INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Serviço Preditivo para Equipamentos Industriais

ANDRÉ PEREIRA ASSUNÇÃO
Outubro de 2021

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Serviço Preditivo para Equipamentos Industriais

André Pereira Assunção

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Sistemas de Informação e Conhecimento

Orientador: Goreti Marreiros

Co-orientador: Marta Fernandes

Porto, Outubro 2021

Resumo

Com a constante e crescente digitalização dos negócios, as indústrias caminham no sentido da quarta revolução industrial, denominada de Indústria 4.0. Uma das tecnologias constituintes desta revolução é a Inteligência Artificial. O *Machine Learning* é um dos ramos da Inteligência Artificial, que combina dados e utiliza modelos computacionais para analisar padrões e efetuar previsões.

A Amorim Cork é uma empresa industrial destinada à produção de rolhas de cortiça, que está a implementar tecnologias e mecanismos para se aproximar cada vez mais do conceito de Indústria 4.0. Entre as várias fases do processo de produção, uma das mais críticas é a fase de limpeza e tratamento de rolhas. Nesta fase são eliminadas diversas bactérias, nomeadamente o TCA que é responsável pelo maior problema atualmente existente na cortiça.

Assim, a presente dissertação propõe um serviço preditivo para previsão de anomalias nas máquinas de limpeza e tratamento com uma antecedência de uma hora. Este serviço tem uma arquitetura *cloud* e utiliza métodos de *Machine Learning* para combinar os dados gerados pelas máquinas e prever anomalias nas mesmas. Com este serviço estima-se uma melhoria na eficiência do processo, reduzindo a manutenção reativa e propondo uma manutenção preditiva, tendo em vista uma redução de custos.

Primeiramente foi realizado um estado de arte do projeto. Neste foi feito um enquadramento do tema Inteligência Artificial, foram abordados diversos algoritmos, bem como a avaliação e seleção dos mesmos. Foram também elencadas plataformas de *Machine Learning* e trabalhos existentes no tema em que se insere a presente tese.

A implementação do serviço incluiu maioritariamente duas vertentes, o desenvolvimento do modelo preditivo e orquestração de dados. Para a orquestração foi utilizado o *Azure Data Factory*, já para o modelo preditivo foi utilizado *Azure ML*. O desenvolvimento do modelo compreendeu diversas tarefas de análise e exploração de dados, preparação de dados e avaliação de algoritmos. No final, após diversos ensaios, foi utilizado um modelo que utiliza validação cruzada estratificada e o algoritmo *Two Class Decision Forest*.

O serviço desenvolvido é robusto e altamente escalável, estando apto à introdução de novas fontes de dados para enriquecer o modelo preditivo e possuindo um elevado potencial de utilização industrial.

Palavras-chave: Indústria 4.0, Inteligência Artificial, *Machine Learning*, Previsão de Anomalias, Rolhas de Cortiça, TCA

Abstract

With the constant and growing digitalization of business, industries are moving towards the fourth industrial revolution, called Industry 4.0. One of the constituent technologies of this revolution is Artificial Intelligence. Machine Learning is one of the branches of Artificial Intelligence, which combines data and uses computational models to analyze patterns and make predictions.

Amorim Cork is an industrial company dedicated to the production of cork stoppers, which is implementing technologies and mechanisms to increasingly approach the concept of Industry 4.0. Among the various stages of the production process, one of the most critical is the cleaning and treatment of stoppers. At this stage, several bacteria are eliminated, namely the TCA, which is responsible for the biggest problem currently existing in cork.

Thus, this dissertation proposes a predictive service for anomaly prediction in cleaning and treatment machines. This service has a cloud-based architecture and uses Machine Learning methods to combine the data generated by the machines and predict failures and stops. With this service, it's estimated an improvement in the efficiency of the process, reducing reactive maintenance and proposing predictive maintenance, with the objective of reducing costs.

First, a state-of-the-art was carried out. In this, the Artificial Intelligence theme was framed, several algorithms were approached, as well as their evaluation and selection. Machine Learning platforms and existing works about the theme of this thesis were also listed.

The service implementation mainly included two aspects, the development of the predictive model and data orchestration. For the orchestration was used Azure Data Factory, while for the predictive model was used Azure ML. The development of the model included several tasks of data analysis and exploration, data preparation and algorithm evaluation. In the end, after several tests, was used a model that uses stratified cross-validation and Two Class Decision Forest algorithm.

The service developed is robust and highly scalable, being able to introduce new data sources to enrich the predictive model and having a high potential for industrial use.

Keywords: Industry 4.0, Artificial Intelligence, Machine Learning, Anomaly Prediction, Cork Stoppers, TCA

Índice

Índice de Figuras

Índice de Tabelas

Índice de Excertos de Código

Glossário

Anomalia	No contexto do presente documento, entende-se anomalia por falha ou paragem de uma MLT.
API	*"Application Programming Interface"* Conjunto de comandos, funções, protocolos e objetos que os programadores podem utilizar para criar *software* ou interagir com um sistema externo.
BI	*Business Intelligence*
CI/CD	*Continuous Integration/Continuous Deployment*
Cloud	A prática de utilizar uma rede de servidores remotos hospedados na Internet para armazenar, gerir e processar dados, em vez de um servidor local ou um computador pessoal.
Deploy	Processo de instalar uma aplicação num servidor.
Drag and drop	Mover um componente para outra parte do ecrã usando o rato ou dispositivo semelhante.
FAST	*Function Analysis system Technique*
IA	Inteligência Artificial
IOT	*Internet of Things*
ISEP	Instituto Superior de Engenharia do Porto
JSON	*JavaScript Object Notation* (JSON) é um formato baseado em texto para representar dados estruturados com base na sintaxe de objeto JavaScript.
ML	*Machine Learning*
MLT	Máquina de Limpeza e Tratamento
PaS	*Paltform as a Service*. Modelo de computação *cloud* que oferece *hardware, software* e infraestrutura para desenvolvimento e gestão de aplicações.

Pay as you go	Sistema de cobrança de custos à medida que surgem.
Pipeline	Fluxo de dados composto por diversos componentes, em que cada um deles possui uma tarefa específica.
Python	Linguagem de programação alto nível.
SCADA	*Supervisory control and data acquisition*
SparkSQL	Módulo *Spark* para processamento de dados estruturados.
SQL	*Structured Query Language*
TCA	2,4,6-Tricloroanisol
TOPSIS	*Technique for Order of Preference by Similarity to Ideal Solution*

1 Introdução

Atualmente o mundo assiste a uma digitalização da maioria dos negócios que proliferam no tecido empresarial público e privado [1]. Esta tendência denomina-se de Indústria 4.0 e é considerada a quarta revolução industrial, que surge da necessidade de melhorar os sistemas de produção e os modelos de negócio. Alguns dos conceitos chave desta revolução são *Big Data, Internet of Things* (IoT)*,* Inteligência Artificial (IA)*, Smart Factory, Cyber Security* e *Cloud Computing* [2]*.*

No que diz respeito ao conceito de IA aplicada a um qualquer processo de negócio, para que seja possível aplicar qualquer tipo de inteligência é necessário possuir dados. É nesta fase que surge o *Big Data*, armazenando grandes volumes estruturados de dados estruturados e não estruturados em *Data Lakes* e/ou *Data Warehouses*. No entanto, antes de possuir dados é necessário ter mecanismos para gerar os mesmos. É nesta fase que surge o *IoT* e assim sucessivamente. Ou seja, todos os conceitos chave da Indústria 4.0 complementam-se e formam assim a grande revolução industrial digital.

1.1 Âmbito

As empresas portuguesas, bem como o governo português, estão a acompanhar toda esta digitalização com agrado. Como tal, em 2017 o governo lançou uma estratégia nacional para a digitalização da economia, com o objetivo de agarrar a oportunidade que a Indústria 4.0 poderia traduzir para o país. Segundo o então primeiro-ministro português, António Costa, a Indústria 4.0 "é a primeira revolução industrial em que Portugal não parte em desvantagem" [3]. Isto porque Portugal possui "uma boa infraestrutura tecnológica de comunicações e, sobretudo, um conjunto de quadros altamente qualificados, universidades e politécnicos dinâmicos, e um tecido empresarial apto a receber o conhecimento" [3]. A presente dissertação integra-se com o cenário acima descrito, sendo que, esta gera conhecimento que a empresa Corticeira Amorim está apta para receber.

1.1.1 Indústria 4.0

Desde o fim do século 18 que o mundo sofre transformações industriais significativas, que justificam apelidá-las de revoluções industriais, ilustradas na Figura 1. Atualmente, o mundo encontra-se no seio de uma transformação significativa no que diz respeito à forma como os produtos são produzidos, isto graças à digitalização dos negócios e da indústria. O termo Indústria 4.0 serve para traduzir esta digitalização, ou seja, a quarta revolução indústria

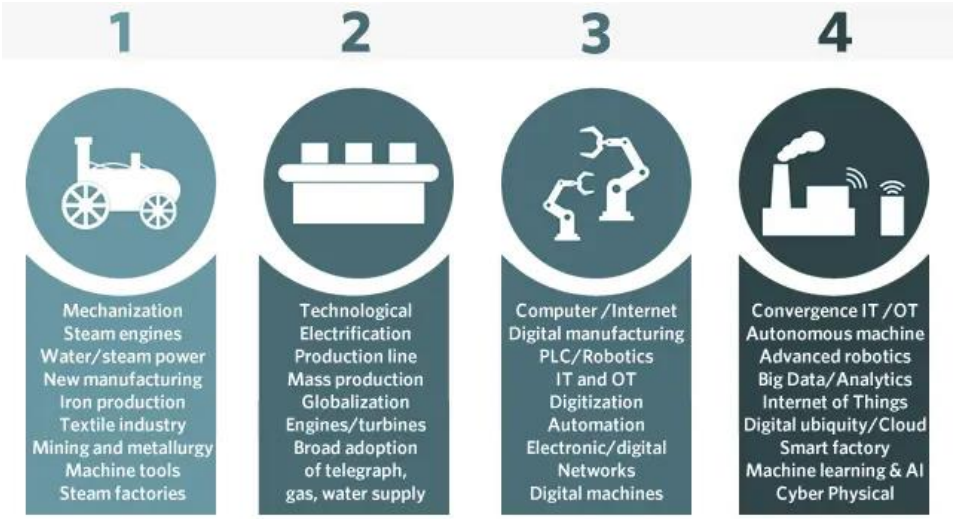


Figura 1 - Revoluções Industriais [4]

A primeira revolução industrial aconteceu no fim do século XVII, em 1765, e foi caracterizada pela mecanização. O fator impulsionador desta revolução foi a invenção da máquina a vapor, que motivou a criação de um novo tipo de energia e consequente extração em massa de carvão [5].

A segunda revolução industrial deu-se em 1870, no fim o século XIX. Esta foi caracterizada pela descoberta de novas fontes de energia, a eletricidade, o gás e o petróleo. Como resultado destas descobertas foi criado o motor de combustão interna que naturalmente deu espaço à invenção do automóvel e do avião. Um outro conjunto importante de invenções desta revolução, que permitiu revolucionar a forma de comunicação, são o telefone e o telégrafo [5].

