes, sendo amplamente usado para a classificação na indústria, podendo generalizado à classificação multiclasse [37].

$$\text{Logistic Function} = \frac{1}{1 + e^{-x}} \quad (3.1)$$

A principal diferença entre regressão linear e regressão logística é que o intervalo da regressão logística é limitado entre 0 e 1. Além disso, em oposição à regressão linear, a regressão logística

não requer uma relação linear entre variáveis de entrada e de saída. Isto deve-se ao uso de uma transformação logarítmica.

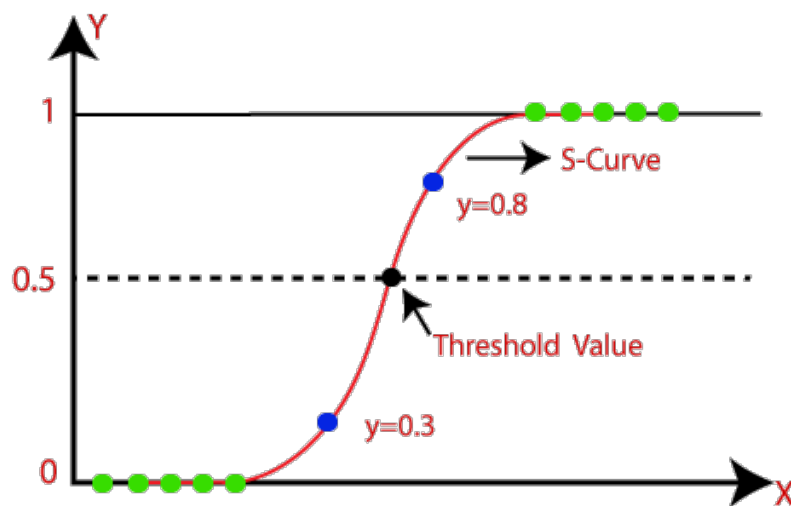


Figura 3.9: Gráfico representativo do sistema de classificação utilizando um algoritmo regressão logística.

3.6.5 Support Vector Machine

"Support Vector Machine"(SVM) é um algoritmo de aprendizagem supervisionada de máquinas que pode ser usado tanto para projetos de classificação como de regressão. No entanto, é sobretudo aplicado em problemas de classificação [38].

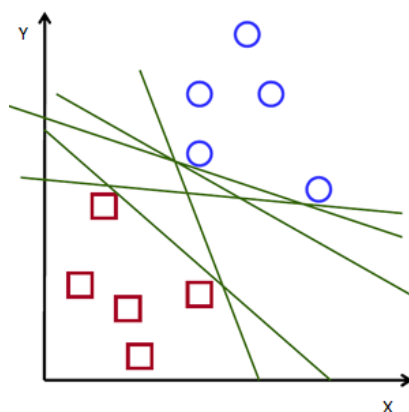


Figura 3.10: Gráfico representativo do início do processo de classificação do algoritmo SVM

Para possibilitar a separação das duas classes a partir dos dados, são criados hiperplanos possíveis de serem escolhidos, descrito no gráfico 3.10. O objetivo deste algoritmo passa por encontrar um plano que tenha a margem máxima, ou seja, a distância máxima entre as coordenadas dos

pontos de ambas as classes. Ao maximizar a distância de margem, proporciona credibilidade para futuros dados que possam ser classificados com mais confiança (figura 3.11).

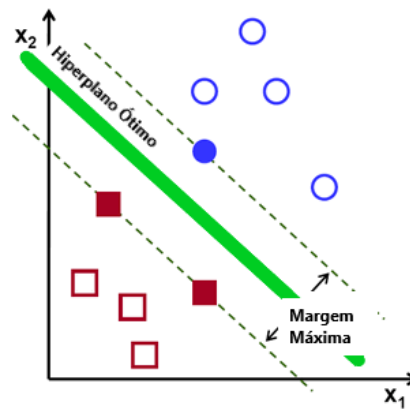


Figura 3.11: Gráfico representativo das margens criadas com o algoritmo SVM.

Na regressão logística, a saída da função linear é compreendida dentro do intervalo $[0,1]$, com a utilização da função sigmoide se o valor for superior a um valor *threshold* de 0.5, atribui-se valor 1 ou 0. Em SVM, caso a saída da função linear seja superior a 1, é atribuída uma classe x e se a saída for -1, é definida uma classe y . Uma vez que os valores *threshold* são definidos para -1 e 1, obtém-se este intervalo que atua como margem.

Capítulo 4

Método Implementado - Base de Dados

A maioria das empresas industriais já tem alguma experiência na recolha de dados. Normalmente, registam alguns dados sobre o processo de produção ou montagem, tais como o início e duração do ciclo de produção, a quantidade de matérias utilizadas e de produtos produzidos. Isto permite às empresas obter uma visão geral da sua performance em todo o seu horário de funcionamento, destacando inconsistências na produção, através do desvio padrão de certos valores de padrão de funcionamento.

A recolha destes dados nem sempre é fácil devido a várias razões: algumas máquinas e/ou tecnologias mais antigas podem não estar equipadas para fornecer os dados necessários; As máquinas mais recentes podem ter a capacidade de o fazer, mas muitas vezes apenas num formato prioritário do fabricante; Noutros casos, os dados recolhidos dos processos de produção e montagem são retidos apenas durante o processo e já não estão disponíveis para análise posterior. Isto resulta em ilhas desconectadas de dados temporariamente disponíveis, o que significa que os gestores de produção são incapazes de obter uma visão global do processo de produção e são, portanto, incapazes de analisar a fonte dos problemas.

Outro obstáculo pode ser uma infra-estrutura informática desatualizada. As bases de dados desatualizadas não conseguem armazenar grandes volumes de dados de produção e não permitem a consulta e o processamento analítico rápido e eficiente de tais dados. Se uma empresa quiser ter dados disponíveis para a melhoria da qualidade da produção, manutenção preditiva e gestão de recursos, precisa de estar dotada de uma arquitetura tecnológica universal que possa recolher e armazenar dados de diferentes sistemas.

Uma arquitetura de base de dados implementada, pode ser utilizada para recolher e analisar dados do processo de produção (tais como temperatura, pressão, taxas de fluxo de líquido ou gás, vibração e consumo de energia) que podem ter um grande impacto na qualidade e durabilidade do produto. Ao mesmo tempo, pode ser utilizada para prever falhas de máquinas; tornar a manutenção mais eficiente; otimizar o consumo de energia, gás, ar comprimido e água; melhorar a segurança industrial.

4.1 Organização da Base Dados

A base de dados da PREH é desenvolvida num sistema banco de dados da Microsoft, SQL Server Management Studio. A base de dados da empresa, quanto à sua organização, está dividida por produtos desenvolvidos, gestão pessoal, materiais, dados EOL, registo de todos os usuários, entre outros.

Cada base de dados, mais concretamente a base de dados em que incidiu esta dissertação "*BMW BZM AUTO*", recolhe todas as informações e dados de toda a linha de produção deste produto, dividido em múltiplas tabelas para melhor organização.

Além da possibilidade de realizar *queries* para apresentar os dados pretendidos através do filtro de várias tabelas, na PREH a utilização de *Store Procedures* é recorrente quando se pretende automatizar as *queries* para serem utilizadas pelas aplicações embutidas nas máquinas ou computadores, como é o caso da aplicação mais utilizada na empresa, que permite visualizar o histórico da peça. Este processo é altamente benéfico no contexto de uma empresa de manufatura pois permite alterar o teor das *queries* sem ter de mudar o código da respetiva aplicação, permitindo alterar todas as aplicações numa só vez. As alterações são realizadas através do SQL Server.

4.1.1 Análise das Tabelas e das Variáveis

No contexto da dissertação, o objetivo foi apurar os dados desde que a peça obtém um erro até que a mesma é reparada. Para tal, foi necessário transformar os dados das várias tabelas para permitir obter o erro proveniente juntamente com o retrabalho realizado.

As tabelas em estudo da base de dados da BMW BZM foram:

- **dbo_DeviceErrorResults** Nesta tabela são registados todos os erros provenientes da linha EOL e da estação da qualidade. Além de registar o erro, são também guardados os valores dos parâmetros reprovados, a máquina onde ocorreu a reprovação, a descrição do erro, entre outros. Neste contexto, as variáveis mais relevantes são:
 - **ID_DeviceErrorResult:** Regista em formato inteiro, um valor único sequencial respetivo ao erro e peça realizada. Por exemplo, o valor da variável relativo à próxima peça nesta linha em que se detete um erro será o valor do anterior mais um. Por outras palavras, esta coluna é o index da tabela;
 - **ID_WSJob:** Nesta variável é registado, em formato inteiro, um valor único sequencial em que, a cada trabalho realizado em qualquer máquina na linha da BMW BZM, o valor da variável é incrementado em mais um valor. Basicamente, regista todos os trabalhos realizados em todas as máquinas da respetiva linha do produto;
 - **ID_DeviceResultTestCharacteristic:** Nesta variável é registado o primeiro segmento do erro produzido. Guarda um valor numérico categórico, referente ao tipo de teste que foi realizado.
 - **ID_DeviceResultTestObject:** Nesta variável é registado o segundo segmento do erro produzido. Guarda um valor numérico categórico, referente à localização da peça onde o tipo de teste realizado reprovou;

- **TraceNr:** Nesta variável é registado em formato inteiro, um valor único sequencial em que cada peça obtém um valor único, permitindo rastreabilidade da peça;
- **TextValue:** Nesta variável, fica registado em formato *string*, o registo do erro das peças provenientes da estação qualidade.¹
- **dbo.WSJobs:** Nesta tabela são registados todos os trabalhos feitos pelas máquinas desta linha. A variável ID_WSjob referida anteriormente provém desta tabela. As variáveis mais relevantes são:
 - **ID_WSJob:** Descrito anteriormente, na tabela dbo_DeviceErrorResults;
 - **TraceNr:** Descrito anteriormente, na tabela dbo_DeviceErrorResults;
 - **JobStartTime e JobEndTime:** Nestas variáveis é registado o começo e o fim do trabalho da máquina, correspondente à peça com o respetivo TraceNr.
- **dbo.WsSubDevices:** Nesta tabela são registadas todas as máquinas da linha conforme a função que tem.
 - **Station Group:** Nesta variável fica registado o tipo de estação a que cada máquina pertence;
 - **ID_SubWS:** Nesta variável fica registado, em formato inteiro, o número único correspondente a cada máquina;
 - **SubWSName:** Nestas variáveis é registado a função de cada respetiva máquina;
- **dbo.Samples_TraceNrs:** Nesta tabela, estão registados as peças de teste da linha, ou seja, são as peças que servem de calibração das linhas, principalmente da linha EOL.²
 - **TraceNr:** Descrito anteriormente, na tabela dbo_DeviceErrorResults;
 - **ID_Sample:** Variável onde se regista em formato numérico e único o número da peça sample respetiva;
 - **Error_Code:** Esta variável é responsável de guardar o erro característico ou recorrente da peça *sample* respetiva;
- **Rework.ReworkResults:** Nesta tabela, ficam registados todos os retrabalhos realizados, sendo as seguintes variáveis as mais relevantes:
 - **ID_WSJob:** Descrito anteriormente, na tabela dbo_DeviceErrorResults;

¹ Ao contrário, das variáveis ID_DeviceResultTestCharacteristic e ID_DeviceResultTestObject que registam o erro em código das peças provenientes da estação EOL. No caso da estação de qualidade, estas variáveis são carimbadas com um código categórico referente à estação, ID_DeviceResultTestCharacteristic é 6273 e ID_DeviceResultTestObject é 603 e a identificação do erro é contido no TextValue, que é realizado por um técnico da estação.

² Para a *query* originada, estes dados serviram para retirar os erros originados provenientes de peça de amostras. Cada uma destas peças, apresentam erros específicos para as máquinas detetarem. Deste modo, a utilização dos erros destas peças causaria possivelmente ruído na obtenção dos dados;

- **ID_SubWS:** Descrito anteriormente, na tabela `dbo.WsSubDevices`;
 - **TraceNr:** Descrito anteriormente, na tabela `dbo_DeviceErrorResults`;
 - **ID_Rework:** Nesta variável fica registado, em formato inteiro categórico, o número do retrabalho correspondente;
 - **GroupName:** Nesta variável fica registado em que módulo foi realizado o retrabalho à peça respetiva;
 - **ID_ReworkDiagnostic:** Nesta variável fica registado o diagnóstico do operador para cada peça;
 - **Defect Type:** Fica registado que tipo de defeito ocorreu, funcional ou cosmético;
 - **Result Description:** Nesta variável fica registada toda a informação referente ao retrabalho da peça, em formato texto;
-
- **Rework.Reworks:** Nesta tabela ficam registados os nomes específicos de cada retrabalho, como também a estação para qual tem de ser enviadas as peças após a realização do respetivo retrabalho
-
- **ID_Rework:** Descrito anteriormente, na tabela `dbo_DeviceErrorResults`;
 - **ReworkName:** Variável que regista o nome do retrabalho utilizado;
 - **ID_Sub_ReleaseStation:** Esta variável é responsável de guardar, consoante o retrabalho, o ID da estação para onde a peça tem que ser transferida após o respetivo retrabalho.

Na figura 4.1 foram esquematizadas as tabelas, com as respetivas variáveis e as suas relações entre elas.



Figura 4.1: Diagrama UML das relações da base de dados.

4.2 Processamento de Dados

Na fase de processamento dos dados é necessário ter conhecimento profundo dos processos inerentes que tenham um impacto nos dados que se deseja obter. Este conhecimento não pode ficar limitado a conhecer bem a base de dados e as possíveis relações entre tabelas mas também deve ser alongado até onde os dados são produzidos. Isto é, perceber os ínfimos fatores que possam influenciar o trabalho na linha, ou por *bugs* nos softwares das máquinas ou por falha do operador ao não seguir convenientemente os procedimentos de trabalho. Estes fatores podem desvirtualizar ligeiramente os resultados recolhidos. Limitar-se unicamente na base de dados, em certos casos, não é suficiente para se conseguir detetar desvios mínimos nos dados, principalmente quando são produzidos nesta linha cerca de duas mil e quinhentas peças por dia e são retrabalhadas cerca de

quatrocentas.

No contexto da dissertação, o processamento de dados foi vital para se conseguir obter recomendações fidedignas acerca do retrabalho a realizar consoante o erro da peça. Neste modo, antes de remover os *outliers* dos dados, é necessário determinar primeiro que formato de dados se deseja recolher. Para este objetivo, é necessário apenas recolher os erros provenientes das colunas ID_DeviceResultTestCharacteristic³, ID_DeviceResultTestObject⁴ e TextValue, juntamente com o retrabalho realizado, da coluna ID_Rework, como foi exemplificado na tabela 4.1.

Test_Characteristic	Test_Object	Erro	ID_Rework	Rework_Name
44	1155	44 - 1155	24	GWS - Trocar Blenda

Tabela 4.1: Formato dos dados após processamento.

Além de se realizar uma boa estruturação dos dados, é necessário filtrar ou remover aqueles que possam estar errados e influenciar negativamente os retrabalhos a recomendar. Deste modo, delinearam-se regras na recolha dos dados, sendo as seguintes:

- Uma peça que chegue à estação de retrabalhos e que através do histórico apresente vários erros diferentes, o retrabalho realizado corresponde ao último erro diagnosticado (figura 4.2);

JobID / LastJob	Data/Hora	Ação	Resultado	Código Erro	Detalhe
205467742	07-06-2022 13:29	EOL2 8 - Vision	OK		
205467915	07-06-2022 13:30	EOL2 9c - Gear Shift Calibration	E 2	3.6.7609.498	KeysMechanical/PushButton/Snap (2nd)/GWS NA_R: [19.5%]
205470814	07-06-2022 13:41	EOL2 9b - Gear Shift Calibration	E 2	11.34.6215.662	System/Software/GearShift/CheckPosition_VV: (Esperado: V\
205494780	07-06-2022 15:14	Rework Station	OK		
205494781	07-06-2022 15:14	Analysis: Released to EOL 0.0 - Haptic Feedbac...			
205498783	07-06-2022 15:30	EOL4 0.0 - Ext Base	OK		
205498783	07-06-2022 15:30	---> Config.File: 90170_148_0032 - Version (Ma...			
205498920	07-06-2022 15:30	EOL4 0.1 - Acoustic	OK		
205514750	07-06-2022 16:33	EOL 1.0 - Pin/Screw Test	OK		
205514819	07-06-2022 16:33	EOL 1.1 - Input	OK		
205514895	07-06-2022 16:34	EOL 2 - RunIn	OK		
205514990	07-06-2022 16:34	EOL 3a - Keys Test	OK		
205515324	07-06-2022 16:35	EOL 4 - FBM Test	OK		
205515410	07-06-2022 16:36	EOL 5 - Gap Measurement	OK		
205515507	07-06-2022 16:36	EOL 4a - Keys Test	OK		
205515608	07-06-2022 16:37	EOL 4b - Keys Test	OK		
205516104	07-06-2022 16:39	EOL 6 - Laser	OK		
205516222	07-06-2022 16:39	EOL 7 - Illumination Calib.	OK		
205516338	07-06-2022 16:40	EOL 7a - Illumination Measurement	OK		
205516521	07-06-2022 16:41	EOL 8 - Vision	OK		
205516679	07-06-2022 16:41	EOL 9c - Gear Shift Calibration	E 2	3.6.7609.498	KeysMechanical/PushButton/Snap (2nd)/GWS NA_R: [19.5%]
205755836	08-06-2022 07:54	Rework Station	OK		

Figura 4.2: Histórico da peça - Erros diferentes, retrabalho considera o último

- Se uma peça, após o retrabalho, conter retrabalhos seguidos⁵ e/ou retrabalhos que não foram realizados convenientemente ou que o registo na base dados não tenha sido realizado com sucesso, estes casos não entram para o *dataset* final (figuras 4.3 e 4.4));

³Representa a mesma variável que o Test_Characteristic

⁴Representa a mesma variável que o Test_Object

⁵Nestes casos não se sabe qual dos dois retrabalhos está certo.