Na terceira revolução industrial, que ocorreu em 1969 na segunda metade do seculo XX, foi descoberta mais uma fonte de energia, a energia nuclear. Assistiu-se também à invenção da eletrónica, do computador e da internet [5]. Invenções essas que tiveram a sua aplicação a nível industrial com o desenvolvimento de Controladores Lógicos Programáveis (PLC's) e robôs, dando assim início a uma era de automação de alto nível. A automação permitiu aumentar o tempo de atividade e a eficiência da produção, reduzindo os custos [6].

Atualmente o mundo está a presenciar a quarta revolução industrial estimulada pelo rápido crescimento da internet e os avanços tecnológicos na área dos sensores. Estes avanços permitem a conexão de objetos físicos com a finalidade de atingir objetivos coletivos [6]. O principal objetivo desta revolução industrial é conectar diversos paradigmas tecnológicos, como ilustrado na Figura 2 [6]. A Indústria 4.0 permite "aumentar e manter a eficiência operacional dos processos de produção atuais e fazer avançar as metodologias de automação atuais para novos patamares" [6]. Esta é composta por um conjunto de tecnologias e conceitos interligados. Os conceitos chave são a descentralização, a capacidade em tempo real, a interoperabilidade, a virtualização e a modularidade. Já as tecnologias chave são os sistemas ciber físicos, IoT, *big*

data, computação *cloud*, computação descentralizada, redes *wireless*, Inteligência Artificial, ciber segurança, entre outros.

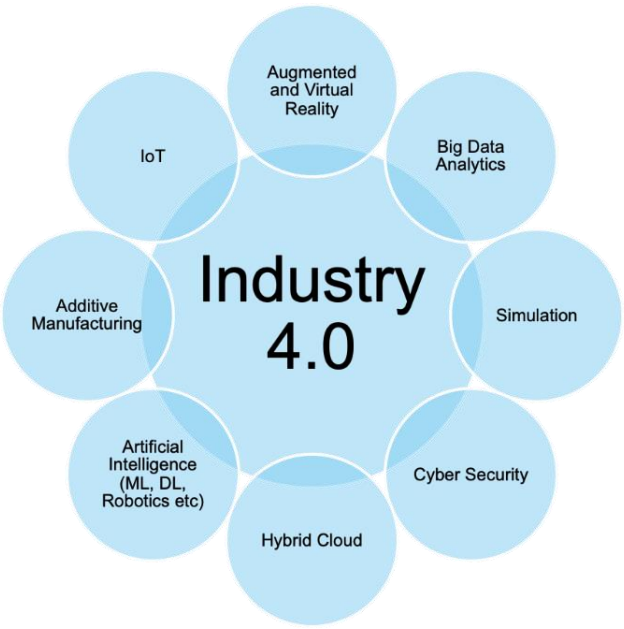


Figura 2 – Conceitos e tecnologias que formam a Indústria 4.0 [2]

1.1.2 Sistemas Ciber Físicos

No que aos sistemas ciber físicos diz respeito, estes podem ser definidos como "sistemas de automação industrial que são interconectados através de redes de comunicação" [14]. A ligação entre estes permite a realização de operações coletivas de vários equipamentos unitários, permitindo um aumento da produtividade. Uma outra característica desta ligação é que permite a troca de informações entre os diversos dispositivos físicos e a infraestrutura em tempo real, podendo assim ser possível obter informações em tempo real do que realmente está a acontecer na fábrica [6].

1.1.3 Internet of Things

O *IoT* é expansão da Internet para o domínio físico. Assim, os dispositivos de hardware individuais podem ser transformados em sistemas inteligentes operando de forma coletiva através da troca de dados pela Internet. Idealmente esta troca de informações deverá ocorrer através de redes wireless, o que aliás, é uma das tecnologias que faz parte do universo da indústria 4.0 [6].

1.1.4 Big data

A rápida adoção de sistemas ciber físicos de *IoT* culmina em enormes quantidades de dados brutos nos sistemas de armazenamento. Esses dados podem ser estruturados, semiestruturados e não estruturados e necessitam de ser processados e analisados em tempo real. É aqui que surge o problema *big data*. Tipicamente o *big data* é caracterizado segundo o modelo "5V" [7]. Este divide o problema em cinco dimensões, para melhor o conseguir explicar, sendo elas:

- Volume – com a crescente e rápida adoção de sistemas inteligentes, sensorização de máquinas e digitalização de negócios, os dados gerados por estes são de elevado volume e tendem a crescer cada vez mais ao longo do tempo [8];
- Velocidade – os dados são gerados a uma grande velocidade e pretende-se que sejam processados e analisados em igual velocidade, isto para gerarem valor em tempo útil [8];
- Variedade – os dados gerados ao longo do tempo e que necessitam de ser analisados, são provenientes de fontes heterogéneas. Estes não seguem um padrão e podem possuir diversos formatos, como por exemplo, *logs* aplicacionais, vídeos, documentos, imagens, entre outros [8];
- Veracidade – esta dimensão inclui dois aspetos, a consistência e a confiabilidade. A consistência refere-se aos dados históricos, por exemplo de uma organização, que podem já não corresponder à verdade atual dos factos, isto é, os processos podem ter sido alterados e ajustado ao longo do tempo. A confiabilidade diz respeito à autenticidade dos dados e à proteção contra acesso e alteração indevida dos mesmos [7];
- Valor – dados por si só não geram valor. É necessário analisar a que processos e/ou eventos eles se referem para perceber se estes podem trazer valor acrescido para o processo ou atividade [7].

Para o problema do *big data* existem uma série de tecnologias e processos que armazenam, processam e analisam estes enormes volumes de dados vindos diretamente da fábrica. O ecossistema da indústria 4.0 exige assim a troca constante de dados e como tal, "a falta de tecnologias de *big data* reduz a robustez e eficácia dos processos de troca de informações" [6].

1.1.5 Machine Learning

Uma das tecnologias que permite que a indústria 4.0 ganhe espaço nos negócios e no "chão de fábrica" é a IA, nomeadamente o *Machine Learning (ML)*. Isto porque, através deste, é possível por exemplo detetar erros, prever falhas e avaliar a qualidade dos produtos em tempo real. Este facto otimiza o processo de produção, traduzindo-se assim num ganho de eficiência e numa redução de custos, uma vez que o desperdício por erros é diminuído e aumenta-se a continuidade do processo de produção sem falhas. A adoção de ML só é possível depois da

adoção do *big data* e consequentemente do *IoT* e sistemas ciber físicos, uma vez que são estes que produzem os dados em estado bruto e que processam em grande escala os mesmos [9].

Os modelos de ML, ao aprenderem com os dados históricos e ao receberem dados em tempo real da fábrica, devem ser autossuficientes na tomada de decisão não sendo necessário a intervenção humana. Esta autossuficiência é possível graças à computação descentralizada. As decisões autónomas devem ser comunicadas aos sistemas ciber físicos, traduzindo-se assim num ciclo de *feedback* no qual "a melhoria contínua da produção pode ser sustentada através de treino contínuo" [6]. Assim, forma-se então o ecossistema da Indústria 4.0 totalmente integrado que é capaz de responder e se adaptar às mudanças em tempo real no processo de produção [6].

1.2 Problema

Portugal é o país líder mundial no mercado de cortiça. A cortiça é a casca do sobreiro que, por sua vez, é um tecido vegetal 100% natural. Num único centímetro cúbico de cortiça contam-se quase 40 milhões de células – cerca de 800 milhões numa só rolha. A produção de rolhas de cortiça inclui diversas fases de processo, sendo uma destas a fase de limpeza e tratamento. Nesta fase são eliminadas diversas bactérias e fungos das rolhas, nomeadamente o 2,4,6 – Tricloroanisol (TCA). Este é responsável pelo maior problema atualmente existente no mundo dos vinhos, uma vez que induz um odor a mofo no vinho. Deste modo, uma vez presente, este tem a capacidade de deteriorar qualquer vinho e assim representar perdas de qualidade de produto e monetárias. Desta forma, a irradicação desta bactéria é de extrema importância. Inclusive o cliente comprometeu-se, em meados de 2018, em "eliminar o TCA até 2020" [10].

Para resolver o problema do TCA foi implementado um processo de limpeza e tratamento de rolhas, seguido da análise das mesmas. Na fase de limpeza e tratamento, as rolhas passam por um sistema[1], que é constituído por um total de 18 máquinas, denominadas de Máquina de Limpeza e Tratamento (MLT). Estas aplicam o conceito de dessorção térmica, que faz uso da energia térmica para libertar mais de 150 compostos voláteis, um dos quais o TCA. Cada conjunto de rolhas que entra na MLT sofre este processo durante um determinado tempo e a isto dá-se o nome de ciclo.

A MLT possui sensores de temperatura e pressão em vários pontos de controlo da máquina. Todas as 18 máquinas estão ligadas entre si e são controladas através de um sistema SCADA[2], ilustrado na Figura 3. Neste sistema, estão presentes informações ao longo do tempo sobre a temperatura e pressão das máquinas nos vários pontos de controlo, ciclos e etapas de tratamento, alarmes, falhas, entre outros.

[1] Por motivos de confidencialidade, este processo não será detalhado.
[2] Por motivos de confidencialidade, os dados foram ocultados.

Figura 3 - Representação das MLT no SCADA[2]

Após o término de cada ciclo, as rolhas necessitam de repousar durante algum tempo e terão de ser descarregadas para análise. Tempos elevados na descarga poderão ser prejudiciais, levando por exemplo a que as rolhas fiquem queimadas, como tal são denominados de ciclos anormais. Na fase de análise, as rolhas entram individualmente num sistema desenvolvido pelo cliente em parceria com outras entidades para detetar a presença do TCA na rolha. Este sistema possui uma taxa de acerto de 99% e é composto por um conjunto de algoritmos de ML que detetam a presença de TCA. Os resultados desta análise são enviados para a equipa responsável pelo controlo do processo de limpeza e tratamento, onde são posteriormente inseridos em ficheiros *excel*. Atualmente a correlação entre os dados de limpeza e os dados de análise é feita através da ingestão dos dados do SCADA e dos ficheiros *excel* num relatório em PowerBI.

Como indicado, a fase de limpeza e tratamento de rolhas é uma fase crítica e impactante no negócio, uma vez que esta está diretamente ligada à qualidade do produto final. Esta fase gera uma grande quantidade de dados, que, à data, não está a ser convertida em conhecimento. Isto é, não está a ser retirado todo o potencial dos dados, para se entender e/ou melhorar particularidades do processo, nomeadamente a previsão de anomalias. No presente documento uma anomalia designa-se como uma falha e paragem nas MLT. Sendo assim, surge a necessidade de utilizar os dados gerados pelo negócio, para assim ser possível prever anomalias nas máquinas MLT. Posto isto, pretende-se explorar técnicas de IA e utilizar as mesmas para combinar os dados de forma que a informação, em estado bruto, possa trazer conhecimento e inteligência ao processo. Para resolver esta questão pretende-se desenvolver um serviço preditivo que preveja anomalias nos equipamentos, extraindo assim conhecimento dos dados gerados pelos mesmos. Assim, este serviço possibilitará o aumento da eficiência do processo, reduzindo os de custos inerentes às manutenções reativas e minimizando as falhas na eliminação de bactérias ou perdas de qualidade no produto.

1.3 Objetivos

O objetivo do presente projeto é extrair conhecimento dos dados gerados pelos sensores existentes nos equipamentos de limpeza e tratamento de rolhas cortiça, recorrendo a técnicas de ML. Desta forma será possível melhorar esta fase do processo de produção, tentando aumentar a eficiência das máquinas e diminuir os custos. Ou seja, pretende-se desenvolver um serviço de previsão de anomalias em equipamentos industriais. Um outro objetivo será estudar e implementar diferentes abordagens de ML, que possam ser aplicadas à previsão de anomalias. Para tal, será necessário pesquisar e comparar algoritmos de ML. Após uma avaliação criteriosa dos resultados da comparação, devem ser implementadas as técnicas mais adequadas para o problema em estudo num serviço preditivo. Este, por sua vez, deverá ser consumido por outros sistemas interessados em adquirir as previsões efetuadas. No que diz respeito aos objetivos específicos do presente projeto, estes são:

- Revisão de literatura e sintetização de trabalhos realizados no âmbito da indústria 4.0, manutenção preditiva e soluções de ML aplicadas no contexto industrial.
- Extração de conhecimento a partir dos dados brutos gerados pelos sensores de máquinas de limpeza, adotando uma abordagem exploratória dos dados das mesmas utilizando técnicas de ML.
- Avaliação e seleção dos métodos de aprendizagem mais adequados, para resolver o presente problema.
- Desenvolvimento de um modelo preditivo para previsão de anomalias, nos equipamentos de limpeza e tratamento, utilizando os métodos de aprendizagem previamente selecionados. Este deverá utilizar um ciclo de desenvolvimento bem definido.
- Treino, teste e validação do modelo que deverá ser criteriosamente avaliado, tendo em conta a sua precisão e performance.
- Desenho e implementação de um serviço preditivo que implementa o modelo e que permita a adoção do mesmo em ambiente industrial. Este deverá ser consumido por outros sistemas.
- Testes de integração do serviço preditivo.
- Avaliação e análise dos resultados.

1.4 Resultados Expectáveis

A presente dissertação propõe um serviço preditivo para um processo de produção de rolhas de cortiça. No final da mesma é expetável um conjunto de contribuições para o cliente, área de negócio e área tecnológica, nomeadamente:

- Um estado de arte de *Machine Learning*;

- Um estudo dos vários algoritmos e abordagens de ML de previsão de anomalias;
- Um serviço preditivo para previsão de anomalias na fase de limpeza de rolhas de cortiça;

1.5 Metodologia de Trabalho

O produto final da presente dissertação é um serviço preditivo. Este distribui-se em dois âmbitos de desenvolvimento distintos, uma vez que é necessário desenvolver o modelo preditivo e o serviço que alberga o mesmo.

No desenvolvimento do serviço será seguido do ciclo de vida de desenvolvimento de sistema ágil, que assenta sobre a metodologia *Agile*. Esta surgiu em 2001 e é composta por um conjunto de princípios e valores que se encontram compilados no *Agile Manifesto* [11]. O ciclo citado e ilustrado na Figura 4 compreende as seguintes fases:

1. Iniciação (*Initiation*) – nesta fase, também apelidada de conceção, é realizada uma avaliação geral do projeto, identificando o negócio do mesmo. Posto isto, inicia-se a discussão da viabilidade do projeto, não sendo aprofundados detalhes específicos. São também identificados os membros constituintes da equipa para determinar o tempo e recursos de trabalho necessários [12].
2. Planeamento (*Planning*) – aqui são levantados e analisados os requisitos funcionais e não funcionais em conjunto com as partes interessadas do projeto. Seguidamente são definidas as prioridades e dependências entres estes para a elaboração de um *backlog* de tarefas [12].
3. Desenvolvimento (*Development*) – ao longo desta fase a equipa trabalha iterativamente na entrega das tarefas em *backlog*, sob a forma de *sprints*, cumprindo prioridades e dependências entre requisitos. Ao longo da *sprint* todo o desenvolvimento deverá ser continuamente testado. No final de cada *sprint* é espectável existir um produto funcional e utilizável que é disponibilizado às partes interessadas. Com o desenrolar das várias *sprints* o produto ficará mais robusto e completo, até que termine o desenvolvimento do projeto e este passe em todos os testes de qualidade e aceitação [12].
4. Lançamento (*Release*) – fase na qual o produto final está disponibilizado aos utilizadores finais e torna-se necessário monitorizar o mesmo e providenciar suporte continuado [12].
5. Aposentadoria (*Retirement*) – nesta fase o produto encontra-se em fim de vida e deixa de usufruir de suporte continuado. Tipicamente acontece quando existe uma versão superior em produção [12].

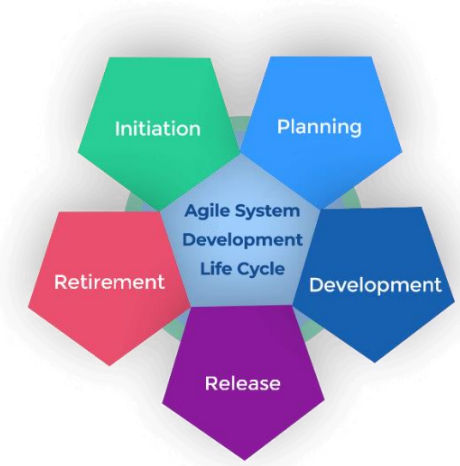


Figura 4 – Fases do Ciclo de Vida de Desenvolvimento de Sistema Ágil [12]

Para o desenvolvimento do modelo preditivo, e consequente a extração de conhecimento, deverá ser adotado um processo bem definido. Tal como na engenharia de software existe um processo e um ciclo de vida bem definido, na gestão de dados em ML também existe. Esse ciclo é oriundo da *framework* CRISP-DM [13], que é considerada um padrão na mineração de dados e é composta por seis etapas, como identificado na Figura 5.

A primeira etapa diz respeito à compreensão do negócio. Nesta fase deverá ser feita uma análise aos processos de negócio, o problema deverá ser identificado e analisado, bem como deverão ser delineados os objetivos e critérios de sucesso do projeto [13].

Na segunda fase deverá ser realizada a exploração e compreensão do *dataset* com recurso a métodos estatísticos e visualização de dados. Os dados deverão ser interpretados de forma que seja possível identificar o significado de cada atributo, os atributos chave e os dados em falta e/ou irrelevantes [13].

Depois da exploração de dados, é necessário pré-processar os dados. Nesta fase são realizadas maioritariamente transformações de seleção e limpeza de dados, uma vez que nem todos estes poderão ser uteis para a resolução do problema e poderá existir um certo ruido nos mesmos, por exemplo um atributo de estado civil incluir dados “999”. Esta etapa também é caracterizada por outras transformações como agregações, normalizações, entre outras [14].

Na fase de modelação será desenvolvido o modelo utilizando técnicas de ML e seguidamente cada modelo será avaliado na fase de avaliação.

Por fim, após terem sido avaliados, serão selecionados os modelos que melhor respondem aos critérios de sucesso do projeto e contêm a melhor precisão para serem implantados.

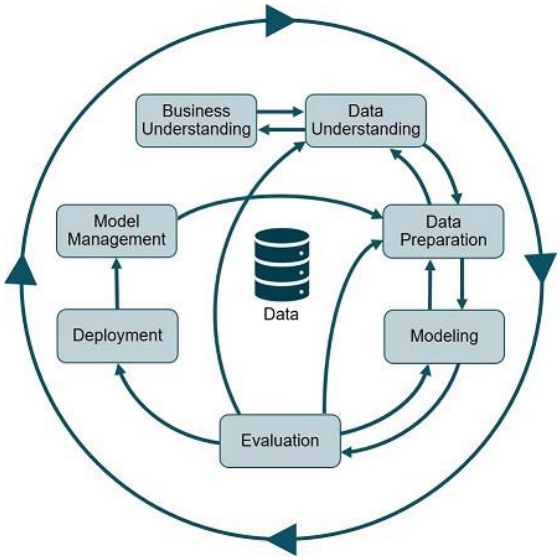


Figura 5 - Ciclo CRISP-DM [14]

1.6 Estrutura do Documento

O presente documento encontra-se organizado segundo a seguinte estrutura de capítulos:

Introdução, onde é definido o âmbito do presente projeto e no qual são esclarecidos alguns conceitos importantes para compreensão total do presente documento. Neste capítulo é também realizada uma descrição do problema, dos objetivos, dos resultados expectáveis do projeto e identificada a metodologia de trabalho.

Estado de Arte, neste capítulo é efetuada uma análise do atual estado da arte do projeto, onde é realizada uma resenha histórica da Inteligência Artificial, são analisados algoritmos e tecnologias de *Machine Learning*. São também abordados trabalhos relacionados com a presente dissertação.

Análise de Valor, onde é realizada a análise de valor do projeto. Nesta é identificado o cliente, enquadrado o processo de inovação e a oportunidade do projeto e é detalhado e analisado o conceito de valor. Posto isto, é realizada uma proposta de valor e efetuada uma análise FAST e TOPSIS.

Desenho da Solução, onde é proposta e desenhada uma arquitetura para resolver o referente problema e são especificados os principais requisitos funcionais e não funcionais.

Implementação, capítulo no qual é demonstrada a implementação da solução desenhada, seguindo uma estrutura assente em: Compreensão dos Dados, Preparação dos Dados, Pipelines e Orquestração de Dados e Teste.

Experimentação e Avaliação, neste capítulo são efetuados ensaios e experimentações, de acordo com uma estratégia de avaliação, com o intuito de avaliar o modelo de ML. Posto isto, são detalhados e analisados os resultados alcançados e selecionado o melhor modelo de ML.

Conclusões, onde é feita uma apreciação global do trabalho, são apresentados os objetivos realizados e identificadas limitações da solução abordada que poderão ser alvo de melhoria futura.

2 Estado de Arte

No presente capítulo será descrito o estado de arte do projeto. Este encontra-se dividido em cinco secções: 1) Inteligência Artificial; 2) Algoritmos de ML; 3) Critérios de escolha de algoritmos; 4) Plataformas de ML; 5) Trabalhos relacionados.

Na primeira secção será feito um enquadramento do tema Inteligência Artificial desde que surgiu até à atualidade. Adicionalmente, também é explorado um dos ramos de IA, o *Machine Learning*.

Na segunda secção são abordados um conjunto algoritmos. Estes são devidamente explicados, seguindo uma metodologia de categorização pelo seu tipo de aprendizagem. Ao longo desta secção são também abordados conceitos de extrema importância para a compreensão do algoritmos e medidas de avaliação, secção seguinte. A escolha dos algoritmos foi realizada após uma intensa pesquisa de artigos sobre "*Machine Learning*", "Previsão de Falhas", "Manutenção Preditiva", "Indústria 4.0" e "Serviços Preditivos para Equipamentos Industriais" [39], [40], [41], [42], [43], [44], [45], [46], [47].