JobID / LastJob	Data/Hora	Ação	Resultado	Código Erro	Detalhe
201774972	24-05-2022 12:47	EOL2 5 - Gap Measurement	OK		
201775072	24-05-2022 12:47	EOL2 4a - Keys Test	E 2	3.6.7935.542	KeysMechanical/PushButton/Electrical Contact C5/Unlock;
201798104	24-05-2022 14:33	Rework Station	E 1		
201798143	24-05-2022 14:33	Rework Station	E 1		
201798238	24-05-2022 14:33	Rework Station	E 1		
201798298	24-05-2022 14:34	Rework Station	E 1		
201798406	24-05-2022 14:34	Rework Station	E 1		
201798518	24-05-2022 14:35	Rework Station	E 1		
201798578	24-05-2022 14:35	Rework Station	E 1		
201798726	24-05-2022 14:35	Rework Station	E 1		
201799531	24-05-2022 14:39	Rework Station	E 1		
201710544	24-05-2022 14:42	Disassembled child: 00300167145695 by Eduar...			13031-445/0005 - BG GWS Normal LHD B2M
201800540	24-05-2022 14:43	Rework Station	OK		

Figura 4.3: Histórico da peça - Retrabalhos mal sucedidos (a vermelho)

JobID / LastJob	Data/Hora	Ação	Resultado	Código Erro	Detalhe
175574703	31-01-2022 04:05	EOL 6 - Laser	OK		
175574840	31-01-2022 04:05	EOL 7 - Illumination Calib.	E 2	6.13.42.144	Vision/Luminance/LDM_BeforeCalib/Led MSA: [1cd/m2] < [0cd/m
175577038	31-01-2022 04:14	EOL 7 - Illumination Calib.	E 2	6.13.42.144	Vision/Luminance/LDM_BeforeCalib/Led MSA: [1cd/m2] < [0cd/m
175577123	31-01-2022 04:14	Analysis: Blocked by ID:0			
175584543	31-01-2022 04:47	Rework Station	OK		
175584544	31-01-2022 04:47	Analysis: Released to EOL 0.0 - Haptic Feedbac...			
175605611	31-01-2022 06:34	Rework Station	OK		

Figura 4.4: Histórico da peça - Retrabalhos bem sucedidos sem erro intercalado

- Uma peça para entrar para o dataset tem que apresentar um erro e um retrabalho, obrigatoriamente. A mesma peça pode entrar várias vezes para o *dataset* caso esta reprove várias vezes na linha EOL (figura 4.5);

JobID / LastJob	Data/Hora	Ação	Resultado	Código Erro	Detalhe
172859491	20-01-2022 15:39	EOL 10 - Tubus	OK		
172859713	20-01-2022 15:39	EOL 11 - Extraction	E 2	8.23.50.179	Software/EEPROM/EEPROM_READ/GetDTCInformation: [DTC: 0
172896894	20-01-2022 17:51	Rework Station	OK		
172896897	20-01-2022 17:51	Analysis: Released to EOL 0.0 - Haptic Feedbac...			
172897377	20-01-2022 17:52	Muro Volante	OK		
172897956	20-01-2022 17:55	EOL3 0.0 - Memory Clean	E 2	11.34.7942.822	System/Software/UUT/Error Field Block - No Description known:
172898127	20-01-2022 17:55	EOL3 0.0 - Memory Clean	E 2	11.34.7942.822	System/Software/UUT/Error Field Block - No Description known:
172898157	20-01-2022 17:55	Analysis: Blocked by ID:0			
172919235	20-01-2022 18:54	Rework Station	OK		
172919237	20-01-2022 18:54	Analysis: Released to EOL 0.0 - Haptic Feedbac...			
172923324	20-01-2022 19:04	Muro Volante	OK		
172923924	20-01-2022 19:06	EOL3 0.0 - Memory Clean	E 2	11.34.7942.822	System/Software/UUT/Error Field Block - No Description known:
173118311	21-01-2022 06:57	Rework Station	OK		
173118325	21-01-2022 06:57	Analysis: Released to OP280 - GWS & Carrier A...			

Figura 4.5: Histórico da peça - Exemplo de casos bem sucedidos de erros e retrabalhos

Outro fator a ter em consideração ao realizar o processamento dos dados é a distinção entre as peças provenientes da linha EOL e da estação de qualidade. Esta necessidade em distinguir entre os dois tipos de erros advém da diferente estruturação do código de erro. No caso da estação de qualidade o erro aparece no formato em texto/string (ver anexo C.1), enquanto nos caso dos erros provenientes da linha EOL são em formato inteiro (ver anexo C.2). Para os conseguir distinguir, as peças que chegam à estação qualidade provenientes da linha de EOL e que são submetidas a uma análise mais aprofundada, o código do erro de EOL (ID_DeviceResultTestCharacteristic e ID_DeviceResultTestObject) é alterado para os valores 6273 e 603, respetivamente. Este código simboliza que a peça foi examinada e o erro respetivo à falha da peça é acrescentado na coluna TextValue. As outras peças com erros provenientes da linha EOL, tem neste campo o valor *NULL* (figura 4.6).

JobID / LastJob	Data/Hora	Ação	Resultado	Código Erro	Detalhe
176849336	04-02-2022 00:35	EOL2 7 - Illumination Calib.	OK		
176849840	04-02-2022 00:37	EOL2 7a - Illumination Measurement	OK		
176850019	04-02-2022 00:38	EOL2 8 - Vision	OK		
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.4.498	KeysMechanical/PushButton/F1/GWS NA_R: [14.5N] < [0N] < [2
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.6184.498	KeysMechanical/PushButton/F7/GWS NA_R: [1N] < [0N] < [999
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.6186.498	KeysMechanical/PushButton/S5/GWS NA_R: [10mm] < [7.1mm
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.7608.498	KeysMechanical/PushButton/Snap (1st)/GWS NA_R: [19.5%] <
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.85.498	KeysMechanical/PushButton/S2/GWS NA_R: [4mm] < [0mm] <
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.87.498	KeysMechanical/PushButton/F3/GWS NA_R: [23.5N] < [17.5N] <
176850159	04-02-2022 00:38	EOL2 9 - Gear Shift Calibration	E 2	3.6.91.498	KeysMechanical/PushButton/S4/GWS NA_R: [9.5mm] < [0mm]
176853838	04-02-2022 00:51	EOL2 9a - Gear Shift Calibration	OK		
176854489	04-02-2022 00:54	EOL2 10 - Tubus	OK		
176854576	04-02-2022 00:54	EOL2 11 - Extraction	OK		
176854576	04-02-2022 00:54	====> Config File: 90170_616_0030 - Version (Sa...			
176864754	04-02-2022 01:31	Muro Qualidade 2.2	E 2	1.1.6273.603	Operator/Evaluations/QualityWall/CheckParts: [Auto Blenda GW
176884804	04-02-2022 02:44	Rework Station	OK		

Figura 4.6: Histórico da Peça - Erro EOL examinado pela estação qualidade

4.3 Query Desenvolvida

Para se conseguir criar um sistema de recomendações é necessário ter dados corretos num formato adequado, como também é necessário que haja um histórico para se conseguir retirar uma padronização e informação sobre os dados.

Tendo como objetivo recomendar os três melhores retrabalhos para o erro em questão é necessário organizar e agrupar os dados, com base na frequência de cada retrabalho para cada erro distinto, assim como o prévio processamento de dados referido no capítulo anterior. O código desenvolvido esta disponibilizado na figura 4.7.

Table_Count_Reworks AS

```

(SELECT Test_Characteristic, Test_Object, ID_Rework, Count(Error_job) AS County
FROM Final_Process
WHERE NOT (Test_Characteristic=6273 AND Test_Object=603)
GROUP BY Test_characteristic, Test_object, ID_Rework
),Show_Top3 AS
(SELECT Test_Characteristic, Test_Object, ID_Rework, County
FROM (SELECT *, Row_number() OVER (PARTITION BY Test_Characteristic, Test_Object ORDER BY County DESC)
FROM Table_Count_Reworks) T2
WHERE Seqnum <= 3)

```

Figura 4.7: Excerto de código da Query não otimizada

Uma vez que o sistema implementado na linha possui duas tipologias diferentes quanto à amostragem do erro, houve necessidade de implementar duas *queries* diferentes. Estas partilham o mesmo pré-processamento de dados, porém nos erros provenientes da estação qualidade (6273 - 603), é necessário recolher o erro, em formato texto, da coluna TextValue. Em anexo, constam as duas versões das *queries* A.1 desenvolvidas.

Na PREH, os softwares que correm em todas as linhas de produção são desenvolvidos e otimizados pelo departamentos Software da PREH, por isso é necessário recorrer a metodologias que promovam, quanto possível, a automatização das aplicações para o ambiente em que estas irão ser inseridas. Uma vez que todas as máquinas estão interligadas com o sistema, possuem um ID único

e portam periféricos diferentes, não era vantajoso alterar cada uma das máquinas os ficheiros de configuração e as *queries* de obtenção dos dados, inseridas no código da aplicação.

Desta forma, o recurso a *Store Procedures* é fulcral para permitir realizar pequenos ajustes na forma em que todas as aplicações recolhem os dados a partindo apenas da base de dados. Assim sendo neste sistema de recomendações, a *query* originada foi transformada para um sistema de *Store Procedures* A.2, que recebe o *TraceNr* e devolve os três melhores retrabalhos.

Um dos problemas inerentes em empresas onde existem linhas de alto rendimento de produção, é a tentativa de minimizar ao máximo os tempos de execução de todos os processos da linha, de modo a que a empresa seja mais produtiva. Neste contexto, com a introdução do sistema de recomendações na aplicação, a *query* demora cerca de dez segundos a devolver os resultados das recomendações. Na prática, este tempo implica que o operador terá de aguardar dez segundos sempre que necessite de realizar um retrabalho, o que não é viável.

Por forma a viabilizar a execução deste projeto, foi necessário aplicar métodos de otimização de *queries* para reduzir o tempo de execução da mesma assim como do CPU e dos recursos alocados para alcançar os resultados, como, por exemplo, tabelas ou quantidades exageradas de instâncias recolhidas⁶. O desrespeito destes procedimentos pode implicar no mau funcionamento dos registos e das leituras da base de dados. A utilização extensiva de *queries* pode provocar impedimento na comunicação das aplicações com as respetivas tabelas, não permitindo escrever novos dados. Fenómenos como estes provocam paragens totais no funcionamento da linha de produção.

4.4 Implementação do Sistema de Recomendações na Aplicação

Durante o processo de implementação do sistema de recomendações na aplicação de retrabalho, foi necessário alterar a aplicação criando uma nova página e reestruturando o processo condutor da aplicação, assim como desenvolver o sistema de recomendações, que foi explorado neste capítulo.

Anteriormente, a estrutura da aplicação apresentava o ambiente inicial que aguardava enquanto o operador lia o QR code e dava a escolher entre a realização de um retrabalho ou de um trabalho de laser⁷. Após o operador iniciar a opção de retrabalho, a aplicação redirecionava-o para o modo manual, dado que esta era a única opção disponível. Após a realização do retrabalho, este regressava para o ambiente inicial, onde aguarda pela próxima peça (figura 4.8).

⁶Neste contexto a variável *ResultDate*, consoante o filtro temporal aplicado, é diretamente responsável pelo número de amostras recolhidas

⁷Esta opção, apesar de existir na aplicação de retrabalho, não foi alvo de qualquer alteração

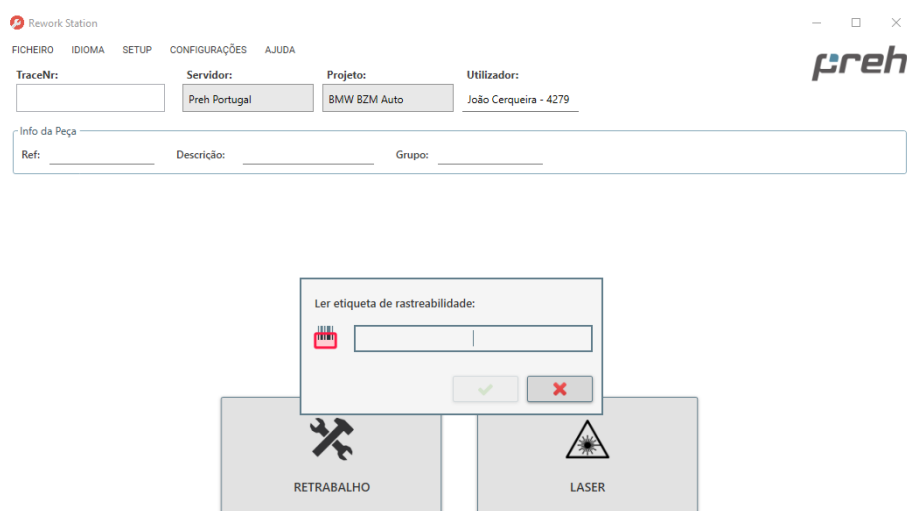
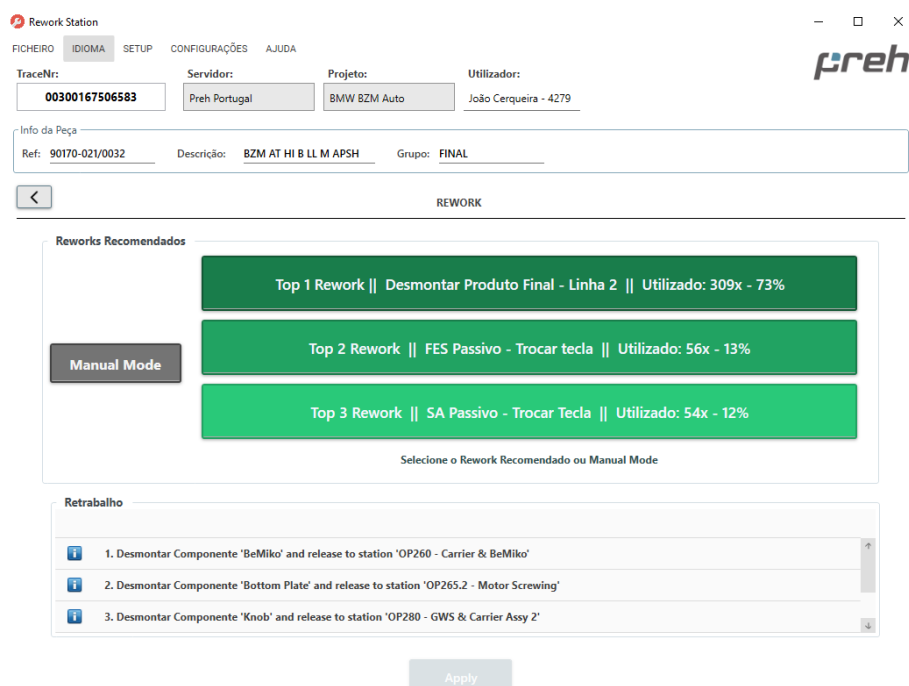


Figura 4.8: Ambiente Inicial da Aplicação

A nova versão da aplicação mantém o mesmo formato das diferentes vistas ou estados da aplicação. Foi acrescentada uma página de recomendações que fica disponível após a leitura do QR code e que apresenta as recomendações de retrabalhos, caso estejam disponíveis. No caso de não se verificarem recomendações disponíveis o operador é redirecionado para o modo manual. Nos anexos B.2, poderá ser consultado o *FlowChart* da implementação.