A terceira secção aborda a avaliação e seleção de algoritmos. Ou seja, são enunciadas e explicadas algumas medidas de avaliação, divididas por tipo de avaliação, e é explicado com se procede à seleção de algoritmos. Isto é particularmente interessante, uma vez que um dos objetivos da presente dissertação passa por identificar e selecionar um ou vários algoritmos de ML para resolver o problema da mesma.

A quarta secção é relativa ao estudo de três plataformas para o desenvolvimento de soluções ML. A escolha das ferramentas abordadas foi realizada após investigação de quais as mais utilizadas atualmente [42], [39].

Por fim, na última secção são abordados alguns trabalhos realizados na área de ML em ambiente industrial e ou no âmbito da Indústria 4.0. Através dos trabalhos realizados no passado poderá ser possível encontrar respostas, ou ajudar a responder ao problema da presente dissertação.

2.1 Inteligência Artificial

Atualmente não há uma definição universalmente aceite para o termo IA. No entanto este é um termo abrangente para muitos outros conceitos, como, *Machine Learning*, redes neuronais ou computação cognitiva. Segundo Stuart Russell em "*Artificial Intelligence: A Modern Approach"* [15], IA pode definir-se como um sistema que:

- Pensa como um humano - por exemplo, arquiteturas cognitivas e neuronais;
- Age como um humano - por exemplo, passa no teste de Turing;
- Pensa racionalmente - por exemplo, solucionadores lógicos;

- Age racionalmente - por exemplo, software inteligente que cumpre os objetivos de forma planeada, com raciocínio e através de treino.

Uma determinada pessoa pode ser especialista numa área ou num conjunto restrito de áreas e certamente conseguirá resolver um problema complexo nessas mesmas áreas. Uma vez que o ser humano possui capacidade de raciocínio, este conseguirá também arranjar sugestões alternativas de solução para as áreas que não domina, mesmo que não sejam as melhores. No entanto, esta pessoa não conseguirá resolver um problema complexo na área de conhecimento para a qual não foi formada. À semelhança do ser humano, o grande propósito da IA é ser "um sistema único que pode aprender e resolver qualquer problema apresentado" [16]. Isto, é exatamente aquilo que um humano, com todas as suas faculdades racionais e psíquicas, faria. No entanto, convém notar que neste momento a IA está longe do propósito enunciado e que um sistema de IA "não significa que um algoritmo de IA ou ML pense como um humano" [49]. Pelo contrário, os algoritmos de IA ou ML apelas replicam aquilo para o qual foram treinados.

Posto isto, um sistema de IA combina diversos métodos de análise de dados, para possuir inteligência, como exemplificado na Figura 6. Segundo Munakata em "*Fundamentals of the New Artificial Intelligence*" [50], o que define se um sistema é inteligente é o tipo de problemas que este resolve. Ou seja, sistemas que resolvem problemas de inferência baseados no conhecimento, de previsão, classificação e otimização são considerados inteligentes. A título de exemplo, um sistema de IA que realiza reconhecimento de voz, utiliza *Machine Learning* para aprender, que por sua vez utiliza métodos que lhe atribuem a capacidade de classificar essa mesma voz. Por fim, este sistema utiliza a tecnologia para tornar a inteligência visível, isto é, neste caso os sensores captam o som, os algoritmos de ML classificam o som e o autofalante emite o resultado do sistema.

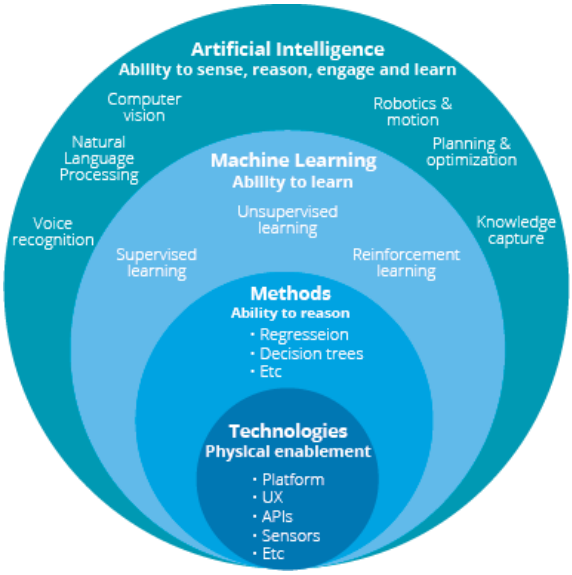


Figura 6 - Terminologia da Inteligência Artificial [16]

2.1.1 A evolução

A Inteligência Artificial é uma tecnologia que o Homem já explora há largos anos, mesmo sem saber o que esta seria. Está provado que na era da Grécia antiga já existiam mitos sobre robots e que a população Chinesa e Egípcia antiga já possuía a ideia de dar vida e inteligência a objetos inanimados [17].

Ao longo do tempo, diversas ideias foram escritas, diversos projetos foram desenvolvidos e testados. Destacando-se o nome de Charles Babbage que, em 1884, construi uma máquina cujo objetivo seria apresentar um comportamento inteligente. De facto, esta máquina era inteligente e fazia cálculos complexos, no entanto Babbage chegou à conclusão que "não seria capaz de produzir uma máquina que exibisse comportamentos tão inteligentes quanto um ser humano" [17].

O aparecimento oficial de IA surgiu em 1956 com a conferência "Artificial Intelligence" que ocorreu em *Dartmouth College* no Estados Unidos da América no estado de New Hampshire. Esta conferência foi organizada por um professor assistente de Matemática de *Dartmouth College*, John McCarthy, que disse "na época, eu acreditava que se conseguíssemos reunir todos os interessados no assunto para se dedicarem a ele e evitar distrações, poderíamos fazer um progresso real" [18]. Nesta conferência participaram alguns cientistas do MIT e um colega de faculdade de McCarthy, Marvin Minsky que escreveu no seu livro "*Stormed Search for Artificial Intelligence*" que "o problema da modelação da Inteligência Artificial será resolvido dentro de uma geração" [18]. Após esta conferência, os estudos e investigações sobre IA tiveram um novo rumo, existindo já um termo para aquilo que a humanidade procurava com a robotização e tornar objetos inanimados inteligentes. Todavia, apenas a partir dos anos 80 é que a IA foi amplamente utilizada em projetos reais.

Mais próximo da atualidade, em 2005, a empresa Honda Motor Co., Ltd lança o primeiro robô humanoide, com habilidades humanas, como a capacidade motora. Este, graças um conjunto de sensores visuais, sensores de superfície do piso, sensores ultrassônicos e sensores cinestésicos, consegue agir em sincronia com as pessoas, correr e transportar objetos [51]. Em 2010 este robô já conseguia agir usando o "poder da mente" [17], uma vez que conseguia "compreender e responder a comandos de voz simples, reconhecer rostos e até mesmo evitar obstáculos em movimento" [52]. Passados cinco anos, em 2015, a empresa americana Hanson Robotics desenvolveu a *Sophia*, um robô humanoide com feições humanas que tem a capacidade de aprender comportamentos humanos pela interação com as pessoas. Este humanoide possui IA utilizando algoritmos preditivos com base em estatísticas computacionais e um processamento rápido das informações que recebe e uma ampla capacidade de reconhecer rostos e vozes [19].

Por fim, em 2020, a empresa chinesa Cheetah Mobile Inc lançou uma solução para auxiliar no combate à pandemia Covid 19 que se instalou em todo o mundo, composta por um conjunto de três robôs. Segundo o presidente da empresa, Fu Sheng, os robôs são orientados pela IA e "podem oferecer aconselhamento médico, entregar medicamentos, orientar rotas ou medir a temperatura dos pacientes, o que pode ajudar a aumentar a eficiência e reduzir a infeção

cruzada entre os profissionais da área médica" [20]. Esta solução é composta pelos robôs *Smart GreetBot, Smart DeliverBot* e *Smart ThermoBot.* Os primeiros têm como função interagir, maioritariamente por voz, com os pacientes fornecendo aconselhamento médico e respondendo a questões por exemplo sobre o seu estado de saúde [21]. Os segundos conseguem efetuar entregas, por exemplo de medicamentos, possuindo mais de 10 000 metros de alcance [21]. Por último, os *Smart ThermoBots* têm como funcionalidade efetuar medições de temperatura sem contacto físico. Estes conseguem efetuar 5 medições por segundo, com uma precisão de ± 0,3°C [21]. Esta solução foi implementada no Hospital Huoshenshan de Wuhan, o hospital de campo construído em dez dias.

2.1.2 Machine Learning

ML é um ramo da IA, como ilustrado na Figura 6 e pode ser definido como "o conjunto de métodos computacionais que usam a experiência para melhorar o desempenho ou para fazer previsões mais precisas" [22], segundo o livro "*Foundations of Machine Learning*". Sendo que, neste caso, a experiência refere-se às informações do passado que assumem a forma de dados recolhidos de sistemas. O tamanho e a qualidade destes são fulcrais para o sucesso das previsões.

Com efeito, o sucesso de um algoritmo de ML depende essencialmente do tamanho da amostra de treino e da complexidade dos treinos. Assim, como isto está inteiramente relacionado com os dados utilizados, seja pela sua qualidade seja pela complexidade, pode-se afirmar que as técnicas de aprendizagem são métodos orientados por dados. Métodos esses que combinam conceitos fundamentais da ciência de computação com ideias de estatística, probabilidade e otimização [22].

2.2 Algoritmos de Machine Learning

Nesta secção serão abordados diversos algoritmos de ML. Uma vez que estes podem ser classificados pelo seu estilo de aprendizagem e pelas suas semelhanças de funcionamento [53], serão então explicadas as duas formas de classificação de algoritmos. Ao longo da explicação da classificação por semelhanças, para cada uma delas será exemplificado com um algoritmo.

2.2.1 Aprendizagem Supervisionada

A aprendizagem supervisionada é a aplicação do conceito de "aprender a partir de exemplos" [54]. A partir da exemplificação rotulada do que se pretende identificar, o algoritmo deverá aprender a rotular determinados problemas. Neste tipo de aprendizagem existem dois tipos *datasets*, o de treino e o de teste.

O *dataset* de treino contém evidencias históricas de problemas rotulados e serve para o algoritmo aprender a identificar os esses mesmos problemas sem estarem rotulados. Ou seja,

este *dataset* permite ao algoritmo identificar um padrão e desenvolver uma série de regras, o modelo, para identificar os problemas em causa sem estarem perfeitamente catalogados. Segundo Learned-Miller, o *dataset* de treino é composto por um conjunto "de *n* pares ordenados $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$, onde cada x_i é uma medida ou conjunto de medidas de um único ponto de dados de exemplo e y_i é o rótulo para esse ponto de dados" [54].

Por outro lado, o *dataset* de teste contém exemplos do mesmo tipo de treino, mas sem estarem catalogados. Este irá permitir aferir se o modelo está bem treinado, isto é, se aprendeu a identificar os problemas corretamente e com que *accuracy* [54].

O objetivo deste tipo de aprendizagem é maximizar a *accuracy* preditiva de novos dados não rotulados e não nos dados de treino. No entanto, com o esforço excessivo para atingir este objetivo corre-se o risco de esta *accuracy* contemplar o ruído existente nos dados e assim memorizar várias particularidades dos mesmos. Ou seja, ao invés de se criar um conjunto de regras gerais de previsão (capacidade de generalização), estas estão demasiado focados nos dados de treino. A este fenómeno dá-se o nome de *overfitting* [55].

O enviesamento (*bias)* e a variância são outros dois conceitos pertinentes de abordar. O enviesamento corresponde à diferença entre a previsão média do modelo e o valor correto. Ao passo que, a variância corresponde à dispersão da previsão do modelo para um determinado ponto de dados. O *overfitting* acontece quando existe um baixo enviesamento e uma alta variância. O objetivo de um qualquer modelo é possuir um baixo enviesamento e uma baixa variância [56].

Este tipo de aprendizagem pode ser dividido em duas categorias de dados diferentes, dados discretos e dados contínuos [57]. Para lidar com os dados discretos, para por exemplo classificar se um *e-mail* é *spam* ou não, é utilizado o problema de classificação. Se existirem duas classes é chamada de classificação binária e se existirem várias, é chamada de classificação multi-classe. Já para lidar com os dados contínuos, para por exemplo prever a faturação de uma empresa, é utilizado o problema regressão. Na regressão, se as variáveis de entrada estiverem ordenadas em função do tempo este é denominada de série temporal.

2.2.1.1 K Nearest Neighbor (KNN)

O KNN, ilustrado na Figura 7, é um algoritmo de aprendizagem supervisionada utilizado para problemas de classificação e regressão. Este algoritmo utiliza todos os dados de treino e classifica um novo ponto de dados, com base na classe da maioria dos seus *k* vizinhos mais próximos [58]. Estes vizinhos mais próximos são dados pela distância euclidiana, identificada abaixo como $E(x, y)$, entre o novo registo e os vizinhos deste. Quanto maior a distância entre os pontos de dados mais diferentes eles são e se esta for zero, então eles são exatamente iguais [54].

$$E(x, y) = \sqrt{\sum_{i=0}^{n} (x_i - y_i)^2}$$

O KNN é mais indicado para *datasets* de treino reduzidos, isto porque na fase de teste estes estão continuamente a ser utilizados para procurar os vizinhos mais próximos [59].

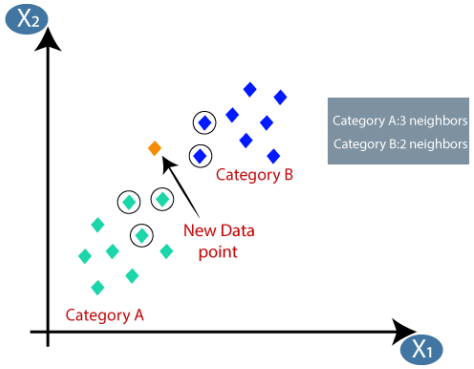


Figura 7 - K *Nearest Neighbor* [60]

2.2.1.2 Naive Bayes

O *Naive Bayes* é um algoritmo probabilístico que pressupõe que as variáveis de entrada são independentes. Este é baseado no teorema de *Naive Bayes* e pode ser utilizado para problemas de classificação. Como classificador este algoritmo cria redes *Bayesianas*, que são árvores, geradas com base na probabilidade condicional de ocorrer um resultado [57]. A fórmula utilizada para criar essas árvores encontra-se ilustrada na Figura 8. Este algoritmo possui o nome de ingénuo, porque assume que a presença de um determinado recurso numa classe não está relacionada com a presença de outro qualquer recurso.

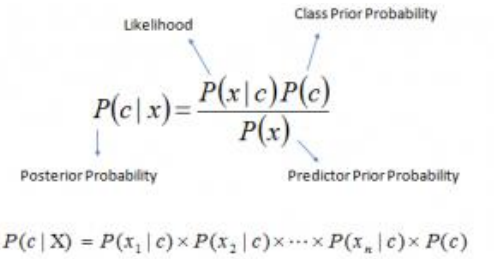


Figura 8 - Fórmula do Classificador *Naive Bayes* [61]

2.2.1.3 Árvores de Decisão

Os algoritmos de árvores de decisão utilizam aprendizagem supervisionada e podem ser usados para problemas de classificação e regressão. Este algoritmo passa pela construção, na fase de treino, de um modelo de decisões tomadas com base nos valores reais de atributos nos dados. Neste algoritmo, o conhecimento extraído dos dados é organizado hierarquicamente numa estrutura recursiva composta nós e ramos, com base nos atributos mais significativos e/ou variáveis independentes. Cada nó interno representa um atributo e está associado a um teste relevante para a classificação dos dados. Os nós folha representam as classes. Por sua vez, cada ramo representa um resultado possível de teste [58].

Esta técnica visa maximizar a classificação correta de todos os dados. Para evitar o *overfitting* e otimizar a eficiência computacional é realizada uma poda (*prunning*) [58]. Esta pode ser aplicada à árvore antes (*pre-prunning*) e depois (*post-prunnig*) de treinada. O *pre-prunnig* consiste na interrupção do crescimento da árvore, com base num conjunto de critérios de paragem. Este tipo de poda tem a vantagem de não gerar a árvore inteira. O *post-prunning* passa por deixar a árvore entrar em *overfitting* e logo de seguida aplica-se um corte às ramificações com pouco poder expressivo, de acordo com um critério predefinido. Os nós cortados passam a pertencer à classe com maior semelhança [62].

Uma vantagem deste tipo de algoritmo é a compreensibilidade das estruturas geradas, podendo assim ser possível rastrear o motivo da classificação. No entanto, para dados de entrada com um elevado número de atributos esta técnica pode não ser robusta [58].

2.2.1.4 Random Forest

O *Random Forest* (RF) é um algoritmo de aprendizagem supervisionada utilizado para problemas de classificação e regressão. Este é um exemplo de um algoritmo que utiliza a técnica *ensemble*. A técnica *ensemble* combina vários modelos base para produzir um modelo ideal. Cada um dos modelos base efetua uma previsão e posteriormente os resultados são combinados e integrados por forma a ser realizada a previsão final. Esta agregação de vários modelos geralmente aumenta a complexidade do algoritmo final, mas pode-se traduzir numa maior *accuracy* do mesmo [56].

Na Figura 9 é possível visualizar um exemplo do RF e comprovar que este utiliza a técnica *ensemble*, uma vez que este é composto por várias árvores de decisão. Neste algoritmo são gerados aleatoriamente vários subconjuntos a partir do *dataset* original. Sendo T um *dataset* de treino com *n* registos e onde cada registo tem *m* atributos. Para cada árvore, um novo *dataset* de treino T0 é construído amostrando T aleatoriamente com substituição (amostragem *bootstrap*). A previsão final, em problemas de regressão é dada pela média das previsões de todas as árvores de decisão e no caso de problemas de classificação pela moda [58].

Geralmente, este algoritmo possui uma velocidade de treino rápida, é robusto a ruido nos dados, é escalável, tem uma implementação simples e quanto mais árvores de decisão existirem menor será o erro [63]. Apesar da possibilidade de serem utilizados grandes números de árvores, não existe o problema de *overfitting* [58]. No entanto, à medida que o número de árvores utilizadas cresce, este algoritmo torna-se lento para previsões em tempo real [63].

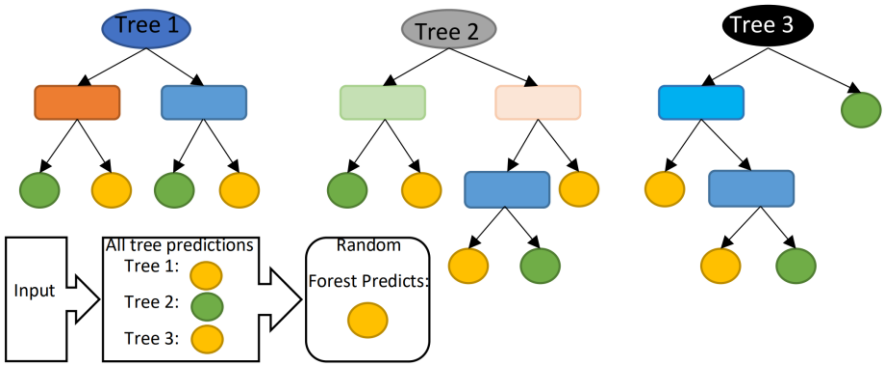


Figura 9 - Exemplo de Random Forest [39]

2.2.1.5 Redes Neuronais Artificiais

As redes neuronais artificiais (RNA) são técnicas computacionais inspiradas nos neurónios humanos. Estas utilizam aprendizagem supervisionada e servem para resolver problemas de regressão e classificação. As RNA consistem num grande número de processadores simples com muitas conexões, designados de nós, que ao invés de seguirem um conjunto de regras especificadas, aprendem leis básicas a partir exemplos simples [58].

As RNA estão organizadas em três ou mais camadas e cada uma delas possui diversos nós interligados, como exemplificado na Figura 10. A camada de entrada tem como objetivo receber os valores dos atributos explicativos de cada observação e possui tantos nós quantas variáveis explicativas [64].

A camada oculta, que podem ser várias, aplica as transformações aos valores de entrada na rede. Nesta, existem arcos que vêm de outros nós ocultos ou de nós de entrada e se ligam posteriormente aos nós de saída ou a outros nós ocultos. Aqui, o processamento é feito através de um sistema de conexões ponderadas, sendo que os valores que entram num nó oculto são multiplicados por pesos. As entradas ponderadas são então somadas para produzir um único número e comunicar à camada seguinte [64].

A camada de saída encontra-se ligada a uma camada oculta e retornam um valor de saída que corresponde à previsão da variável de resposta [64].

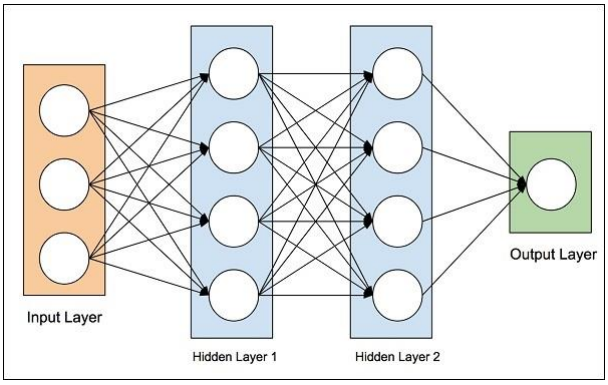


Figura 10 - Arquitetura de uma Rede Neuronal Artificial [65]

No geral, as RNA são robustas a ruido existente nos dados e podem representar funções lineares e não lineares de diversas formas. Como desvantagens desta técnica inclui-se a necessidade de ajuste de parâmetros e baixa interpretabilidade dos conceitos aprendidos pela RNA, uma vez que estão codificados sob a forma de pesos [58].

2.2.1.6 Redes Neuronais Recorrentes

As Redes Neuronais Recorrentes (RNC) utilizam técnicas de aprendizagem supervisionada e derivam das RNA. A diferença perante as RNA é que as RNC possuem conexões de *feedback*. Existem várias arquiteturas de RNC, sendo que estas vão desde redes totalmente interligadas, a redes parcialmente conectadas, incluindo redes *feedback* entre camadas [66].