Outra condição imposta após a leitura do TraceNr é verificar se o erro proveniente já ocorreu mais que cinco vezes desde a sua origem. Sabendo que o conhecimento só está prevalecido nos técnicos da estação de retrabalhos, se estes inicialmente não realizam uma avaliação minuciosa do erro para escolherem o retrabalho mais adequado, não é possível realizar recomendações para esse erro de forma fidedigna. Desta forma, optou-se que para um erro ser recomendado este tem de ser avaliado manualmente cinco vezes antes que o sistema possa aconselhar retrabalhos. Assim sendo, evita-se ou diminui-se a probabilidade de existirem retrabalhos menos adequados para os novos erros provenientes da linha EOL.

A página de recomendações apresenta três opções de recomendação, organizadas por top três de retrabalhos mais implementados para o erro em questão. Cada uma das opções descreve o nome do retrabalho, o número de vezes que o retrabalho foi utilizado para o mesmo erro e a percentagem de utilização. Encontra-se também disponível a opção de modo manual caso o operador não considere adequada nenhuma das recomendações disponibilizadas (figura 4.9).



^aPode-se observar que quando um retrabalho não foi selecionado, o botão de aplicar fica bloqueado.

Figura 4.9: Página de Recomendação da Aplicação ^a

Optando por uma das recomendações, a aplicação apresenta as instruções de trabalho que devem ser implementadas e indica em que máquina da linha de produção é que a peça irá ser introduzida, dependendo do retrabalho escolhido. As instruções aparecem em modo pdf, no lado direito da aplicação, que poderão ser desativadas nos ficheiros de configuração da aplicação (figura 4.10).

A níveis funcionais foram criadas algumas verificações de forma que o operador não se engane a selecionar o retrabalho ou que realize um retrabalho sem o ter selecionado. Desta forma, foi criada uma etiqueta que está colocada a seguir aos botões de retrabalho, que disponibiliza o ultimo retrabalho selecionado na aplicação. Assim, o operador sabe qual é o retrabalho que está selecionado. Foi também programado o botão de “aplicar” para ficar bloqueado até ser selecionado um dos retrabalhos propostos, para evitar que o operador aplique um retrabalho que não tenha previamente selecionado.

Num sistema, onde se pretende diminuir a variância e o erro na seleção dos retrabalhos, é preciso ter em consideração todos os fatores que possam induzir o operador em erro. Inicialmente implementou-se um sistema de semáforo, onde o botão do primeiro retrabalho teria uma cor verde, o segundo amarelo e por ultimo vermelho. Todavia, num sistema de recomendações que se pretende recomendar os melhores retrabalhos existentes, onde poderá existir mais que um retrabalho possível para cada erro, não seria coerente representa-los de forma em que exista retrabalhos que teriam a cor vermelha ou amarela, uma vez que neste sistema de recomendações, a posição relativa da recomendação é com base na frequência do retrabalho respetivo. Implementar um sistema em

semáforo, pode induzir o operador em erro na seleção do retrabalho, visto que a cor verde, num código de cores em semáforo, representa o que está correto e o vermelho o que está errado.

Desta forma, alterou-se para um estilo dégradé, onde as cores não influenciam nem retém mais informação do que o próprio sistema de recomendação.

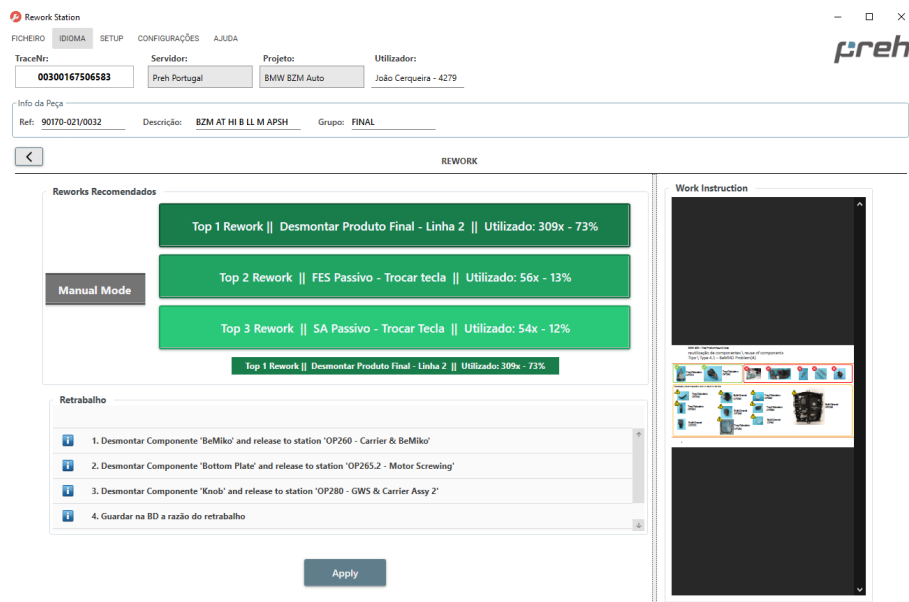


Figura 4.10: Ambiente de Recomendações da Aplicação com Instruções

Capítulo 5

Prova de Conceito - Machine learning

Neste capítulo será descrito o sistema de recomendação com a implementação em *Machine Learning*. Serão explicados os métodos utilizados para a preparação dos dados e para o treino do modelo, bem como a sua implementação. Por último, será tratado o método de avaliação desenvolvido que avalia quanto à sua assertividade nas recomendações.

Face à globalidade deste projeto e ao estarmos perante um problema de dados, verificou-se que seria interessante solucionar o problema com base em algoritmos supervisionados de classificação. Sabendo que, para cada erro poderão existir mais que dois retrabalhos diferentes, encontramos-nos perante um problema de multi-classificação.

Durante esta fase, recorreu-se a duas linguagens que são utilizadas para projetos de ML, *Python* [39] com recurso às bibliotecas *Pandas* e *Sklearn* e *C#* com recurso a biblioteca *ML.NET* [40] [41]. Inicialmente recorreu-se à linguagem *Python*, por ser a linguagem com mais suporte a bibliotecas de ML e pela facilidade na sua implementação. Porém, para realizar uma boa integração com as aplicações desenvolvidas e respeitar o ambiente no qual a aplicação irá ser executada, optou-se por realizar a implementação dos algoritmos de ML em *C#*. Isto permite realizar o treino do modelo e obter as recomendações sem que seja necessário que o modelo esteja num servidor ou num ficheiro de *script* à parte.

Por outro lado, apesar de que o desenvolvimento em *Python* [42] simplifica a manipulação dos dados e o treino dos modelos de ML para os programadores, comparativamente ao *C#* apresenta velocidades inferiores de treino do modelo.

5.1 Variáveis Categóricas

Na fase de formatação dos dados, para permitir uma aprendizagem correta do modelo ML, é necessário identificar corretamente o tipo de variável a recolher, definir quais são as variáveis de entrada e de saída do modelo e realizar procedimentos adicionais de codificação das variáveis para ser possível realizar o treino do modelo.

Durante este processo, identificaram-se as variáveis de entrada do sistema, que neste caso são compostas por duas colunas com valores do tipo inteiro, `ID_DeviceResultTestCharacteristic` e `ID_DeviceResultTestObject`. Desta forma, foi necessário realizar uma transformação destes dados, uma vez que os algoritmos de classificação interpretavam as variáveis inteiras como sendo valores contínuos e não como variáveis categóricas. Esta transformação envolve transformar as colunas para formato *string*, como também realizar a junção dos dois erros produzidos, (exemplo 4567 - 343). Assim sendo, em vez de termos duas variáveis de entrada, temos só uma que representa a totalidade do erro, evitando perder informação ao realizar esta ação para este caso.

Apesar que a maioria dos algoritmos de ML funcionam apenas com valores numéricos, existem também muitas variáveis importantes que não são numéricas mas sim categóricas. As variáveis do tipo categóricas podem assumir níveis/grupos ou valores, tais como a idade, estado civil ou tipo de cliente de uma empresa. Alternativamente, estes podem ser criados através da combinação de características numéricas subjacentes, tais como o agrupamento das faixas etárias (por exemplo, 0-10, 11-18, 19-30, 30-30, 30-50)

Uma vez que estas características categóricas não podem ser utilizadas diretamente na maioria dos algoritmos de aprendizagem de máquinas, as características categóricas precisam de ser transformadas em características numéricas. Com a utilização *One Hot Encoding* uma variável categórica é convertida num conjunto de indicadores binários ¹ (pode ser observado na figura 5.1). Assim, numa categoria que contenha os estados do tempo, por exemplo: sol aberto, parcialmente nublado, a chover, vento, neve, nublado ou nevoeiro; serão criadas sete novas variáveis que somente contenham os valores 0 ou 1. Depois, para cada observação, a variável que corresponde à categoria será definida como 1 e todas as outras variáveis definidas como 0.

```
var dataProcessPipeline = mlContext.Transforms.Categorical
    .OneHotEncoding(@"Erro", inputColumnName: nameof(ErroData.Erro))
    .Append(mlContext.Transforms.Concatenate(@"Features", @"Erro"))
    .Append(mlContext.Transforms.Conversion.MapValueToKey(@"Label", nameof(ErroData.Label)))
    .Append(mlContext.Transforms.NormalizeMinMax(@"Features", @"Features"));
```

Figura 5.1: Excerto de código relativo ao processamento de dados

Em relação aos resultados preditivos do modelo treinado, uma vez que a nossa saída é um *ID* em formato inteiro, os próprios algoritmos identificam-no como um conjunto de valores contínuos. Caso não se realize a transformação para formato de texto, o modelo poderá recomendar um *ID* compreendido entre o maior e o menor valor da variável *ID_Rework*, mesmo que este não exista.

¹Um por categoria em todo o conjunto de dados

Em formato de texto, impõem-se ao modelo que intérprete cada *ID* como um valor único e discreto, só podendo recomendar *ID*'s que existam nos dados introduzidos.

5.2 Treino do Modelo

Na fase de desenvolvimento do treino do modelo realizou-se a escolha entre os algoritmos de Classificação binária, ou de multiclasse ou de Regressão.

Dado que este projeto envolve um problema de multiclasse, para que os algoritmos binários assim como os algoritmos de regressão consigam interpretar dados em que existam mais que duas classes diferentes, foi necessário a utilização de métodos heurísticos que permitam que os algoritmos binários consigam realizar o treino convenientemente, como por exemplo, métodos como OVA *One-Vs-All* [43] [44] que envolve a divisão do conjunto de dados de multiclasse em múltiplos problemas de classificação binária. Cada classificador binário é treinado em cada problema de classificação e o resultado dessas previsões é com base no modelo mais confiante (figura 5.2). Para bases de dados extensas ou com inúmeras classes, estas heurísticas prejudicam consideravelmente o tempo de treino, uma vez que requer a criação de um modelo para cada classe.

```
var trainer= mlContext.MulticlassClassification.Trainers
    .OneVersusAll(mlContext.BinaryClassification.Trainers
    .LbfgsLogisticRegression(LabelColumnName: @"Label", featureColumnName: @"Features"))
    .Append(mlContext.Transforms.Conversion.MapKeyToValue("PredictedLabel", "PredictedLabel"));
```

Figura 5.2: Excerto de código relativo ao processamento de dados

A implementação de modelos ML para a produção a fim de realizar inferências, ou seja, prever resultados em novas instâncias de dados, costuma levar a muitas complicações devido aos dados de produção não serem semelhantes aos dados de treino. Caso o formato do erro seja ligeiramente diferente, o modelo não consegue relacionar aos retrabalhos que aprendeu anteriormente, identificando o erro como diferente.

De outro modo, existem outros contratempos a nível de hardware ou de software, que por vezes implica utilizar outras tecnologias para permitir alcançar o mesmo fim. Sendo essencial, numa fase inicial, realizar um planeamento estruturado de todas as fases inerentes à criação de um modelo preditivo de ML.

Nesta dissertação, a implementação do modelo em C# simplificou este processo, referido anteriormente. Definiu-se que o treino do modelo irá se realizar sempre que a aplicação é executada. Denotou-se que a aplicação demorava mais tempo a iniciar, devido ao treino do modelo. Em termos funcionais, respeitando o ambiente das linhas produção, verificou-se que não causava quase nenhum transtorno uma vez que as aplicações só são novamente executadas no começo de cada turno. Deste modo, consegue-se obter recomendações de erros que sejam relativamente novos.

5.3 Algoritmo Desenvolvido

Para se conseguir realizar a avaliação da acurácia do modelo quanto à exatidão das recomendações, é necessário recorrer às bibliotecas dedicadas a ML, para possibilitar o cálculo da precisão do modelo. Estes métodos de avaliação possuem várias ferramentas de análise, que a partir do teste de novas instâncias de erros, conseguem avaliar a sua performance. Quanto maior for o grupo de instâncias de teste, maior é a confiança atribuída à performance do modelo. Porém, estes métodos de avaliação só permitem testar quanto à exatidão do melhor retrabalho para o erro correspondente e não para os três melhores retrabalhos.

Deste modo, para que se possa garantir que o modelo esteja a obter uma boa performance comparativamente ao método colocado em produção, foi necessário desenvolver um algoritmo próprio. Este algoritmo de avaliação está composto por dois algoritmos, um de pertença que avalia quanto à existência do retrabalho e outro de ordenação que avalia quanto à posição correta do retrabalho atribuído.

Relativamente ao algoritmo de pertença, este compara os três retrabalhos sugeridos com os três retrabalhos mais realizados, recolhidos através da query referido no capítulo anterior. Por cada retrabalho erradamente sugerido, o algoritmo penaliza com um terço da cotação, sabendo que a escala de avaliação insere-se no intervalo entre zero e um (figura 5.3).

```
public static (decimal, int) AvaliateExistence(Get_All_Results a, Get_All_Results b)
{
    int missingValues = 0;
    List<string> reworkB = new List<string>();

    reworkB.Add(b.Rework_1);
    reworkB.Add(b.Rework_2);
    reworkB.Add(b.Rework_3);

    if (!reworkB.Contains(a.Rework_1)) missingValues++;
    if (!reworkB.Contains(a.Rework_2)) missingValues++;
    if (!reworkB.Contains(a.Rework_3)) missingValues++;

    if (missingValues >= 3) return (0, 3);

    return (decimal.Parse((1 - (0.333333 * missingValues)).ToString()), missingValues);
}
```

Figura 5.3: Excerto de código relativo ao algoritmo de pertença

Quanto ao algoritmo de ordenação, este avalia a recomendação consoante a posição, penalizando dinamicamente dependendo da distância à correta posição do retrabalho suposto. Caso o modelo recomende um retrabalho que esteja numa posição consecutiva à posição correta, este é pontuado com um sexto, no caso de mais do que uma posição não é pontuado. Se acertar a posição do retrabalho é pontuado com um terço, até perfazer o valor máximo de um, numa situação ideal (figura 5.3).

```

public static decimal AvaliateOrdenation(Get_All_Results a, Get_All_Results b)
{
    double ordernationDiscount = 0;
    List<string> reworkB = new List<string>();

    reworkB.Add(b.Rework_1);
    reworkB.Add(b.Rework_2);
    reworkB.Add(b.Rework_3);

    if (a.Rework_1 == b.Rework_1)
        ordernationDiscount = ordernationDiscount + 0.33333;

    if (a.Rework_1 == b.Rework_2)
        ordernationDiscount = ordernationDiscount + 0.16667;

    if(a.Rework_2 == b.Rework_2)
        ordernationDiscount = ordernationDiscount + 0.33333;

    if(a.Rework_2==b.Rework_1 || a.Rework_2 ==b.Rework_3)
        ordernationDiscount = ordernationDiscount + 0.16667;

    if (a.Rework_3 == b.Rework_3)
        ordernationDiscount = ordernationDiscount + 0.33333;

    if (a.Rework_3 == b.Rework_2)
        ordernationDiscount = ordernationDiscount + 0.16667;

    return Convert.ToDecimal(ordernationDiscount);
}

```

Figura 5.4: Excerto do código relativo ao algoritmo de ordenação

No contexto geral, estes algoritmos, de pertinência e de ordenação, têm pesos diferentes para a pontuação final (figura 5.5). Sabendo que se trata de um sistema de recomendações, considera-se mais grave que um retrabalho seja mal recomendado do que a própria posição do retrabalho estar correta. Por outro lado, seria incorreto classificar quanto à correta ordenação dos retrabalhos quando estes não chegam a ser recomendados, acabando sempre por obter uma má performance. Por esse motivo, os pesos são de setenta por cento para o algoritmos de pertinência e de trinta por cento para o algoritmo de ordenação.

```

public static decimal Avaliation(Get_All_Results a, Get_All_Results b)
{
    var output = AvaliateExistence(a, b); //output item2 is missingvalue quantity; output item 1
                                         //is percentage to be used for avaliation

    if (output.Item2 == 3) return 0;

    decimal percentageExistence = 0.7m + (0.1m * output.Item2);
    decimal percentageOrdenation = 0.3m - (0.1m * output.Item2);

    return (percentageExistence * output.Item1) + (percentageOrdenation * AvaliateOrdenation(a, b));
}

```

Figura 5.5: Excerto do código relativo algoritmo de avaliação

Porém estes pesos alteram-se dinamicamente consoante o número de retrabalhos corretamente recomendados. A cada retrabalho errado é acrescentado dez por cento ao peso do algoritmo de

pertença e é retirado o mesmo valor ao peso do algoritmo de ordenação, de forma a contrariar o problema anteriormente referido.

5.4 Resultados Gerais

Nesta fase, foram criados vários modelos utilizando algoritmos diferentes. Foram utilizados algoritmos suportados para multiclasse assim como algoritmos binários e de regressão com auxílio a métodos heurísticos. Quanto aos algoritmos implementados, utilizou-se os algoritmos disponibilizados na biblioteca de ML.NET que obtiveram melhor performance nos testes realizados.

Os pesos destes parâmetros, na globalidade deste *dataset*, é de noventa por cento e de dez por cento, respetivamente.

5.4.1 Análise Estatística dos Dados

Nomeadamente nesta linha de produção em estudo, a recolha de dados foi realizada a partir do dia um de janeiro de dois mil e vinte e dois. O principal motivo desta escolha deveu-se às irregularidades que poderiam ocorrer nos dados envoltos aos retrabalhos das peças, anteriores a esta data.

Durante a fase de análise dos dados, um dos problemas existentes no *dataset* é que apresenta dados desbalanceados [45] quanto às classes utilizadas ou retrabalhos utilizados. Dados desequilibrados referem-se tipicamente a um problema de classificação em que as classes não são representadas de forma igual. Por exemplo, num problema de classificação de duas classes (binário) com cem instâncias. No total, oitenta instâncias são rotuladas com a Classe A e as restantes vinte instâncias são rotuladas com a Classe B. Origina um conjunto de dados desequilibrado e o rácio de instâncias da Classe A para a Classe B é de 80:20 ou mais precisamente 4:1.

Quando se trabalha com um problema de classificação com dados desequilibrados, (anexos C.3 C.4) a classe minoritária é tipicamente a que tem mais interesse. Isto significa que a habilidade de um modelo em prever corretamente a etiqueta ou probabilidade da classe minoritária é mais importante do que a classe ou classes maioritárias.

A classe minoritária é mais imprevisível porque por definição existem poucos exemplos. Isto significa que é mais difícil para um modelo aprender as características dos exemplos desta classe e diferenciar os exemplos desta classe da classe maioritária (ou classes). A abundância de exemplos da classe maioritária (ou classes) pode inundar a classe minoritária.

A maioria dos algoritmos de ML para modelos de previsão de classificação são concebidos e demonstrados em problemas que assumem uma distribuição igual das classes. Isto significa que um modelo não otimizado, pode concentrar-se apenas na aprendizagem das características das observações abundantes, negligenciando os exemplos da classe minoritária, que em certos casos, é de maior interesse e cujas previsões são mais valiosas e mais difíceis de obter corretamente.

Indicadores	EOL Dataset	QWALL Dataset
Classes Min	1	1
Classes Max	12	20
Classes Média	2,67	5,24
Classes Diferentes	37	35
Erros diferentes	1071	165
Total Instancias	25170	20029

Tabela 5.1: Tabela representativa das características dos *datasets* em estudo

Neste contexto, existe uma grande predominância de certos retrabalhos, porém, devido ao método de seleção com base na frequência, elege os retrabalhos que mais vezes foram realizados. Desta forma, promove o desbalanceamento dos dados e a utilização de ferramentas que combatem este tipo de problemas, desvirtuando os resultados obtidos.

Através da tabela 5.1, pode-se constatar que existem erros onde existe uma grande variedade de retrabalhos diferentes a serem realizados. Esta desigualdade é um dos fatores principais para a diminuição das métricas de performance dos algoritmos.

Com base no que foi referido, a tabela 5.2 demonstra a desigualdade existente nos *datasets*, onde cada linha representa um erro diferente e cada coluna um retrabalho diferente. O valor da célula representa o número de ocorrências do retrabalho para o erro respectivo.

ID_Rework	51	37	36	35	32	26	25	24	22	21	19	18	17	16	15	14	12	10	9	8	7	6	5	4	3	2	Total Por Erro
21 - 4		3						2	8				2	1			9		124	3	20	1	1				174
22 - 36								1	4			1	11				8	4	145	2	23				1		200
3447 - 380									105										25								130
37 - 384								1	161										8								170
4 - 27									12							1											13
43 - 144		1			1			1	47				2				3		30						1	5	91
46 - 11	1	18			5	1		3	201			1		3	1		14	1	57		1	12					319
6 - 788	9	7	2		2			9	141	1	2	1	8	7	1		61	5	339		3	102				2	702
6453 - 249		64	7	4	19	2		16	391	1	7	1	16	22			308	2	400		1	197					1458
7942 - 822	2	38		14	10		14	8	643	10		1	1	1			10		93	2	6	22					875
Total Por Retrabalho	12	131	9	18	37	3	14	41	1713	12	9	5	40	34	2	1	413	12	1221	2	16	376	1	1	1	8	4132

Tabela 5.2: Recolha de alguns erros relativos que demonstra a desigualdade de classes presente nos *datasets*

Com o apoio desta tabela, é possível constatar a desigualdade de classes presente nos *datasets* em estudo, como também a variedade de retrabalhos utilizados para o mesmo. O objetivo do pré-processamento referido neste projeto foi tentar minimizar esta variedade de retrabalhos utilizados para a resolução de um erro.

5.4.2 Resultado das Recomendações dos Algoritmos

Neste sub-capítulo serão referidos os algoritmos implementados como também os resultados obtidos nos diferentes parâmetros de avaliação, para cada algoritmo utilizado para o efeito pretendido.

- **Algoritmos de Multiclasse**

- **Naive Bayes:** é um classificador probabilístico que pode ser utilizado para problemas multiclasse. Utilizando o teorema de Bayes, a probabilidade condicional para uma amostra pertencente a uma classe pode ser calculada com base na contagem de amostras para cada grupo que contenha as mesmas características.
- **Lbfgs Maximum Entropy:** fornece um critério construtivo para estabelecer distribuições de probabilidade com base no conhecimento parcial e conduz a um tipo de inferência estatística que é definida como a estimativa de entropia máxima [46]. É a estimativa menos tendenciosa possível sobre a informação dada;
- **SdcaMaximumEntropy:** é uma variação da regressão logística que se baseia no método *Stochastic Dual Coordinate Ascent* (SDCA).
- **Light GBM:** (*Gradient Boosting Machine*) [47] utiliza duas técnicas, nomeadamente a amostragem com base no gradiente e no agrupamento de características exclusivas. No primeiro caso, exclui a parte significativa de instâncias de dados com pequenos gradientes e utiliza apenas os dados restantes para estimar o ganho de informação. No segundo, agrega características mutuamente exclusivas. Com estas duas alterações, acelera o tempo de treino do algoritmo em vinte vezes, comparativamente aos algoritmos de *Gradient Boosting Machine*

- **Algoritmos de Classificação Binária**

- **Fast Forest:** é uma implementação de um algoritmo muito convencional RF. Consiste num conjunto de árvores de decisão. Cada árvore produz uma distribuição gaussiana através de uma previsão. É realizada uma agregação sobre o conjunto de árvores criadas, para permitir encontrar uma distribuição gaussiana mais próxima da distribuição combinada.
- **Fast Tree:** é uma implementação eficiente do MART (*Multiple Additive Regression Trees*. *Gradient Boosting*) é uma técnica de ML para problemas de classificação e de regressão. Constrói cada árvore de regressão de forma gradual, usando uma função de perda predefinida para medir o erro para cada passo e corrige-o no passo seguinte. Portanto, este modelo de previsão é na realidade um conjunto de modelos de previsão mais fracos.
- **Linear SVM (*Support Vector Machine*):** observa cada vetor característico como um ponto num espaço adimensional. No seu núcleo, este algoritmo coloca todos os vetores

de características num gráfico imaginário n-dimensional e desenha uma linha imaginária n-dimensional (um hiperplano) que separa exemplos com etiquetas positivas de exemplos com etiquetas negativas em caso de classificação, ou recolhe o máximo de amostras possíveis em caso de regressão.

- **Algoritmos de Regressão Logística**

- **Lbfgs Logistic Regression:** é basicamente um algoritmo de classificação e de regressão. Num problema de classificação, a saída alvo pode tomar apenas valores discretos para um dado conjunto de características(ou entradas), X . O modelo constrói um modelo de regressão para prever a probabilidade de uma dada entrada de dados pertencer à categoria indicada. Tal como a regressão linear assume que os dados seguem uma função linear, a regressão logística modela os dados usando a função sigmoide.

Quanto à avaliação dos algoritmos, estes dispõem de ferramentas de avaliação. Desta forma, serão brevemente descritas as quatro medidas de avaliação utilizadas:

- **Micro-Accuracy:** agrega as contribuições de todas as classes para calcular a métrica média. É a fração de instâncias previstas corretamente; Quanto mais perto da 1.00 melhor. Num projeto de classificação multiclasse, este parâmetro é preferível ao *Macro-Accuracy* se se suspeitar que possa haver um desequilíbrio de classes.
- **Macro-Accuracy:** representa a precisão média ao nível da classe. A precisão para cada classe é calculada e a *Macro-Accuracy* é a média destas precisões. Basicamente, cada classe contribui igualmente para a métrica da precisão. As classes minoritárias têm o mesmo peso que as classes maiores. Quanto mais perto da 1.00, melhor. Calcula a métrica independentemente para cada classe e depois realiza a média dos valores obtidos.
- **Class Log-Loss:** a perda logarítmica mede o desempenho de um modelo de classificação onde a previsão de entrada é um valor de probabilidade entre 0,00 e 1,00. A perda logarítmica aumenta à medida que a probabilidade prevista diverge da etiqueta real. Quanto mais perto de 0,00, melhor. Um modelo perfeito teria uma perda de registo de 0,00. O objetivo dos modelos de ML é minimizar este valor.
- **Top-k Accuracy:** representa o valor preditivo do modelo para os melhores resultados.

Com base nestes indicadores, os resultados da performance destes algoritmos serão demonstrados nas tabelas seguintes. As colunas mais representativas da performance real dos algoritmos, são as medidas de avaliação *Micro-Accuracy* e o *Top-k Accuracy*.

Os testes foram realizados com o *dataset* repartido em oitenta por cento para treino e vinte por cento para teste, sendo realizado uma desordem nas linhas das tabelas para diminuir a existência de erros nos dados de teste que não existam nos dados treino. Visto que os dados estão ordenados por tempo de ocorrência, durante a repartição do *dataset*, pode ocorrer que um erro tenha somente

ocorrido muitas vezes numa semana e que quase não tenha ocorrido noutra altura temporal, criando um ambiente de aprendizagem enganador para o algoritmo e para as métricas de avaliação, prejudicando a sua performance.

Na tabela, as células que contenham a cor verde representam os algoritmos que obtiveram os melhores valores nesse parâmetro do mesmo *dataset* e o contrário acontece para a cor vermelha.

- Tabela 5.3 com os resultados dos algoritmos com o *dataset* QWALL

Modelo	QWALL DataSet			
	Micro Accuracy	Macro Accuracy	Per Class Loss	TOP-3 Accuracy
OVA - Fast Forest	0.7314	0.2918	0.5532	0.8718
OVA - Fast Tree	0.7399	0.2758	0.5006	0.8909
OVA - LbfgsLogisticRegression	0.7409	0.2698	0.5368	0.8924
OVA - LinearSvm	0.7314	0.3417	0.6956	0.8415
Multi Class - NaiveBayes	0.5804	0.0935	34.5388 *	0.7136
Multi Class - LbfgsMaximumEntropy	0.7416	0.2683	0.5779	0.8932
Multi Class - LightGbm	0.7407	0.2704	0.5495	0.8894
Multi Class - SdcaMaximumEntropy	0.7410	0.3033	0.5546	0.9034

Tabela 5.3: Tabela representativa dos resultados dos algoritmos utilizados com os dados dos erros provenientes da estação de qualidade

Nota (*): O valor referente ao algoritmos NB, subjetivo ao parâmetro Per Class Loss, pode apresentar um erro inerente ao código das bibliotecas das métricas de avaliação.

- Tabela 5.4 com os resultados dos algoritmos com o *dataset* EOL

Modelo	EOL DataSet			
	Micro Accuracy	Macro Accuracy	Per Class Loss	TOP-3 Accuracy
OVA - Fast Forest	0.5225	0.1147	0.7469	0.8416
OVA - Fast Tree	0.5778	0.1326	0.6158	0.8578
OVA - LbfgsLogisticRegression	0.5810	0.1049	0.6455	0.8555
OVA - LinearSvm	0.5743	0.1333	0.7126	0.8315
Multi Class - NaiveBayes	0.5460	0.0461	34.5388	0.8031
Multi Class - LbfgsMaximumEntropy	0.5811	0.1025	0.6367	0.8551
Multi Class - LightGbm	0.5782	0.0956	0.6383	0.8445
Multi Class - SdcaMaximumEntropy	0.5795	0.1530	0.6483	0.8580

Tabela 5.4: Tabela representativa dos resultados dos algoritmos utilizados com os dados dos erros provenientes da linha de EOL

A próxima tabela 5.5, demonstra os tempos de treino para cada algoritmo e para cada *dataset*. Denota-se que os algoritmos preparados para problemas de multi-classe alcançam treinos mais rápidos. Os algoritmos que consistem em árvores de decisão alcançam sempre os piores resultados, visto que não são ótimos para problemas com muitas instâncias nem para problemas com classes diferentes.

Pode-se constatar que para o *dataset* de EOL onde apresenta mais erros únicos, porém apresenta mais vinte por cento de instâncias relativamente ao *dataset* QWALL, os tempos de treino são quase o dobro relativamente aos dados QWALL. Isto acontece que para cada erro, os algoritmos à

base de árvores de decisão necessitam de criar mais nós, estendendo o comprimento da árvore que origina um aumento considerável nos tempos de treino.

No caso dos algoritmos com tempo de treino mais rápido é o NB, que sendo um algoritmo com base em heurísticas probabilísticas, consegue realizar a computação do treino mais rápido que outros algoritmos que funcionam através de métodos mais complexos. Contudo é o algoritmo que obteve os piores resultados de performance, em ambos os *datasets*.

Modelo	Time	
	QWALL (m:s:ms)	EOL (m:s:ms)
OVA - Fast Forest	00:12.6	00:23.5
OVA - Fast Tree	00:13.7	00:24.5
OVA - LbfgsLogisticRegression	00:02.7	00:02.3
OVA - LinearSvm	00:02.6	00:02.1
Multi Class - NaiveBayes	00:00.9	00:01.0
Multi Class - LbfgsMaximumEntropy	00:02.8	00:02.9
Multi Class - LightGbm	00:03.0	00:02.2
Multi Class - SdcaMaximumEntropy	00:05.4	00:05.6

Tabela 5.5: Tabela representativa dos tempos de treino de todos os algoritmos utilizados, para cada tipo de erro

Quanto à performance dos algoritmos (figura 5.6) quando submetido ao algoritmo de avaliação desenvolvido e referido no capítulo 5.3, obtiveram resultados de forma geral semelhantes aos resultados dos testes anteriores. Os algoritmos que se destacaram nas métricas anteriores destacam-se da mesma forma e proporção nesta avaliação. Por outro, o algoritmo NB apresenta resultados muito abaixo relativamente aos outros algoritmos classificadores. Como referido anteriormente, a biblioteca da ML.NET pode não estar otimizada ou existir um bug no código.

Modelo	QWALL DataSet	EOL Dataset	Total
OVA - Fast Forest	0.5803	0.6400	0.6102
OVA - Fast Tree	0.8319	0.7587	0.7953
OVA - LbfgsLogisticRegression	0.7095	0.7069	0.7082
OVA - LinearSvm	0.5075	0.6625	0.5850
Multi Class - NaiveBayes	0.3633	0.5995	0.4814
Multi Class - LbfgsMaximumEntropy	0.6963	0.7123	0.7043
Multi Class - LightGbm	0.7586	0.6957	0.7271
Multi Class - SdcaMaximumEntropy	0.8946	0.8100	0.8523

Tabela 5.6: Tabela representativa dos resultados dos algoritmos utilizados com recurso ao algoritmo de avaliação desenvolvido

Nesta tabela 5.7, decidiu-se extrair os resultados dos dados utilizando somente o algoritmo de Pertença desenvolvido, para somente classificar quanto há existência de retrabalhos corretos.

Comparativamente à avaliação global destes algoritmos obteve-se resultados ligeiramente superiores. Demonstra que para os algoritmos a tarefa de recomendar os retrabalhos corretos é mais facilmente alcançável do que posicioná-los na posição correta.