Um exemplo de um algoritmo que utiliza RNC é o *Long/Short Term Memory* (LSTM), ilustrado na Figura 11. O presente algoritmo introduz um tipo de células diferente na camada oculta, as células de memória. Estas são compostas por elementos, designados de portas, que são recorrentes e controlam a forma como as informações são mantidas em memória, ou não. A porta de entrada decide quantas informações da última amostra são mantidas em memória, a porta de saída regula a quantidade de dados passados para a próxima camada e a porta de esquecimento controla a taxa de memória armazenada. Assim, à semelhança do cérebro humano, este algoritmo consegue manter em memória acontecimentos passados para contextualizar eventos futuros [67].

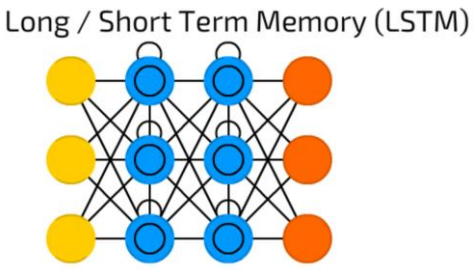


Figura 11 - Arquitetura exemplo de um LSTM [68]

2.2.1.7 Support Vector Machine (SVM)

SVM é uma técnica de aprendizagem supervisionada para problemas de classificação, ilustrado na Figura 12. Este algoritmo utiliza o princípio do cálculo de margens. Ou seja, dado um *dataset* T, o algoritmo procura um híper plano capaz de separar os dados de T com um erro de classificação mínimo, maximizando a margem de separação entre as classes [58]. As margens são definidas como a distância entre dois vetores de suporte separados pelo híper plano. O algoritmo assume que os dados são linearmente separáveis para que o peso associado aos vetores de suporte possa ser desenhado facilmente e a margem calculada [57].

Este algoritmo é algo complexo, no entanto a sua performance e *accuracy* são independentes do tamanho dos dados e dependentes da quantidade de ciclos de treino. Outra característica do SVM é que possui a habilidade de generalização e o treino é lento (devido à necessidade de grande quantidade de ciclos) [63].

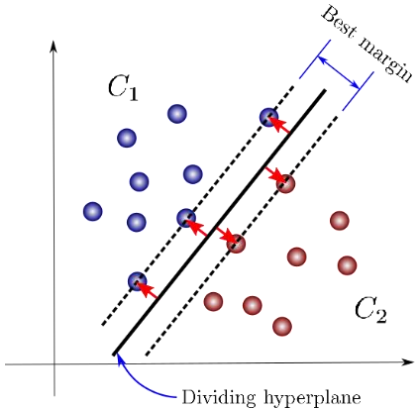


Figura 12 - Support Vector Machine (SVM) [48]

2.2.1.8 Regressão Linear

A regressão linear é uma técnica de aprendizagem supervisionada utilizada para problemas de regressão. Esta técnica utiliza uma função matemática linear para prever uma variável de destino, ajustando a melhor relação linear entre a variável independente (x) e dependente (y) [69]. A relação linear é descrita através de uma reta, denominada de “linha de melhor ajuste” [69].

Tomando como um exemplo de regressão linear a Figura 13, a linha vermelha presente na mesma é a reta de melhor ajuste para os valores dispersos. Desta forma, um algoritmo de regressão linear, com base nos dados fornecidos, tenta traçar uma linha que modele melhor os dados linearmente.

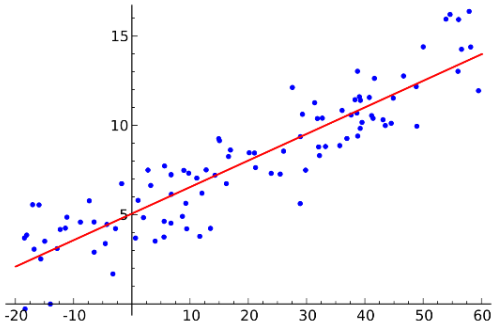


Figura 13 - Regressão Linear [70]

Na regressão linear existem dois tipos de categorias, a Regressão Linear Simples (RLS) e a Regressão Multi-Linear (RML). A RLS utiliza uma variável independente para prever uma variável dependente, ao passo que a RML utiliza múltiplas variáveis independentes para prever uma variável dependente [69].

A RLS é dada pela fórmula $Y = MX + C$, onde X é a variável independente, Y a variável dependente, M é a declive da linha e C é a interceção da linha.

A RML é dada pela fórmula $Y = \beta_0 + \beta_1 X_1 + \beta_1 X_2 + \beta_3 X_3 \ldots + \beta_n X_n$, onde X é a variável independente, Y a variável dependente, β_0 é a interceção da linha e β_n é a declive da linha.

2.2.2 Aprendizagem Não Supervisionada

A aprendizagem não supervisionada é um tipo de aprendizagem na qual uma função é inferida para representar a estrutura oculta de dados não rotulados [71]. Ou seja, ao contrário da aprendizagem supervisionada não existe uma referência histórica rotulada para determinado algoritmo aprender. O objetivo desse tipo de aprendizagem é descobrir semelhanças e padrões nos dados, tendo por base algumas informações à priori explícitas ou implícitas sobre o que é importante [72].

A técnica de aprendizagem não supervisionada mais utilizada é o c*lustering*. Esta envolve a descoberta de agrupamentos naturais de dados, denominados de clusters no espaço de recursos. Um *cluster* é geralmente uma área de densidade no espaço de recursos onde os dados estão mais próximos de um determinado cluster do que de outro(s). O cluster pode ter um centroide, que é ponto central dos dados e pode ter um limite [73].

2.2.2.1 K-Means

O algoritmo de *clustering* mais utilizado é o *K-Means*. Este é um algoritmo particionamento iterativo que tenta particionar o *dataset* inicial em *K* subconjuntos distintos, onde cada registo apenas pode pertencer a um subconjunto, Figura 14. Os pontos de dados são atribuídos a um *cluster* de forma que a distância entre os pontos e o centroide do cluster seja mínima. Isto faz com que, quanto menos variação existir dentro dos clusters, mais homogéneos serão os pontos de dados dentro do mesmo cluster [73].

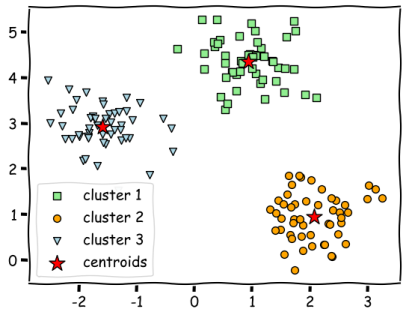


Figura 14 - Exemplo do Algoritmo *K-Means* [74]

A função objetivo J é dada pela fórmula abaixo, onde $||x_i^{(j)} - c_j||^2$ é a distância entre um ponto de dados $x_i^{(j)}$ e o centro do cluster c_j. [75]

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$$

2.2.2.2 Single-Linkage Clustering

O *Single-Linkage Clustering* é método de *clustering* hierárquico, ilustrado na Figura 15. Nestes métodos os *clusters* são formados pela divisão iterativa dos padrões usando uma abordagem de baixo para cima ou de cima para baixo, que correspondem ao *clustering* hierárquico *aglomerative* e *divisive*, respetivamente [75].

O tipo *aglomerative* constrói *clusters* começando com um único objeto. Depois, combina esses *clusters*, de forma a torná-los cada vez maiores. A combinação de *clusters* acontece, até que todos os objetos estejam finalmente dispostos num único *cluster* ou até que certas condições de sejam satisfeitas [75].

Por outro lado, o tipo *divisive* divide o *cluster* que contém todos os objetos em *clusters* menores. Isto acontece até que cada objeto forme um *cluster* por conta própria ou até que satisfaça certas condições de paragem [75].

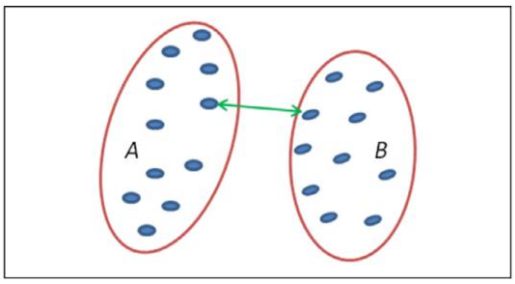


Figura 15 - Exemplo de *Single Linkage Clustering* [75]

O *Single-Linkage Clustering* é do tipo *aglomerative*. Neste, a ligação entre dois *clusters* é feita por um único par de elementos. A distância entre dois *clusters* é determinada pela menor

distância entre qualquer membro de um cluster e de outro, isso também define a semelhança dos dados [75]. Os critérios entre dois conjuntos de clusters A e B são os seguintes:

$$\min \{d\ (a, b):\ a\ \in\ A, b\ \in\ B\}$$

2.2.3 Aprendizagem por Reforço

A aprendizagem por reforço acontece quando um agente aprende a melhor maneira de realizar uma tarefa através de interações com um ambiente dinâmico, que responde com recompensas. O funcionamento básico deste tipo de aprendizagem, que se encontra ilustrado na Figura 16, pressupõem que existem apenas duas entidades, o agente e o ambiente [76].

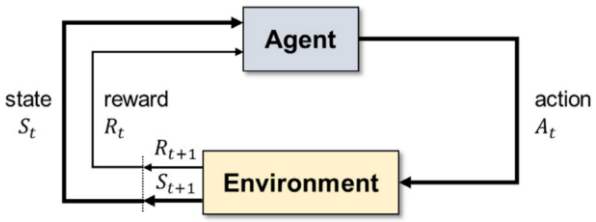


Figura 16 - Representação do Funcionamento da Aprendizagem por Reforço [76]

Num determinado momento t, o agente executa uma ação A_t que afeta o meio ambiente, fazendo com que ele altere de estado, de S_t para S_{t+1}. Como resultado disto, é gerada uma recompensa R_{t+1}. O agente usa essa informação (S_{t+1} e R_{t+1}) para gerar a próxima ação emA_{t+1}, continuando o ciclo. O objetivo do agente é aprender um mapeamento entre estados e ações, designada de política π $(A_t = a|S_t = s)$ [76]. Esta maximiza uma soma de longo prazo de recompensas futuras, chamada de função de valor, dada por:

$$v_\pi(s) = E\{R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+13} \ldots | S_t = s\}$$

Um outro fator importante é o fator de desconto γ, onde $\gamma \in [0,1]$, que determina a importância das recompensas futuras. Quando $\gamma = 0$ o agente age de forma a maximizar a recompensa da próxima etapa, ao passo que, quando $\gamma = 1$, o agente dará igual importância a todas as recompensas futuras. Desta forma, quanto maior for o desconto, maior será o horizonte temporal e por isso o agente terá maior variância. Pelo contrário, quanto menor for o desconto, menor será o horizonte temporal e por isso o agente será mais tendencioso [76].