Modelo	QWALL DataSet	EOL Dataset	Total
OVA - Fast Forest	0.6063	0.6824	0.6444
OVA - Fast Tree	0.8413	0.7796	0.8104
OVA - LbfgsLogisticRegression	0.7238	0.7370	0.7304
OVA - LinearSvm	0.5302	0.6935	0.6118
Multi Class - NaiveBayes	0.3873	0.6278	0.5075
Multi Class - LbfgsMaximumEntropy	0.7111	0.7417	0.7264
Multi Class - LightGbm	0.7683	0.7167	0.7425
Multi Class - SdcaMaximumEntropy	0.9270	0.8370	0.8820

Tabela 5.7: Tabela representativa dos resultados do algoritmos de Pertença perante os algoritmos utilizados

5.5 Implementação do Modelo na Aplicação

Na fase de implementação do sistema de recomendações utilizando ferramentas de ML, recorreu-se à linguagem C# para facilitar esta integração. Inicialmente, a linguagem implementada foi Python, porém, a fase de integração de ambos os sistemas tornava-se mais difícil e utilizava mais recursos.

A aplicação não só recomenda com base das previsões do modelo criado como também treina com, possivelmente, novos dados, quando a aplicação executa e inicia o processo de recolha de dados à respetiva base, antes que esta entre no ambiente inicial da aplicação de retrabalhos. Nos anexos B.1, poderá ser consultado o *FlowChart* da implementação.

A recolha dos dados pré-processados acontece a partir de um ficheiro CSV, que diariamente poderá ser substituído com as novas instâncias, ou uma alternativa melhor, era cada vez que a aplicação iniciava, ir buscar diretamente os dados de treino à base de dados. Esta abordagem não foi implementada porque não havia necessidade visto que a aplicação não ia para produção.

Na fase de realizar as recomendações (figura 5.6), após a inserção do TraceNr, realiza-se uma query para permitir ir buscar o erro da peça. Seguidamente, realiza-se uma transformação no formato do erro para coincidir com o formato dos dados de treino. No caso dos erros EOL, agrupar as duas componentes do erro separando com dois hífen (2354 – 345) e para os erros QWALL remover os caracteres latinos que podem comprometer o reconhecimento do erro (ç, í, ã), mesmo que sejam aplicados codificadores, como UTF 8.

```

IEnumerable<KeyValuePair<string, decimal>> GetTopReworks()
{
    var mlContext = new MLContext(seed: 23);
    DataViewSchema modelSchema;

    ITransformer trainedModel = mlContext.Model.Load(ModelRelativePathQWall, out modelSchema);

    PredictionEngine<ErroData, Prediction> predictionEngine = mlContext.Model.CreatePredictionEngine<ErroData, Prediction>(trainedModel);

    Prediction _predictions = predictionEngine.Predict(inputData);

    var labelBuffer = new VBuffer<ReadOnlyMemory<char>>();
    predictionEngine.OutputSchema["Score"].Annotations.GetValue("SlotNames", ref labelBuffer);
    var labels = labelBuffer.DenseValues().Select(l => l.ToString()).ToArray();

    var index = Array.IndexOf(labels, _predictions.PredictedLabel);
    var score = _predictions.Score[index];

    var top10scores = labels.ToDictionary(l => l, l => (decimal)_predictions.Score[Array.IndexOf(labels, l)]).OrderByDescending(kv => kv.Value).Take(3);

    return top10scores;
}
top3Reworks = GetTopReworks();

foreach (var prediction in top3Reworks)
{
    Top3ReworksID_list.Add(int.Parse(prediction.Key.ToString()));
    Top3ReworksScore_list.Add(prediction.Value);
}

```

Figura 5.6: Demonstração da Aplicação de ML

Ao contrário da aplicação em produção, cada retrabalho apresenta exclusivamente a pontuação da previsão desse retrabalho, em vez de apresentar o número de ocorrência e a percentagem de uso (figura 5.7).

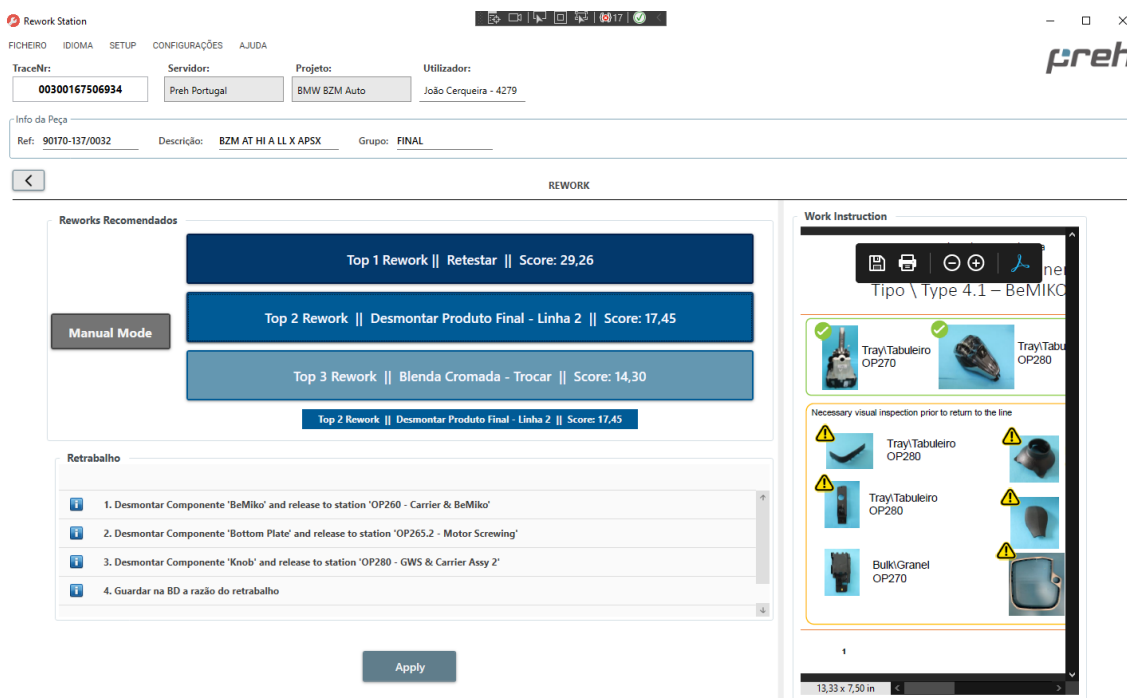


Figura 5.7: Demonstração da Aplicação de ML

Capítulo 6

Conclusões Finais e Trabalhos Futuros

A presente dissertação permitiu retirar um conjunto de conclusões sobre o projeto desenvolvido. Inicialmente, foi possível compreender de que modo as ferramentas de Inteligência Artificial, mais concretamente de DM e de ML, podem ser aplicadas nas linhas de produção, com o surgimento da Indústria 4.0, permitindo uma movimentação no sentido de uma manutenção dos produtos de forma preditiva e prescritiva. Durante o trabalho, foram concebidas duas aplicações no mesmo formato de ambiente e aplicados de forma diferente os métodos desenvolvidos no software de retrabalhos, verificando os diferentes constrangimentos dos mesmos, segundo as restrições das linhas de produção.

O difícil equilíbrio entre intervenção humana e automação é mais facilmente alcançado se os sistemas estiverem equipados com modelos fáceis de interpretar e entender. No entanto, a falta de transparência do processo de decisão inerente a muitos dos modelos de ML poderá ser dúbio e portanto, o desconforto que possa advir deste facto é facilmente ultrapassado, por exemplo, com o uso de *queries* que filtram os dados dos *datasets* que permitam obter a mesma informação por processos lógicos e restritivos. O problema instala-se quando a informação necessária recolhida pelas *queries* é muito complexa e extensa, impondo um aumento não só nos tempos de execução, como nos recursos alocados de hardware e de software, necessitando de ferramentas como ML para alcançar esses fins, por métodos preditivos.

O conjunto de dados utilizado é chave para a resolução de problemas de ML. Uma escolha sensata de quais os dados utilizar e como processá-los de forma a corresponder ao problema é crucial para melhorar o desempenho dos algoritmos. A divisão dos dados em conjuntos de treino, desenvolvimento e teste é fundamental para que a performance real dos sistemas de ML não seja inferior às expectativas geradas por uma avaliação incorreta dos resultados dos modelos, como também dar avaliações otimistas do estado do mesmo.

Com a presente dissertação, o sistema implementado em produção encontra-se totalmente funcional em todas as estações da linha desde o início de junho de 2022. O sistema de recomendações recomenda diariamente, em média, trezentos e cinquenta retrabalhos, cujos resultados são bastante satisfatórios no que concerne tanto ao aumento da assertividade na escolha do retrabalho como também no aumento de performance que diminui os inputs necessários para introduzir o

retrabalho na aplicação.

6.1 Conclusão sobre os Métodos

Neste projeto foram desenvolvidas duas abordagens diferentes embutidas na aplicação retrabalhos. O método em vigor nas linhas de produção foi implementado a partir de *queries* e SP e o método com implementação em ML atuou como prova de conceito e de comparação de metodologias para a resolução deste problema (figura 6.1).

A realização desta dissertação permitiu retirar conclusões sobre as vantagens e desvantagens de cada método. A abordagem através de ferramentas de DM para a complexidade do problema em questão é mais benéfico, pois tanto a utilização de menos recursos diferentes como a rapidez de resposta para o contexto destes dados favorecem a sua adoção.

Um dos maiores constrangimentos para os algoritmos de ML nestes projetos foi conseguir aprender com os dados somente com recurso a uma coluna ou parâmetro de informação, dado que se o *dataset* recolhido tivesse um erro mais detalhado por colunas, a informação aprendida é maior e o seu desempenho melhora consideravelmente, sendo este um dos maiores desafios com as *queries* e SP. Quando a informação proveniente da base de dados se torna demasiado complexa, os servidores não conseguem retirar resultados em tempo exequível.

Com adoção de ferramentas de ML, um dos fatores que dificultou o processo de implementação em produção foi todo o ambiente de segurança e de produtividade das linhas de produção, bem como o hardware onde seria implementado o sistema de recomendação. Outro constrangimento registado foi a instalação do software e *frameworks* que permitissem compilar os algoritmos em computadores onde divergem consideravelmente a versão dos sistemas operativos.

Quanto à aplicação que recorre a ferramentas de DM, a principal dificuldade foi a otimização da *query*, que devido ao seu tempo de execução rondar os dez segundos, comprometia a viabilidade das recomendações a serem impostas nas linhas de produção. Este problema não acontece com o outro método, uma vez que o tempo de realizar a recomendação é muito reduzido. Desta forma, métodos de otimização de *queries* foram implementados para cumprir com as restrições do ambiente de produção.

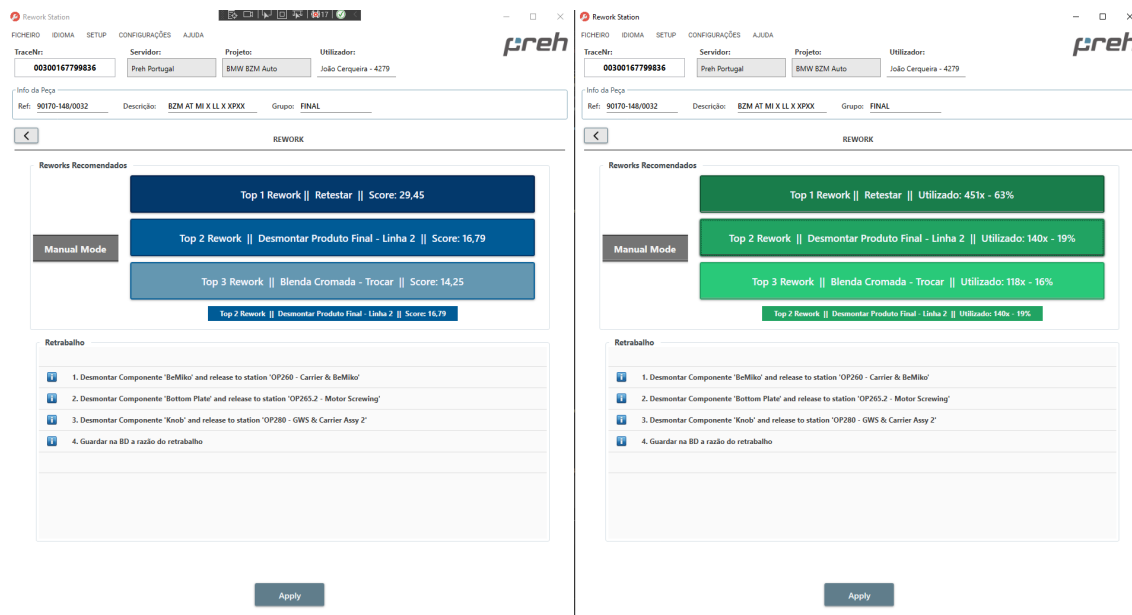


Figura 6.1: Comparação entre o desempenho das duas aplicações - ML vs SQL

No contexto geral, com as condições impostas e com o resultado dos desempenhos, a adoção através de ferramentas de DM foi a melhor opção para este projeto. O recurso a estas ferramentas para obter recomendações com base na frequência de retrabalhos presentes nos *datasets* retém pouca informação para permitirem os algoritmos aprender quais retrabalhos recomendar e quais devem rejeitar. As ferramentas de exploração de dados, por outro lado, são melhores para problemas que o fator seja o número de ocorrências.

6.2 Trabalho Futuro

Num projeto de ML há sempre espaço para melhoria. Nesta dissertação existe algumas experiências e uns detalhes que não foram explorados mais a fundo principalmente devido à falta de tempo. Houve uma fase de aprendizagem que retirou tempo precioso, uma vez que foi a primeira vez que realizei um projeto de DM juntamente com um projeto de ML, que ajudou em certa medida ao atraso na realização de certas tarefas.

De forma a continuar a evolução deste projeto, são enumeradas e descritas melhorias que promovem a execução da metodologia que ficou como prova de conceito:

- Apesar de ter sido possível tornar o método com ferramentas de DM totalmente funcional nas linhas de produção, sem erros encontrados até à data, a implementação em ML para alcançar a mesma performance de resultados necessita que o modelo de treino seja melhorado, fornecendo dados com maior teor de informação e com um maior número de instâncias;
- Alterar o processo de implementação do modelo de ML, e em vez de ser executado dentro da aplicação – que requer softwares de instalação, hardwares mais apropriados e *frameworks* necessários para a sua execução – executar o modelo numa plataforma de nuvem

compatível com Microsoft SQL Server, por exemplo Microsoft Azure, que permita tanto o treino atualizado do modelo como também devolver a recomendação quando o cliente (aplicação retrabalhos) pedir. Retirando esse esforço computacional dos computadores das linhas de produção para a nuvem, mesmo estes estando em ambiente fechado (fora da Internet pública), conseguem aceder localmente aos servidores da PREH, permitindo realizar as mesmas experiências numa cloud interna;

- Implementando a melhoria anterior, desenvolver os modelos de ML em Python, uma linguagem com mais suporte para implementar projetos de ML e que permite não só a exploração com mais detalhe todos os ajustes intrínsecos nos algoritmos como também a possibilidade de utilizar uma maior variedade de ferramentas de análise e de avaliação, monitorizando a performance dos mesmos. Apesar da linguagem C# recorrer a *frameworks* de ML.NET, apresentam muitos conteúdos para explorar e implementar mas não se comparam às bibliotecas de Python.
- Perante o sistema de processamentos de dados da empresa, durante a fase de pré-processamento de dados foi impossível averiguar se o retrabalho realizado para um erro foi apropriado ou não. Uma possibilidade é quando a peça retrabalhada retornar da linha EOL com erro, ter conhecimento se o retrabalho em uso foi correto ou não. Porém, com alguma frequência, após o retrabalho da peça, esta pode apresentar erros em diferentes módulos, desvirtuando a correta recolha dos dados. Pode também ser selecionado o retrabalho correto mas o mesmo não ser bem implementado. Desta maneira, é necessário criar um sistema que consiga detectar, com elevada precisão, o sucesso nos retrabalhos realizados.

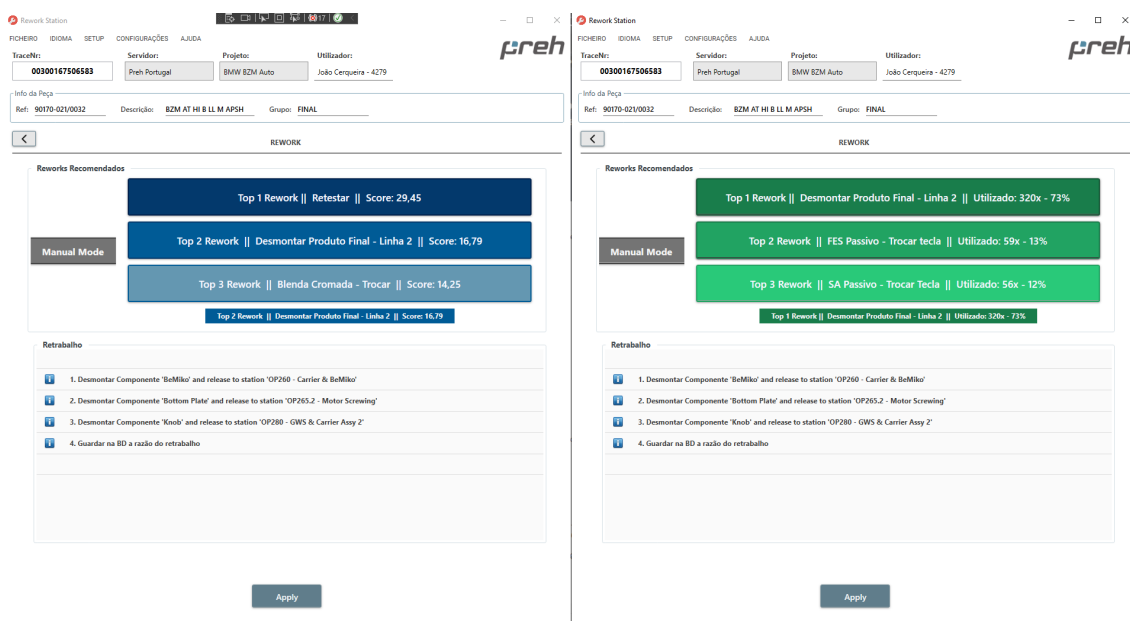


Figura 6.2: Comparação explícita da inconsistência em certas recomendações

De forma geral, não se espera que através dos métodos de ML se resolvam todos os problemas da área de previsão de resultados, mais concretamente em dados desbalanceados de multiclasse (figura 6.2, no entanto, o seu potencial não deve ser, de todo, desprezado. Os modelos de ML são apenas uma parte de um processo maior para resolver um problema específico, e é importante manter um panorama geral em mente. Ao aprofundar os aspetos técnicos do ML, é fácil perder de vista os objetivos finais. É importante manter em mente todas as suposições criadas, de forma explícita ou implícita, quando são construídos modelos de ML.

Anexo A

Anexo A - Códigos Desenvolvidos

.

A.1 SQL SERVER - Código SQL

A.1.1 Query não otimizada desenvolvida, antes de implementar Store Procedure.

[11pt] breakatwhitespace

```

1000 DECLARE @num INT;
1001 SET @num = 3;
1002
1003 ; WITH Main AS
1004     (SELECT DISTINCT
1005         Rr.[ID_WSJob] AS Rework_job ,
1006         Wj.ID_WSJob AS Error_job ,
1007         D_error.[id_deviceresulttestcharacteristic] AS
1008         Test_characteristic ,
1009         D_error.[id_deviceresulttestobject] AS Test_object ,
1010         Rr.[ID_Rework] ,
1011         Reworks.[reworkname] ,
1012         [id_defecttype] AS Defect_type ,
1013         Rr.[id_reworkdiagnostic] ,
1014         Diagnostics.[reworkdiagnostic] AS Diagnostico ,
1015         Rr.[resultdescription] ,
1016         Rr.[resultdate] ,
1017         Rr.[TraceNr] ,
1018         Testobjectname ,
1019         Characteristicname ,
1020         [groupname]
1021     FROM [bmw_bzm_auto].[rework].[reworkresults] Rr WITH (Nolock)
1022     INNER JOIN WSJobs Wj WITH (Nolock)
1023     ON Wj.ID_WSJob = (SELECT Top 1 ID_WSJob
1024         FROM WSJobs WITH (Nolock)
1025         WHERE ID_WSJob < Rr.ID_WSJob
1026         AND TraceNr = Rr.TraceNr
1027         AND Id_subws
1028         NOT IN
1029         (SELECT [id_subws]
1030             FROM [bmw_bzm_auto].[dbo].[wssubdevices]
1031             WHERE Subwsname LIKE '%REWORK%')
1032         AND ID_WSJob
1033         IN (SELECT [ID_WSJob]
1034             FROM Deviceerrorresults WITH (Nolock))
1035         AND Rr.TraceNr

```

```

1035             NOT IN
1036                 (SELECT [TraceNr]
1037                     FROM Samples_tracenrs WITH (Nolock))
1038                 ORDER BY 1 DESC)
1039     INNER JOIN Rework.Reworks ON Rr.ID_Rework = Reworks.ID_Rework
1040     INNER JOIN Rework.Diagnostics ON Rr.Id_reworkdiagnostic =
1041     Diagnostics.Id_reworkdiagnostic
1042     INNER JOIN Deviceerrorresults AS D_error ON Wj.ID_WSJob =
1043     D_error.ID_WSJob
1044     INNER JOIN Deviceresulttestcharacteristics ON
1045     D_error.Id_deviceresulttestcharacteristic =
1046     Deviceresulttestcharacteristics.
1047     Id_deviceresulttestcharacteristic
1048     INNER JOIN Deviceresulttestobjects ON D_error.
1049     Id_deviceresulttestobject = Deviceresulttestobjects.
1050     Id_deviceresulttestobject
1051     WHERE Rr.Resultdate > '20220101'
1052     AND D_error.Id_deviceerrorresult in
1053         (SELECT Max([ id_deviceerrorresult ])
1054             FROM [bmw_bzm_auto].[dbo].[deviceerrorresults]
1055             WITH (Nolock)
1056             WHERE ID_WSJob = Wj.ID_WSJob
1057             GROUP BY ID_WSJob))
1058
1059 , Sub_main AS (SELECT Error_job , Count(Rework_job) AS Conta
1060                 FROM Main
1061                 GROUP BY Error_job
1062             )
1063
1064 ,Sub_main2 AS (SELECT*
1065                 FROM Sub_main
1066                 WHERE Conta > 1
1067             )
1068
1069 , Final_1 AS (SELECT *
1070                 FROM Main
1071                 WHERE Main.Error_job not in (SELECT Sub_main2.Error_job
1072                                                 FROM Sub_main2)
1073             )
1074 , Final_2 AS (SELECT Final_1.*, D_error_2.Textvalue

```

```

1071         FROM Final_1
1072         RIGHT JOIN Deviceerrorresults AS D_error_2 ON
Error_job=D_error_2.ID_WSJob
1073     )
1074
1075 , Table_6273 AS (SELECT Final_2.Error_job , Final_2.Rework_job ,
TraceNr , Textvalue , ID_Rework , Reworkname , Resultdate
1076         FROM Final_2
1077         WHERE Test_characteristic=6273
1078         GROUP BY Textvalue , Final_2.Error_job , Final_2.
Rework_job , TraceNr , ID_Rework , Reworkname , Resultdate
1079     )
1080 , Table_6273_v2 AS (SELECT Textvalue , ID_Rework , Count(Error_job
) AS County
1081         FROM Table_6273
1082         GROUP BY Textvalue , ID_Rework
1083     )
1084 , Test AS (SELECT Textvalue , ID_Rework , County
1085         FROM (SELECT *, Row_number() OVER (PARTITION BY
Textvalue ORDER BY County DESC) AS Seqnum
1086         FROM Table_6273_v2) T
1087         WHERE Seqnum <= @num
1088     )
1089 , counter_Distints_Classes_QWALL as (SELECT DISTinct TextValue ,
ID_Rework
1090         FROM Final_1
1091         WHERE (Test_characteristic=6273 and
Test_object=603)
1092     )
1093 SELECT Textvalue , ID_Rework , County
1094 from Test

```

Listing A.1: Código da query não otimizada

A.1.2 Store Procedure implementado na aplicação de Produção (Pós otimização)

```

[11pt] breakatwhitespace
1000 USE [BMW_BZM_AUTO]
1001 GO
1002 /***** Object:  StoredProcedure [Rework].[GetTop3Reworks]
      Script Date: 15-06-2022 11:39 *****/
1003 SET ANSI_NULLS ON
1004 GO
1005 SET QUOTED_IDENTIFIER ON
1006 GO
1007
1008 ALTER PROCEDURE [Rework].[GetTop3Reworks]
1009     @TraceNr bigint
1010 AS
1011
1012 --DECLARE @TraceNr bigint = 300165897995
1013 DECLARE @ID_DRTestCharacteristic int;
1014 DECLARE @ID_DRTestObject int;
1015 DECLARE @ErrorDetail nvarchar(150);
1016
1017 SELECT TOP 1 @ID_DRTestCharacteristic=
        ID_DeviceResultTestCharacteristic , @ID_DRTestObject=
        ID_DeviceResultTestObject , @ErrorDetail = TextValue
1018 FROM DeviceErrorResults
1019 WHERE TraceNr = @TraceNr and ID_SubWS in (SELECT ID_SubWS from
        WSSubDevices WHERE EOLDevice=1 or SubWSName like '%Muro%')
1020 ORDER BY ID_DeviceErrorResult DESC
1021
1022
1023
1024 ;WITH cteReworks AS
1025     (SELECT DISTINCT RR.ID_WSJob AS Rework_job ,
1026         J.ID_WSJob AS Error_job ,
1027         DER.ID_DeviceResultTestCharacteristic
1028             AS ID_Characteristic ,
1029         DER.ID_DeviceResultTestObject AS ID_Object ,
1030         DER.TextValue AS ErrorDetail ,
1031         RR.TraceNr ,
1032         RR.ID_Rework ,

```

```

1033         ReworkName ,
1034         RR.ResultDescription
1035     FROM Rework.ReworkResults RR WITH (NOLOCK)
1036     INNER JOIN WSJobs J WITH (NOLOCK) ON J.ID_WSJob =
1037         (SELECT TOP 1 ID_WSJob
1038          FROM WSJobs WITH (NOLOCK)
1039          WHERE ID_WSJob < RR.ID_WSJob
1040          AND TraceNr = RR.TraceNr
1041          AND ID_SubWS in (SELECT ID_SubWS
1042                          from WSSubDevices WHERE EOLDevice=1
1043                          OR SubWSName like '%Muro%')
1044     ORDER BY 1 DESC)
1045     INNER JOIN Rework.Reworks ON RR.ID_Rework=Reworks.ID_Rework
1046     INNER JOIN DeviceErrorResults AS DER ON J.ID_WSJob = DER.
1047     ID_WSJob
1048     WHERE RR.Resultdate > '20220101'
1049           AND DER.ID_DeviceResultTestCharacteristic =
1050           @ID_DRTestCharacteristic
1051           AND DER.ID_DeviceResultTestObject =
1052           @ID_DRTestObject
1053           AND (( @ID_DRTestCharacteristic = 6273 AND DER.
1054           TextValue = @ErrorDetail)
1055           OR ( @ID_DRTestCharacteristic <> 6273 AND DER.
1056           TextValue IS NULL)))
1057
1058 SELECT TOP(3) ID_Rework , ReworkName , Count(Error_job) AS Counter
1059 FROM cteReworks
1060 GROUP BY ID_Characteristic , ID_Object , ID_Rework , ReworkName
1061 ORDER BY Counter DESC

```

Listing A.2: Store Procedure Código

A.2 Machine Learning: Python e C#

A.2.1 Machine Learning Colab - Código Python

```
[11pt] breakatwhitespace
1000 from sklearn.naive_bayes import MultinomialNB
1001 from sklearn.metrics import classification_report
1002 from sklearn.ensemble import RandomForestClassifier
1003 from sklearn.tree import DecisionTreeClassifier
1004 from sklearn.ensemble import AdaBoostClassifier
1005 from sklearn.datasets import make_classification
1006 from sklearn.model_selection import KFold
1007 from sklearn.model_selection import cross_val_score
1008 from sklearn.linear_model import LogisticRegression
1009 from sklearn.model_selection import GroupKFold
1010 from sklearn.datasets import make_classification
1011 from sklearn.multioutput import MultiOutputClassifier
1012 from sklearn.ensemble import RandomForestClassifier
1013 from sklearn.utils import shuffle
1014 import numpy as np
1015 import pandas as pd
1016 from sklearn.datasets import make_multilabel_classification
1017 from sklearn.model_selection import train_test_split
1018 from tensorflow.keras.models import Sequential
1019 from tensorflow.keras.layers import Dense
1020 from tensorflow.keras.losses import binary_crossentropy
1021 from tensorflow.keras.optimizers import Adam
1022 from sklearn.metrics import hamming_loss
1023 from sklearn.metrics import accuracy_score
1024 from sklearn.metrics import f1_score
1025 from sklearn.metrics import precision_score
1026 from sklearn.metrics import recall_score
1027 from sklearn.model_selection import train_test_split
1028 from sklearn.ensemble import RandomForestClassifier
1029 from sklearn.preprocessing import OrdinalEncoder
1030 from sklearn.preprocessing import OneHotEncoder
1031 import seaborn as sns
1032 from sklearn import preprocessing
1033 from numpy.random import seed
1034 from numpy.random import randn
1035 from numpy import mean
```

```
1036 from numpy import std
1037
1038 from google.colab import files
1039 uploaded = files.upload()
1040
1041 tables = pd.read_csv( "ML_Data_EOL_Jan-Maio_Correct.csv", sep=',
    ', skipinitialspace=True, encoding='Latin-1', index_col=
    False )
1042 table_v2 = tables.dropna()
1043 table_v2.isnull().sum()
1044
1045 Count = pd.DataFrame(table_v2)
1046 df1=Count.groupby(['Erro', 'ID_Rework']).size().reset_index(name
    ='counts')
1047 df11=Count.groupby(['Erro']).size().reset_index(name='count_Erro
    ')
1048 df5=pd.DataFrame(df1)
1049 df2=df5.groupby(['Erro']).describe()
1050 df3=Count.groupby('Erro')['ID_Rework'].mean()
1051 df4=pd.DataFrame(df3)
1052
1053 df2=df1.groupby(['Erro'])['counts'].describe()[['count', 'mean',
    'max', '50%',
1054 '75%']]
1055
1056 df6 = pd.merge(df5, df2, how="inner", on="Erro") #df2
1057 df7 = pd.merge(df6, Count, how="inner", on=['Erro','ID_Rework'])
1058 df8=df7
1059 df8 = df8.rename(columns = \{"75%":"mediana"\})
1060
1061 df9 = df8.drop(df8[(df8['count'] > 2)&(df8['mediana'] >= df8['
    counts'])].index)
1062 g = df5.groupby(['Erro']).apply(lambda x:
1063 x.nlargest(3,['counts'])).reset_index(drop=True)
1064 df40 = pd.merge(g, df11, how="inner", on=['Erro'])
1065
1066 df_Final = df40.drop(df40[(df40['count_Erro'] < 10) ].index)
1067 diff_df = pd.merge(df7, df_Final, how='outer', indicator='Exist'
    )
1068 df20= diff_df.loc[diff_df['Exist'] == 'both']
```

```
1069
1070 Y = df20.iloc[:, 1]
1071 X = df20.iloc[:, 0]
1072
1073 dum_df = pd.get_dummies(X)
1074
1075 gKFOLD = GroupKFold(n_splits=10)
1076
1077 cv = KFold(n_splits=10, random_state=1, shuffle=True)
1078 # create model
1079 Multi = MultinomialNB()
1080 Logi = LogisticRegression()
1081 Forest = RandomForestClassifier()
1082 Tree = DecisionTreeClassifier()
1083 # evaluate model
1084 scores_Multinomial = cross_val_score(Multi, dum_df, Y, scoring='
    accuracy',
1085 cv=cv, n_jobs=-1)
1086 scores_Logistic = cross_val_score(Logi, dum_df, Y, scoring='
    accuracy',
1087 cv=cv, n_jobs=-1)
1088 scores_RandomFr = cross_val_score(Forest, dum_df, Y, scoring='
    accuracy',
1089 cv=cv, n_jobs=-1)
1090 scores_Tree = cross_val_score(Tree, dum_df, Y, scoring='accuracy
    ',
1091 cv=cv, n_jobs=-1)
1092 # report performance
1093 print('Accuracy Naive Bayes: %.3f (%.3f)' % (mean(
    scores_Multinomial),
1094 std(scores_Multinomial)))
1095 print('Accuracy Logistic Regression : %.3f (%.3f)' % (mean(
    scores_Logistic),
1096 std(scores_Logistic)))
1097 print('Accuracy Random Forest: %.3f (%.3f)' % (mean(
    scores_RandomFr),
1098 std(scores_RandomFr)))
1099 print('Accuracy Tree Classifier: %.3f (%.3f)' % (mean(scores_Tree)
    ,
1100 std(scores_Tree)))
```

```
1101
1102 n = 3
1103 probas = random.predict_proba(X_test)#.iloc[400:410])
1104 top_n_labels_idx = np.argsort(-probas, axis=1)[: , :n]
1105 top_n_probs = np.round(-np.sort(-probas),3)[: , :n]
1106 top_n_labels = [random.classes_[i] for i in top_n_labels_idx]
1107 results = list(zip(top_n_labels, top_n_probs))
1108
1109 tabela_2_reworks_RandomF = pd.DataFrame(results)
1110 tabela_2_reworks_RandomF.rename({'0': 'RandomForest Labels','1':
    'RandomForest
1111 Prob'}, axis=1, inplace=True)
1112 probas_df = X_test.copy()
1113 probas_df=probas_df.reset_index(drop=True)
1114 #tabela_result_1 = pd.concat([X_test_Try,
    tabela_2_reworks_RandomF], axis=1)
1115
1116
1117 #tabela_result.to_excel('update2.xlsx', index = False, header=
    True,encoding='utf-8')
1118 #files.download('update2.xlsx')
1119
1120 print("Random Forest")
1121 print(tabela_2_reworks_RandomF)
1122
1123
1124
1125 probas = classifier.predict_proba(X_test)#.iloc[400:410])
1126 top_n_labels_idx = np.argsort(-probas, axis=1)[: , :n]
1127 top_n_probs = np.round(-np.sort(-probas),3)[: , :n]
1128 top_n_labels = [classifier.classes_[i] for i in top_n_labels_idx
    ]
1129 results = list(zip(top_n_labels, top_n_probs))
1130 print("LogisticRegression")
1131 print(pd.DataFrame(results))
1132
1133 results_logistic = pd.DataFrame(results)
1134 tabela_result_2 = pd.concat([tabela_2_reworks_RandomF,
    results_logistic],
1135 axis=1)
```

```
1136
1137
1138 probas = Decision.predict_proba(X_test)#.iloc[400:410])
1139 top_n_lables_idx = np.argsort(-probas, axis=1)[: , :n]
1140 top_n_probs = np.round(-np.sort(-probas),3)[: , :n]
1141 top_n_labels = [Decision.classes_[i] for i in top_n_lables_idx]
1142 results = list(zip(top_n_labels, top_n_probs))
1143 print("Tree")
1144 print(pd.DataFrame(results))
1145
1146 results_Ddecision_Tree = pd.DataFrame(results)
1147 tabela_result_3 = pd.concat([tabela_result_2,
                             results_Ddecision_Tree], axis=1)
1148
1149 probas = multinomial.predict_proba(X_test)#.iloc[400:410])
1150 top_n_lables_idx = np.argsort(-probas, axis=1)[: , :n]
1151 top_n_probs = np.round(-np.sort(-probas),3)[: , :n]
1152 top_n_labels = [multinomial.classes_[i] for i in
                  top_n_lables_idx]
1153 results = list(zip(top_n_labels, top_n_probs))
1154 print("Multinomial")
1155 print(pd.DataFrame(results))
1156 data = pd.DataFrame(results)
1157 results_Multinomial = pd.DataFrame(results)
1158 tabela_result_4 = pd.concat([tabela_result_3,
                             results_Multinomial], axis=1)
```