2.3 Avaliação e Seleção de Algoritmos

Diversos algoritmos foram propostos ao longo do tempo. Isto faz com que atualmente exista uma grande diversidade dos mesmos e por isso determinar qual ou quais aplicar na resolução de um problema torna-se numa tarefa complexa. Desta forma, os modelos necessitam de ser avaliados segundo determinadas métricas [77]. No entanto, Wolpert e Macready no teorema "*No Free Lunch*" (NFL) sugerem que "Se o algoritmo A supera o algoritmo B em algumas funções

de custo, então, falando em termos gerais, deve existir exatamente o mesmo número de outras funções em que B supera A" [78]. Ou seja, quer isto dizer que as métricas de performance por si só significam pouco, é necessário compreender o problema e os dados para guiar a tarefa de seleção do modelo.

Posto isto, na presente secção serão apresentadas algumas métricas de performance enquadradas com base na previsão e com base no erro. Posteriormente é analisado o modo de seleção de algoritmos.

2.3.1 Métricas com base na previsão

A avaliação de modelos com base nos resultados preditivos do mesmo é realizada tipicamente em problemas de classificação. Isto porque estes problemas resultam em classes e é possível através das mesmas identificar os resultados positivos e negativos previstos. Os resultados são identificados através de [79]:

- Falso Positivos - número de casos positivos previstos de forma errada face à classe real;
- Falso Negativos - número de casos negativos previstos de forma errada face à classe real;
- Verdadeiro Positivos - número de casos positivos previstos corretamente face à classe real;
- Verdadeiro Negativos - número de casos negativos previstos de forma errada face à classe real.

A partir destes é possível construir uma matriz de confusão, ilustrada na Tabela 1.

Tabela 1 - Matriz Confusão

	Positivo Real	**Negativo Real**
Positivo Previso	Verdadeiro Positivo (TP)	Falso Positivo (FP)
Negativo Previsto	Falso Negativo (FN)	Verdadeiro Negativo (TN)

Despois de construída a matriz de confusão é possível calcular algumas métricas, que em conjunto permitem avaliar a performance do modelo:

- *Accuracy* (acc) – mede a percentagem de previsões corretas, no universo das instâncias avaliadas [79]. Esta é calculada pela seguinte fórmula:

$$acc = \frac{tp + tn}{tp + fp + tn + fn}$$

- *Error Rate* (err) – mede a percentagem de previsões incorretas no universo das instâncias avaliadas [79]. É dada pela fórmula:

$$err = \frac{fp + fn}{tp + fp + tn + fn}$$

- *Specificity* (sp) – mede fração de padrões negativos que são classificados corretamente [79]. É calcula pela fórmula:

$$sp = \frac{tn}{tn + fp}$$

- *Sensivity* (sn) = *Recall* (r) – mede a fração de padrões positivos que são classificados corretamente [79]. Esta é obtida pela fórmula:

$$sn = r = \frac{tp}{tp + fn}$$

- *Precision* (p) – mede os padrões positivos que são previstos corretamente a partir do total de padrões previstos para uma classe positiva [79]. É calculada através da fórmula:

$$p = \frac{tp}{tp + fp}$$

- *F1-Score* – avalia o desempenho do modelo de classificação a partir da matriz de confusão, efetuando uma média harmónica entre a *precision* e o *recall* [80]. É calculado através da fórmula:

$$F1\text{-}Score = \frac{2}{precision^{-1} + recall^{-1}} = 2 * (\frac{precision * recall}{precision + recall})$$

2.3.2 Métricas com base no erro

Avaliar um modelo com base nos erros do mesmo, consiste em medir a distância entre os dados das previsões e os dados treinados. Este tipo de métricas são utilizadas em problemas de regressão, uma vez que o resultado destes é expresso de forma numérica. Botchkarev [81] realizou um estudo intensivo sobre este tipo de métricas. O objetivo era perceber quais as métricas, baseadas no erro dos modelos, mais utilizadas nos projetos e artigos dos últimos 25 anos. Assim, nesta secção serão apresentadas as métricas mais utilizadas, de acordo com o estudo de Botchkarev [81].

- *Mean Square Error* (MSE) – medida que indica quão próxima uma linha de regressão está de um conjunto de pontos. As distâncias dos pontos até a linha de regressão são os erros [81]. É calculada pela fórmula:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2$$

- *Root Mean Square Error* (RMSE) – medida que afere quão espalhados estão os erros, ou seja, é o desvio padrão dos erros [81]. É calculada pela seguinte fórmula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y'_i)^2}$$

- *Mean Absolute Error* (MAE) – é a média de todos os erros absolutos, sendo que o erro absoluto é diferença entre o valor medido e o valor real [81]. É dada pela fórmula:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y'_i|$$

- *Mean Absolute Percentage* Error (MAPE) – medida expressa em percentagem, utilizada para avaliar a *accuracy* do modelo [81]. Esta é calculada pela fórmula:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - y'_i|}{|y_i|}$$

2.3.3 Seleção de algoritmos

Após calcular as métricas para avaliar os algoritmos e depois de os avaliar é necessário selecionar os mesmos. Esta seleção tem em conta, como indicado anteriormente pelo teorema NFL, os dados e o problema em questão. Shawkat Ali [77] afirma que "a abordagem de tentativa e erro é um procedimento muito comum para selecionar o melhor classificador" e como tal propõe uma metodologia para a seleção de algoritmos. Esta sugere que o *dataset* de treino seja analisado de forma que sejam extraídas:

- Medidas simples das suas características, como por exemplo: número de atributos, volume de dados, percentagem da classe minoritária, percentagem da classe maioritária, percentagem de variáveis binárias, percentagem de variáveis discretas, percentagem de variáveis contínuas e percentagem de valores em falta [77].
- Medidas estatísticas, como a média geométrica, média harmónica, desvio padrão, variância, enviesamento , entre outras [77].
- Medidas teóricas da informação, entropia média das variáveis, entropia das classes, rácio entre ruido e dados reais, entre outras [77].

Após esta análise é expectável que se conheça profundamente o *dataset* e a performance dos algoritmos. Assim, é possível conjugar estas informações, analisá-las em conjunto e tirar as conclusões finais. Por exemplo analisar o volume de dados com a performance, analisar o enviesamento e a variância para detetar casos de *overfitting*, como ilustrado na Figura 17, analisar o tempo de treino com a *accuracy* preditiva, efetuar uma validação cruzada para aferir a capacidade de generalização [63].

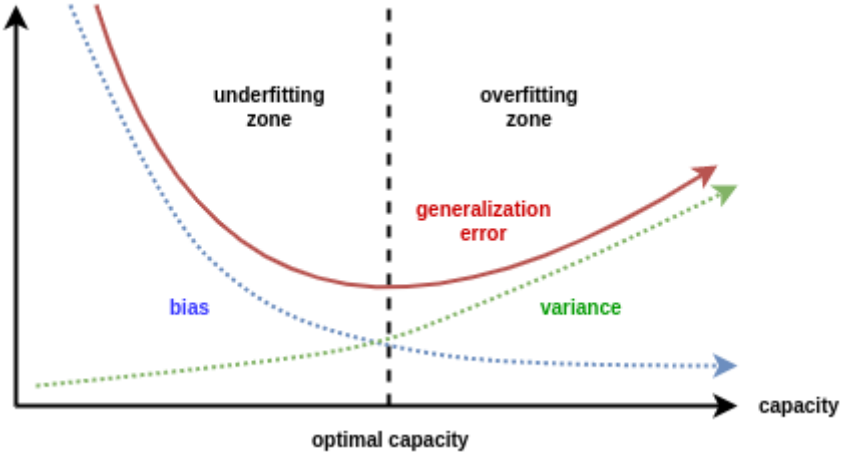


Figura 17 - Análise do Enviesamento e Variância [82]

2.4 Plataformas de Machine Learning

Na presente secção serão primeiramente apresentadas e analisadas diferentes plataformas de ML. Estas foram selecionadas tendo em conta uma análise de literatura [42], [39]. Posto isto, as plataformas são comparadas com base em 5 critérios, facilidade de utilização, custo, integração com terceiros, integração e *deployment* contínuo e adaptabilidade ao cliente.

2.4.1 Azure Machine Learning

O Azure Machine Learning (Azure ML) [90] é um serviço *cloud,* desenvolvido pela Microsoft, destinado ao desenvolvimento, treino e gestão de soluções de ML. Lançado em 2014, este serviço permite o desenvolvimento de todo o tipo de modelos ML, seja de aprendizagem supervisionada ou não supervisionada e é compatível com tecnologias *open-source*, como *Python, R, PyTorch* e *TensorFlow* [90].

Com o Azure ML é possível implementar grande parte das fases do CRISP-DM. Para tal é necessário desenvolver um *worflow*, utilizando o *Azure ML Designer* que é uma opção *low code* de desenvolvimento. Esta opção possibilita efetuar o pré processamento de dados, aplicar modelos de ML pré-carregados no serviço, treinar o modelo e avaliá-lo [92]. Para além do *Azure ML Designer* também é possível desenvolver todas as tarefas do *workflow* através de código. Neste sentido o Azure ML fornece *kits* e serviços de desenvolvimento de software (SDK) para preparar dados, desenvolver e treinar modelos de ML personalizados[90].

Para a fase de *deployment* e gestão do ciclo de vida do modelo de ML este serviço conta com o MLOps, que se trata de uma integração do Azure ML com o *Azure DevOps* e *GitHub*. Assim, todas as alterações ao modelo podem ser geridas e registadas através deste componente. O MLOps permite registar e gerir todas as alterações a qualquer momento do desenvolvimento do modelo. Isto é, quer uma alteração seja feita na fase de preparação de dados, quer seja feita

na fase de treino, todas as alterações ficam registadas. À parte do controlo de versões o MPOps permite também a agilização do processo de Integração Contínua e *Deploy* Contínuo (CI/CD), permitindo por exemplo integração com o *Azure IoT Edge* [93].

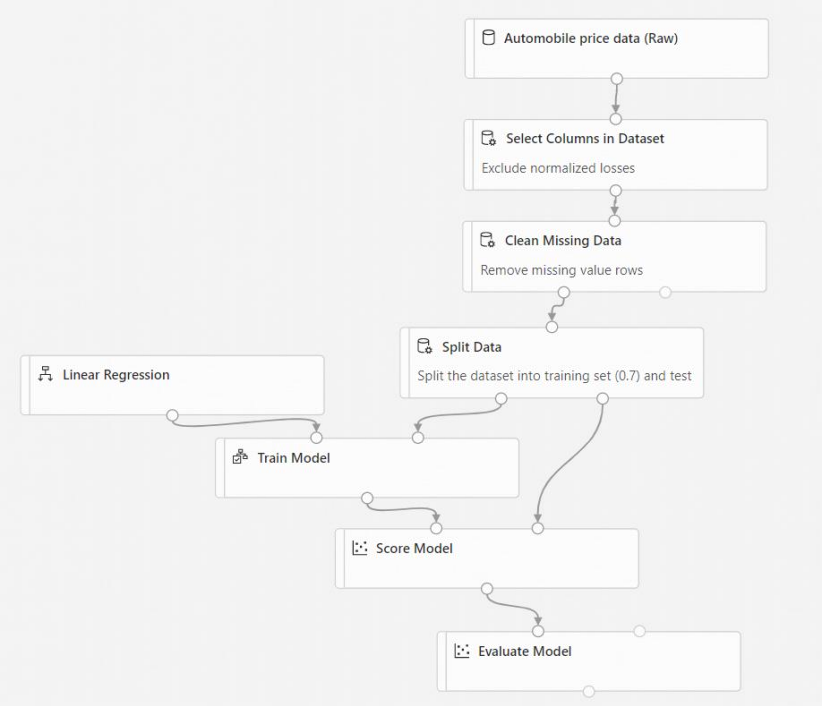


Figura 18 – Exemplo de *Workflow* Azure ML [92]

Para clientes que já possuam as suas operações, em parte ou totalmente, na *cloud* e em particular no Azure, este serviço acaba por ser uma extensão da mesma. Isto porque para ser possível aceder ao serviço Azure ML é necessário criar um *workspace* com vários componentes Azure [94]. Posto isto, como presente na Figura 19, este serviço em termos de arquitetura é um conjunto de pequenos componentes Azure.