Listing A.3: Código Python para ML

A.2.2 Machine Learning ML.NET - C#

Secção de treino do modelo

[11pt] breakatwhitespace

```

1000
1001 public static void BuildTrainEvaluateAndSaveModelEOL(MLContext
      mlContext){
1002
1003 IDataView data = mlContext.Data.LoadFromTextFile<ErroData>(
      BaseDatasetsRelativePath_EOL , separatorChar: ';',
1004           hasHeader: true , trimWhitespace: true);
1005
1006 var partitions = mlContext.Data.TrainTestSplit(data ,
      testFraction: 0.1);
1007
1008 var dataProcessPipeline = mlContext.Transforms
1009   .Categorical.OneHotEncoding(@"Erro", inputColumnName: nameof(
      ErroData.Erro))
1010   .Append(mlContext.Transforms.Concatenate(@"Features", @"Erro")
      )
1011   .Append(mlContext.Transforms.Conversion
1012   .MapValueToKey(@"Label", nameof(ErroData.Label)))
1013   .Append(mlContext.Transforms.NormalizeMinMax(@"Features", @"
      Features"));
1014
1015 var trainer = mlContext.MulticlassClassification.Trainers.
1016   SdcaMaximumEntropy(labelColumnName: @"Label",
1017   featureColumnName: @"Features")
1018   .Append(mlContext.Transforms.Conversion.
      MapKeyToValue
1019   ("PredictedLabel", "PredictedLabel"));
1020
1021 var trainingPipeline = dataProcessPipeline.Append(trainer);
1022
1023 var trainingPipeline_Fit = trainingPipeline.Fit(partitions.
      TrainSet);
1024
1025 var trainingPipeline_Fit_Trans = trainingPipeline_Fit.Transform(
      partitions.TestSet);
1026

```

```
1027 mlContext.Model.Save(trainingPipeline_Fit , data.Schema ,  
    ModelPath_EOL);  
1028  
1029  
1030 var metrics = mlContext.MulticlassClassification.Evaluate(  
    trainingPipeline_Fit_Trans , "Label", "Score", "PredictedLabel"  
    , 3);  
1031  
1032 Console.WriteLine("----ERROR   EOL   --   METRICS NaiveBayes  
    -----");  
1033 Console.WriteLine("EOL MicroAccuracy = {0} ", metrics.  
    MicroAccuracy);  
1034 Console.WriteLine("EOL MacroAccuracy = {0}", metrics.  
    MacroAccuracy);  
1035 Console.WriteLine("EOL PerClassLogLoss = {0}", metrics.  
    PerClassLogLoss[0]);  
1036 Console.WriteLine("-----END-----"  
    );  
1037 }
```

Listing A.4: Código em C# para ML

Anexo B

***Flow Charts* dos dois métodos implementados**

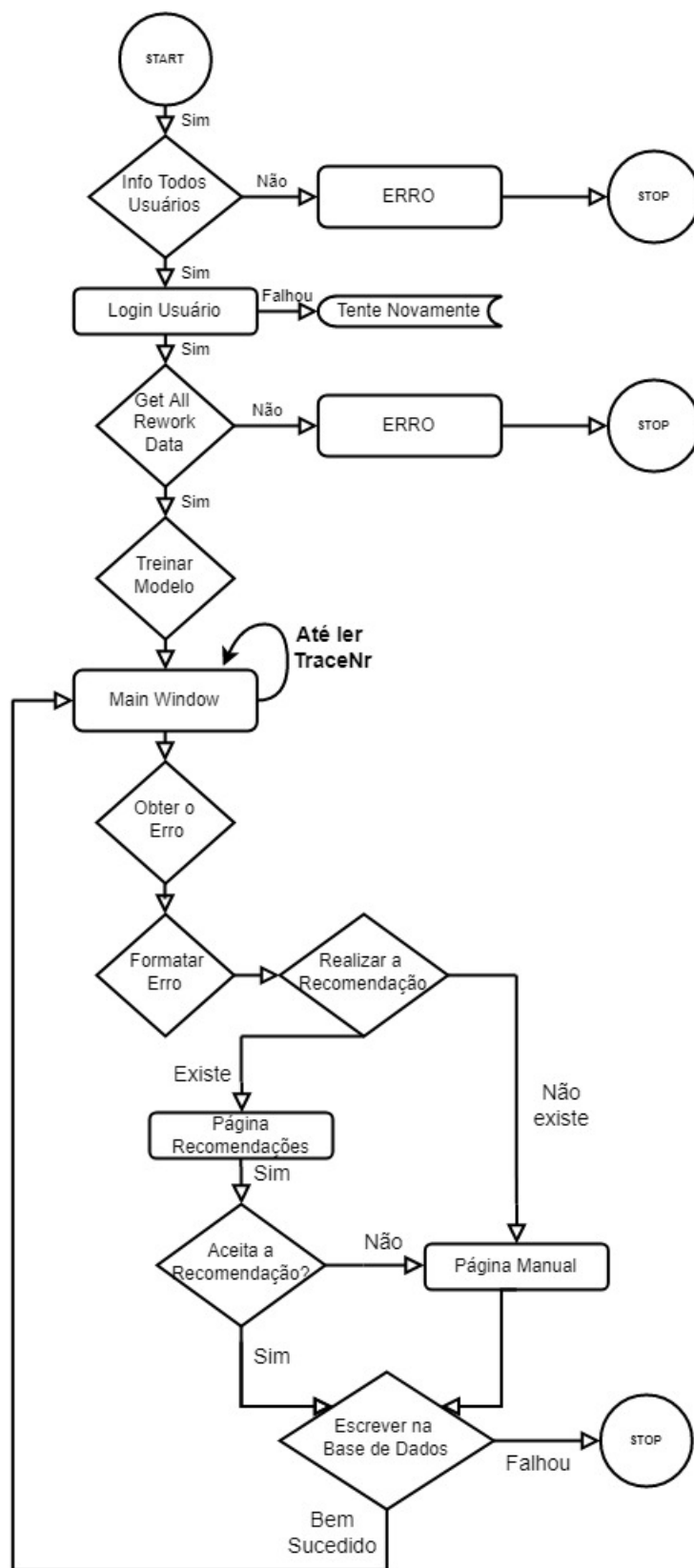


Figura B.1: Diagrama da Implementação do Modelo de ML na Aplicação

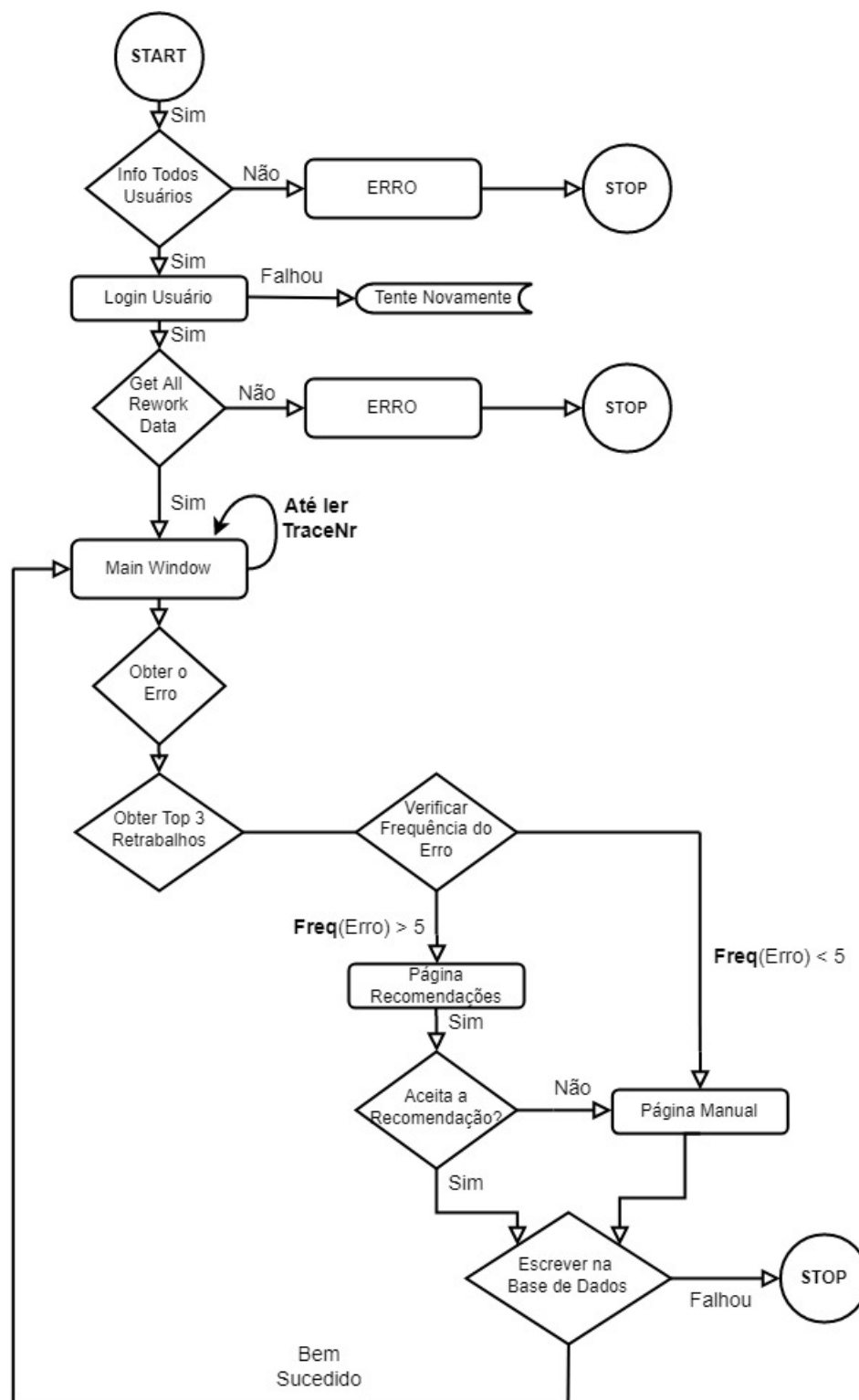


Figura B.2: Diagrama da Implementação através de SP na Aplicação

Anexo C

Tabelas adicionais aos *DataSets* utilizados

C.1 Tabelas Informativas sobre os *DataSets*

C.1.1 Top 30 Erros QWALL mais ocorridos

Erro	Contagem
Auto Friso cromado design NOK	1990
Auto Blenda GWS pintura/injeção NOK	1405
Auto Friso cromado c/ crack	1086
Auto SST cromado c/ risco/dano	988
Auto Blenda GWS c/ risco/dano	955
Auto FES/SA activas injeção/pintura NOK	890
Auto Knob vidro c/ risco/dano	874
Auto FES/SA activas c/ risco/dano	739
FES/SA com Risco/Dano	718
Auto Blenda GWS Iluminação NOK	611
Friso Cromado com Risco/Dano	564
Auto Knob vidro pintura/Injeção NOK	546
Auto FES/SA passivas c/ risco/dano	513
Blenda GWS com Risco/Dano	494
FES/SA com defeito Pintura/Injeção	457
Auto FES/SA passivas injeção/pintura NOK	411
Auto FES/SA ativas Iluminação NOK	400
EPB com Risco/Dano	396
Haptik NOK	389
Blenda GWS com defeito Pintura/Injeção	352
Auto EPB passivo c/ risco/dano	289
Auto Manete c/ ruído (Mov. Manual)	229
SST com Risco/Dano	229
Auto FES/SA passivas acionamento NOK	225
Auto EPB ativo injeção/pintura NOK	215
Auto EPB ativo c/ risco/dano	207
Auto EPB passivo injeção/pintura NOK	167
Auto Manete c/ ruído (Mov. Electrico)	166
Auto EPB ativo Iluminação / fugas luz NOK	164
Auto Knob SWK c/ risco/dano	161
M FES/SA injeção/pintura NOK	161

Tabela C.1: Tabela representativa dos erros mais usuais provenientes da estação de qualidade

C.1.2 Top 30 Erros EOL mais ocorridos

Erro	Contagem
6453 - 249	1459
8463 - 1162	889
6220 - 1461	883
7942 - 822	875
6 - 788	702
8442 - 1155	558
7936 - 735	544
7935 - 728	455
6220 - 548	439
90 - 204	411
86 - 32	391
8430 - 508	364
9651 - 735	344
6184 - 497	339
46 - 11	319
50 - 179	318
89 - 604	310
8462 - 1161	305
55 - 193	280
7942 - 768	274
50 - 182	231
7878 - 542	226
7935 - 542	205
90 - 728	205
4 - 207	203
22 - 37	201
22 - 36	200
90 - 32	197
86 - 204	196
3 - 4	194
9649 - 1571	194

Tabela C.2: Tabela representativa dos erros mais usuais provenientes da linha de EOL

C.1.3 Contagem de Retrabalhos mais Realizados no Dataset QWALL

ReworkName	ID_Rework	Contagem
GWS - Trocar Blenda	24	4228
Desmontar Produto Final - Linha 2	22	4014
Blenda Cromada - Trocar	6	3893
Knob - Trocar	7	1428
Retestar	9	1386
EPB - Trocar Cap	16	1191
SST - Trocar Cap	17	1111
SA Passivo - Trocar Tecla	2	591
Trim - Trocar	12	448
FES Passivo - Trocar tecla	3	419
DESMONTAR PEÇA M	37	321
SES - Trocar tecla	10	287
GWS - Blenda Mal Encaixada	19	109
EPB - Trocar Condutor De Luz	1	105
Knob M-Tocar	35	77
Substituir Bezel GWS - LCI	42	57
Desmontar Produto Final BEV - Motor Linha 2	21	54
GWS BLENDA M-TROCAR	36	52
Desmotar Peça M	32	49
Desmontar produto Final - LCI	51	48
EPB - Mal Encaixado	26	45
Knob - Trocar tecla P	4	39
Shell - Trocar	28	26
Substituir Cap P - LCI	45	19
FES - Cap Mal Encaixada	15	17
Limpar "Caps"	13	12
Substituir Bezel EPB - LCI	40	12
Blenda Cromada Mal Encaixada	18	11
Substituir Bezel Actuation - LCI	41	8
SA - Cap Mal Encaixada	14	7
Unlock - Trocar tecla	5	3
Substituir Side Bezel ZBE - LCI	44	3
Unlock - Trocar Silicon Mat	8	2
Substituir Toggle (Defeito Pintura) - LCI	47	1
Substituir Slider GWS - LCI	49	1
Unlock - Trocar tecla	5	2
Motor - Trocar Base do Motor	27	1

Tabela C.3: Tabela representativa dos retrabalhos mais realizados dos erros provenientes da estação de qualidade

C.1.4 Contagem de Retrabalhos Realizados Para os Erros de EOL

ReworkName	ID_Rework	Contagem
Desmontar Produto Final - Linha 2	22	9953
Retestar	9	7689
Blenda Cromada - Trocar	6	1527
Knob - Trocar	7	1331
GWS - Trocar Blenda	24	934
Trim - Trocar	12	827
SST - Trocar Cap	17	517
DESMONTAR PEÇA M	37	514
EPB - Trocar Cap	16	365
Desmontar Produto Final BEV - Motor Linha 2	21	267
Knob M-Tocar	35	177
SES - Trocar tecla	10	161
FES Passivo - Trocar tecla	3	154
SA Passivo - Trocar Tecla	2	152
Desmontar produto Final - LCI	51	128
Desmotar Peça M	32	124
GWS - Blenda Mal Encaixada	19	90
GWS BLENDA M-TROCAR	36	82
Ligar Cabo	25	48
Unlock - Trocar Silicon Mat	8	41
Knob - Trocar tecla P	4	37
EPB - Trocar Condutor De Luz	1	34
EPB - Mal Encaixado	26	31
Substituir Bezel GWS - LCI	42	30
FES - Cap Mal Encaixada	15	20
Substituir Bezel Actuation - LCI	41	14
Limpar "Caps"	13	12
Substituir Cap P - LCI	45	8
Blenda Cromada Mal Encaixada	18	8
SA - Cap Mal Encaixada	14	7
Substituir Side Bezel ZBE - LCI	44	4
Substituir Bezel EPB - LCI	40	4
Substituir rubber mat GWS - LCI	48	3
Substituir Slider GWS - LCI	49	3
Shell - Trocar	28	2
Unlock - Trocar tecla	5	2
Motor - Trocar Base do Motor	27	1

Tabela C.4: Tabela representativa dos retrabalhos mais realizados dos erros provenientes da estação de qualidade

C.1.5 Contagem de Retrabalhos Realizados Por Dia

Total	Dia
236	2022-06-28
904	2022-06-27
408	2022-06-26
641	2022-06-25
685	2022-06-24
744	2022-06-23
764	2022-06-22
821	2022-06-21
816	2022-06-20
266	2022-06-18
679	2022-06-17
172	2022-06-16
651	2022-06-15
756	2022-06-14
524	2022-06-13
246	2022-06-10
721	2022-06-09
609	2022-06-08
658	2022-06-07
507	2022-06-06
336	2022-06-04
716	2022-06-03
733	2022-06-02
626	2022-06-01

Tabela C.5: Número de retrabalhos realizados diariamente

C.1.6 Contagem de Retrabalhos Realizados com Auxilio do Sistema de Recomendações

Total	Dia
155	2022-06-28
551	2022-06-27
210	2022-06-26
273	2022-06-25
313	2022-06-24
309	2022-06-23
315	2022-06-22
357	2022-06-21
233	2022-06-20
129	2022-06-18
341	2022-06-17
91	2022-06-16
174	2022-06-15
147	2022-06-14
101	2022-06-13
2	2022-06-10
104	2022-06-09
79	2022-06-08
83	2022-06-07
82	2022-06-06
101	2022-06-04
156	2022-06-03
178	2022-06-02
153	2022-06-01

Tabela C.6: Número de recomendações realizados diariamente

Referências

- [1] Ana Azevedo e Manuel Filipe Santos. Kdd, semma and crisp-dm: A parallel overview. página 182 – 185, 2008. Cited by: 191. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-58449118356&partnerID=40&md5=e2815648f01d0d58645e591f1042c698>.
- [2] Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, e Zhuang Wang. Log-based predictive maintenance. página 1867 – 1876, 2014. Cited by: 129. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84907021022&doi=10.1145%2f2623330.2623340&partnerID=40&md5=8453ea0207ea2c95d4f6a2c2441d5114>, doi:10.1145/2623330.2623340.
- [3] URL: https://www.researchgate.net/figure/End-of-Line-testing-area-of-assembly-fig1_338096090.
- [4] URL: <https://www.semanticscholar.org/paper/End-Of-Line-Testing-Klippel/bbb8428dc4e5a787e6453bf25fb06d2d6071c2ce>.
- [5] Plataforma desenvolvimento .net. URL: <https://dotnet.microsoft.com/en-us/download>.
- [6] Frameworks da .net. URL: <https://dotnet.microsoft.com/en-us/download/dotnet-framework>.
- [7] Jan De Vries. *The industrious revolution: Consumer behavior and the household economy, 1650 to the present*. 2008. Cited by: 728. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84923949388&doi=10.1017%2fCBO9780511818196&partnerID=40&md5=7ca1ceaf060cde8f2a2233c34057b93d>, doi:10.1017/CBO9780511818196.
- [8] URL: <https://site.autodoc.com.br/conteudos/industria-4-0-tudo-o-que-voce-precisa-saber/>.
- [9] Ray Y. Zhong, Xun Xu, Eberhard Klotz, e Stephen T. Newman. Intelligent manufacturing in the context of industry 4.0: A review. *Engineering*, 3(5):616 – 630, 2017. Cited by: 1145; All Open Access, Gold Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85040743821&doi=10.1016%2fJ.ENG.2017.05.015&partnerID=40&md5=05014d42447feb267ee8ecadba47ae7c>, doi:10.1016/J.ENG.2017.05.015.
- [10] Li Da Xu, Eric L. Xu, e Ling Li. Industry 4.0: State of the art and future trends. *International Journal of Production Research*, 56(8):2941 – 2962, 2018. Cited by: 1189. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0>.

- 0-85064005739&doi=10.1080%2f00207543.2018.1444806&partnerID=40&md5=bd22cd2bac87605bf42dc17b68c13ee3, doi:10.1080/00207543.2018.1444806.
- [11] Luigi Atzori, Antonio Iera, e Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787 – 2805, 2010. Cited by: 9520. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-77956877124&doi=10.1016%2fj.comnet.2010.05.010&partnerID=40&md5=14b4f3bf4d13ee412acf573a91a817de>, doi:10.1016/j.comnet.2010.05.010.
- [12] URL: <https://www.intelegain.com/the-ultimate-guide-to-iot-driven-digital-transfor>
- [13] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, e Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys and Tutorials*, 17(4):2347 – 2376, 2015. Cited by: 4476. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84941085241&doi=10.1109%2fCOMST.2015.2444095&partnerID=40&md5=2c775e20c00babcb81c91bea21f3d89f7>, doi:10.1109/COMST.2015.2444095.
- [14] Xun Xu. From cloud computing to cloud manufacturing. *Robotics and Computer-Integrated Manufacturing*, 28(1):75–86, 2012. URL: <https://www.sciencedirect.com/science/article/pii/S0736584511000949>, doi:<https://doi.org/10.1016/j.rcim.2011.07.002>.
- [15] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, e K. Ueda. Cyber-physical systems in manufacturing. *CIRP Annals*, 65(2):621 – 641, 2016. Cited by: 925. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84990841502&doi=10.1016%2fj.cirp.2016.06.005&partnerID=40&md5=5f72e61fc476b303ff9cde368f8326d3>, doi:10.1016/j.cirp.2016.06.005.
- [16] Elisabeth J. Umble, Ronald R. Haft, e M. Michael Umble. Enterprise resource planning: Implementation procedures and critical success factors. *European Journal of Operational Research*, 146(2):241 – 257, 2003. Cited by: 1013. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-0037448859&doi=10.1016%2fS0377-2217%2802%2900547-7&partnerID=40&md5=a29dcd41fab50506e5bb2898db43fbfe>, doi:10.1016/S0377-2217(02)00547-7.
- [17] URL: <https://www.redwoodlogistics.com/how-does-erp-software-work/>.
- [18] Madjid Tavana, Vahid Hajipour, e Shahrzad Oveisi. Iot-based enterprise resource planning: Challenges, open issues, applications, architecture, and future research directions. *Internet of Things*, 11:100262, 2020. URL: <https://www.sciencedirect.com/science/article/pii/S2542660520300962>, doi:<https://doi.org/10.1016/j.iot.2020.100262>.
- [19] Jay Lee, Behrad Bagheri, e Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18 – 23, 2015. Cited by: 2583. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84921300723&doi=10.1016%2fj.mfglet.2014.12.001&partnerID=40&md5=eb20b1e6406f4dce39d33254394c80e4>, doi:10.1016/j.mfglet.2014.12.001.
- [20] Ragunathan Rajkumar, Insup Lee, Lui Sha, e John Stankovic. Cyber-physical systems: The next computing revolution. página 731 – 736, 2010. Cited by:

1305. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-77956217277&doi=10.1145%2f1837274.1837461&partnerID=40&md5=9b9616c23f8b877affeb5403a3f62e6b>, doi:10.1145/1837274.1837461.
- [21] Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, e Lejia Shen. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57(7):2117 – 2135, 2019. Cited by: 834. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85055168966&doi=10.1080%2f00207543.2018.1533261&partnerID=40&md5=ae4b6928e624c28ee3db192edcc4999c>, doi:10.1080/00207543.2018.1533261.
- [22] Dmitry Ivanov, Alexandre Dolgui, e Boris Sokolov. Cloud supply chain: Integrating industry 4.0 and digital platforms in the “supply chain-as-a-service”. *Transportation Research Part E: Logistics and Transportation Review*, 160:102676, 2022. URL: <https://www.sciencedirect.com/science/article/pii/S1366554522000709>, doi:<https://doi.org/10.1016/j.tre.2022.102676>.
- [23] Mina Nasiri, Juhani Ukko, Minna Saunila, e Tero Rantala. Managing the digital supply chain: The role of smart technologies. *Technovation*, 96-97:102121, 2020. URL: <https://www.sciencedirect.com/science/article/pii/S0166497220300110>, doi:<https://doi.org/10.1016/j.technovation.2020.102121>.
- [24] J. Moreira, A. Carvalho, e T. Horvath. *A General Introduction to Data Analytics*. Wiley, 2018. URL: <https://books.google.pt/books?id=UoZfDwAAQBAJ>.
- [25] T. Hastie, R. Tibshirani, e J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2001. cited By 36120.
- [26] Ian H. Witten, Eibe Frank, Mark A. Hall, e Christopher J. Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. 2016. Cited by: 3988. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85009962818&partnerID=40&md5=9d48eb8ad0e28bd4ad13ef43f6c5d085>.
- [27] M.I. Jordan e T.M. Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255 – 260, 2015. Cited by: 2414. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84937801713&doi=10.1126%2fscience.aaa8415&partnerID=40&md5=1d384806d6aad8d5de295df324a892aa>, doi:10.1126/science.aaa8415.
- [28] S.B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31(3):249 – 268, 2007. Cited by: 1435. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-36749047332&partnerID=40&md5=18e6b18aae6998616102de67f7ee9696>.
- [29] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, Outubro 2001. URL: <https://doi.org/10.1023/A:1010933404324>, doi:10.1023/A:1010933404324.
- [30] Leo Breiman. Random forests. *Machine Learning*, 45(1):5 – 32, 2001. Cited by: 56998; All Open Access, Bronze Open Access, Green Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-0035478854&doi=10.1023%2fA%3a1010933404324&partnerID=40&md5=4b9f43897146098c0df3a2af232cf2f4>, doi:10.1023/A:1010933404324.

- [31] Marco Tulio Ribeiro, Sameer Singh, e Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. volume 13-17-August-2016, página 1135 – 1144, 2016. Cited by: 4191. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84984985889&doi=10.1145%2f2939672.2939778&partnerID=40&md5=217b33ce367d1ea7001014a5ac694c79>, doi:10.1145/2939672.2939778.
- [32] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81 – 106, 1986. Cited by: 12257; All Open Access, Bronze Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-33744584654&doi=10.1023%2fA%3a1022643204877&partnerID=40&md5=48a99c15698f92848809769c74f260a1>, doi:10.1023/A:1022643204877.
- [33] Yan-Yan Song e Ying Lu. Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2):130 – 135, 2015. Cited by: 519. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84935075029&doi=10.11919%2fj.issn.1002-0829.215044&partnerID=40&md5=fa549e7098020b67f77a3961948be9c5>, doi:10.11919/j.issn.1002-0829.215044.
- [34] Mehdi Salehi. *Bayesian-based predictive analytics for manufacturing performance metrics in the era of industry 4.0*. KIT Scientific Publishing, 2019.
- [35] Irina Rish. An Empirical Study of the Naïve Bayes Classifier. *IJCAI 2001 Work Empir Methods Artif Intell*, 3, Janeiro 2001.
- [36] Stephan Dreiseitl e Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: A methodology review. *Journal of Biomedical Informatics*, 35(5-6):352 – 359, 2002. Cited by: 1082; All Open Access, Bronze Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-0043126911&doi=10.1016%2fS1532-0464%2803%2900034-0&partnerID=40&md5=7b819320e1f32d327ad7c9fdf174378f>, doi:10.1016/S1532-0464(03)00034-0.
- [37] Erin L. Allwein, Robert E. Schapire, e Yoram Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1(2):113 – 141, 2001. Cited by: 1280. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-24044435942&partnerID=40&md5=c90e0d303c0709a52e940aa99d2b7d43>.
- [38] Thorsten Joachims. Training linear svms in linear time. volume 2006, página 217 – 226, 2006. Cited by: 1390. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-33749563073&doi=10.1145%2f1150402.1150429&partnerID=40&md5=68aaf339d721a095322c63633e4aad1f>, doi:10.1145/1150402.1150429.
- [39] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, e Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825 – 2830, 2011. Cited by: 33229. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-80555140075&partnerID=40&md5=63e53cee7a9711760872d4d103e5453a>.

- [40] Zeeshan Ahmed, Saeed Amizadeh, Mikhail Bilenko, Rogan Carr, Wei-Sheng Chin, Yael Dekel, Xavier Dupre, Vadim Eksarevskiy, Senja Filipi, Tom Finley, Abhishek Goswami, Monte Hoover, Scott Inglis, Matteo Interlandi, Najeeb Kazmi, Gleb Krivosheev, Pete Lufrenko, Ivan Matantsev, Sergiy Matuskevych, Shahab Moradi, Gani Nazirov, Justin Ormont, Gal Oshri, Artidoro Pagnoni, Jignesh Parmar, Prabhat Roy, Mohammad Zeeshan Siddiqui, Markus Weimer, Shauheen Zahirazami, e Yiwen Zhu. Machine learning at microsoft with ml.net. página 2448 – 2458, 2019. Cited by: 25; All Open Access, Bronze Open Access, Green Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85071184378&doi=10.1145%2f3292500.3330667&partnerID=40&md5=e1bb0272582e94aaa86ec9c213084dde>, doi:10.1145/3292500.3330667.
- [41] Machine learning em ml.net. URL: <https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet>.
- [42] Linguagem Python, url = <https://www.python.org/>.
- [43] A. Ramanan, S. Suppharangsarn, e M. Niranjan. Unbalanced decision trees for multi-class classification. página 291 – 294, 2007. Cited by: 28; All Open Access, Green Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-51549089027&doi=10.1109%2fICIINFS.2007.4579190&partnerID=40&md5=f8ea27fa4a9df6a2dd3157132d09b009>, doi:10.1109/ICIINFS.2007.4579190.
- [44] Boutkhil Sidaoui e Kaddour Sadouni. Efficient approach one-versus-all binary tree for multiclass svm. *Lecture Notes in Electrical Engineering*, 275 LNEE:203 – 214, 2014. Cited by: 1. URL: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84958553028&doi=10.1007%2f978-94-007-7684-5_15&partnerID=40&md5=3cf609430b557bcdcd965cd9898a12aa, doi:10.1007/978-94-007-7684-5_15.
- [45] C.C. Aggarwal. *Outlier analysis*, volume 9781461463962. 2013. cited By 795. doi:10.1007/978-1-4614-6396-2.
- [46] Alexander Rothkopf. Improved maximum entropy analysis with an extended search space. *Journal of Computational Physics*, 238:106 – 114, 2013. Cited by: 36; All Open Access, Green Open Access. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-84873315705&doi=10.1016%2fj.jcp.2012.12.023&partnerID=40&md5=48704b8b63ba0ae56dc0954bf93de573>, doi:10.1016/j.jcp.2012.12.023.
- [47] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, e Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. volume 2017-December, página 3147 – 3155, 2017. Cited by: 2404. URL: <https://www.scopus.com/inward/record.uri?eid=2-s2.0-85038215940&partnerID=40&md5=e7f6611aa423276ab95bd73aa45ad93e>.