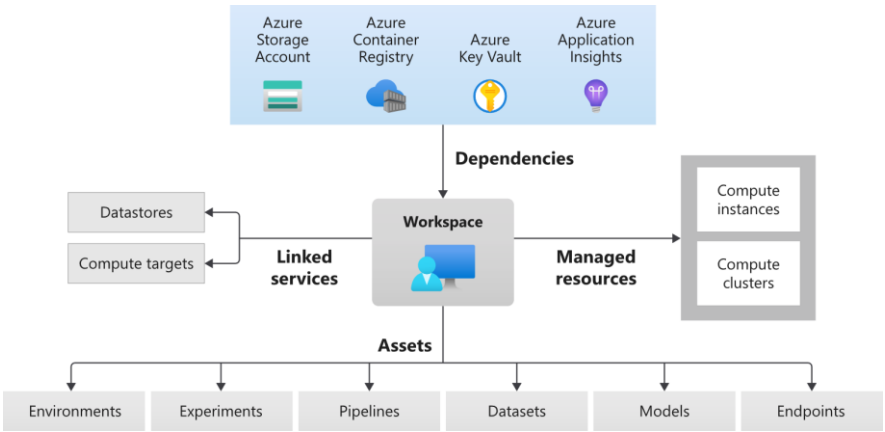


Figura 19 - Arquitetura do Azure ML [94]

2.4.2 *Google Cloud AI Platform*

Google Cloud AI Platform (GCAIP) é uma plataforma da Google que é composta por um conjunto de serviços *cloud* do Google Cloud. Esta plataforma é vocacionada para ciência de dados e ML e foi desenhada com o objetivo de cumprir o "ciclo de vida completo do *Machine Learning*" [95]. A arquitetura desta plataforma, Figura 20, contempla serviços para atendar à preparação de dados desenvolvimento, treino, validação, implantação e gestão de modelos de ML na *cloud* [96].

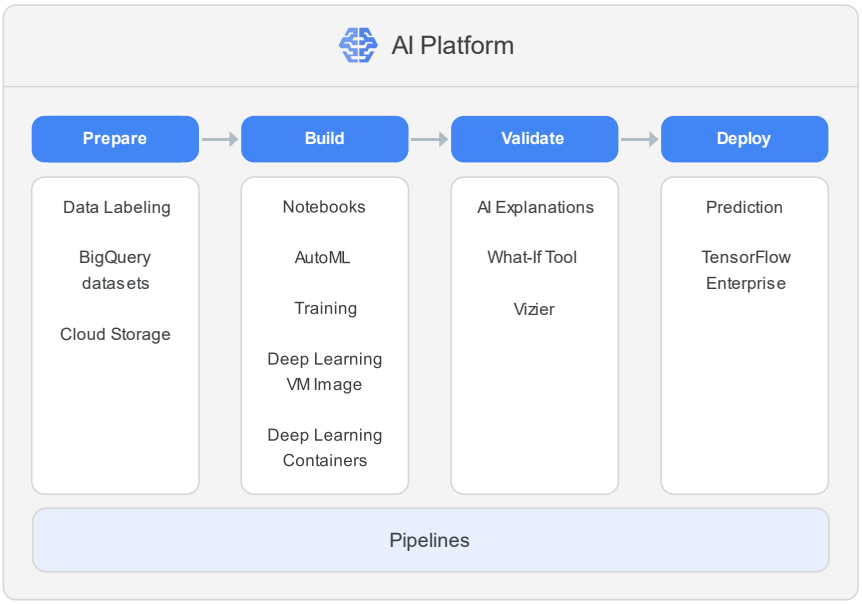


Figura 20 - Arquitetura Google Cloud AI Platform [95]

Na fase de preparação a GCAIP possui o serviço *BigQuery* e *Cloud Storage* para a limpeza, ingestão e armazenamento dos dados. Ainda nesta fase a plataforma da Google Cloud possui um serviço de rotulagem de dados de treino para a classificação, deteção de objetos, extração de entidade e outras finalidades para dados em imagens, vídeos, tabelas e textos [95].

A presente plataforma, na fase de desenvolvimento do modelo possui, através do AutoML, um serviço *codeless* para desenvolvimento de modelos sem escrita de código e um serviço de criação de modelos com código, através de *Notebooks Jupyter*. Esta plataforma possui *frameworks* de aprendizagem como o *Deep Learning VM Image* ou o *Deep Learning Containers*. Nesta fase existe também o *AI Platform Training*, um serviço de treino onde é possível treinar o modelo utilizando o *TensorFlow, Keras TensorFlow, scikit-learn, XGBoost*, ou o *PyTorch* [95].

Nesta plataforma, na fase de validação do modelo, existem os serviços *AI Explanations* e *Vizier* O *AI Explanations* é composto por um conjunto de ferramentas que auxiliam a entender os resultados do modelo, verificar o seu comportamento, reconhecer as tendências do mesmo e obter ideias para melhorar o modelo e os dados de treino. O *Vizier* é um serviço que auxilia no ajuste de híper parâmetros e otimização do resultado do modelo [96].

Para realizar o *deploy* dos modelos esta plataforma possui os serviços *Prediction, AutoML Vision Edge* e o *TensorFlow Enterprise.* O serviço *Prediction* gere a infraestrutura necessária para executar modelo, disponibilizando-o para pedidos on-line e em lote. O AutoML Vision Edge permite o *deploy* dos modelos para dispositivos *edge*, por exemplo smartphones, e gerar ações em tempo real com base em dados locais. O *TensorFlow Enterprise* oferece suporte de nível empresarial para instâncias do *TensorFlow* [95].

Por fim para a gestão e controlo de versões dos modelos existe o serviço *AI Platform Pipelines* que implementa práticas de MLOps. Este serviço utiliza o *Kubeflow* Pipelines ou o *TensorFlow Extended* para possibilitar a criação de pipelines de ML que fornecem *feedback* contínuo [96].

2.4.3 Amazon SageMaker

Lançado pela empresa Amazon Web Services (AWS) em 2017, o Amazon SageMaker é uma plataforma de ML totalmente gerida na *cloud*. Esta possui diferentes módulos, Figura 21, que podem ser usados juntos ou de forma individual para preparar os dados, criar, treinar, validar e efetuar *deploy* de modelos de ML [97]. O Amazon SageMaker, apesar de não ser *open-source*, é compatível com tecnologias, como o *Pythoh, R, PyTorch, Apache Spark* e *TensorFlow*.

Figura 21 - Módulos e Funcionalidades do *Amazon SageMaker* [98]

A fase de preparação de dados, nesta plataforma, contem os módulos [98]:

- *SageMaker Ground Truth* – serviço totalmente gerenciado de rotulagem de dados, através de *workflow*;
- *SageMaker Data Wrangler* – serviço de preparação e agregação de dados, incluindo seleção, limpeza, exploração dos mesmos;
- *SageMaker Processing* – serviço de processamento de dados, como engenharia de recursos e validação de dados, através de *Python, BYO R/Spark*;

- *SageMaker Clarify* – serviço que permite auxiliar na identificação e limitação de tendências e explicar previsões.

Na fase de construção de modelos de ML, importa salientar que é possível utilizar diversos algoritmos pré feitos ou utilizar uma das *frameworks* compatíveis, como *Scikit-Learn, TensorFlow, PyTorch, MXNet* ou *Chainer*. Assim, nesta fase, através desta plataforma é possível encontrar os módulos [98]:

- *SageMaker Studio Notebooks* – *notebooks* colaborativos que fornecem armazenamento persistente. Isto é especialmente útil para partilhar *notebooks* e reproduzir os mesmos resultados em diferentes instâncias;
- *SageMaker Autopilot* – serviço que permite a criação, treino e ajuste automático de modelos de ML para classificação ou regressão com base nos dados fornecidos;
- *SageMaker JumpStart* – serviço que oferece um conjunto de soluções para os casos de uso mais comuns, para que seja possível iniciar o desenvolvimento a partir de um exemplo pré feito.

A presente plataforma na fase de treino apresenta os seguintes módulos [98]:

- *SageMaker Experiments* – serviço que permite comparar e avaliar os diversos ensaios com os modelos de ML;
- *SageMaker Debugger* – serviço de auxílio à otimização de modelos ML, uma vez que permite capturar métricas de treino em tempo real e enviar alertas aquando da deteção de anomalias;
- *Automatic Model Tuning* – serviço de otimização de hiperparâmetros.

Por fim, na fase de *deploy*, a presente plataforma possui os seguintes módulos [98]:

- *SageMaker Model Monitor* – serviço de monitorização da qualidade do modelo, com base em variáveis independentes e dependentes, que envia alertas em tempo real sobre desvios de mesmo;
- *SageMaker Edge Manager* – agente de software que é executado em dispositivos *edge* e permite efetuar *deploy* de modelos ML para os mesmos. Este agente coleta dados de predição e envia uma amostra dos mesmos para a *cloud*, para monitorização, rotulagem e novos treinos.
- *SageMaker Pipelines* – serviço de integração contínua e entrega contínua (CI/CD)

2.4.4 Comparação de Plataformas

Após terem sido abordadas e detalhadas algumas ferramentas de ML, torna-se então necessário comparar as mesmas. Assim, estas serão comparadas sobre alguns critérios importantes para o desenvolvimento da solução e outros critérios importantes para o cliente. Uma vez que todas as ferramentas descritas possuem capacidades semelhantes no que diz

respeito à preparação de dados, criação, treino e validação de modelos, os critérios de comparação selecionados são a facilidade de utilização, o custo, a integração com terceiros, integração e *deployment* contínuo e adaptabilidade.

O critério de facilidade de utilização é bastante importante para o desenvolvimento da solução, uma vez que, quanto menor for a sua curva de aprendizagem, mais rápido será o processo de desenvolvimento. A facilidade de utilização entende-se por uma interface limpa e intuitiva, onde seja facilmente percetível onde estão os principais componentes e para que servem. Neste critério o Azure ML é claramente o que possui maior facilidade de utilização, uma vez que, todo o processo de desenvolvimento pode ser realizado sem código e com *drag and drop*, através do Azure ML Designer. Através deste é possível desenvolver um *workflow* com componentes pré-definidos, o que agiliza todo o processo, bem como ajuda à compreensão do mesmo.

O custo é bastante importante, uma vez que, após o término da fase de projeto este será um critério que poderá ditar o insucesso e a não aplicação do serviço à escala industrial. Ou seja, se o custo for demasiado elevado, o cliente poderá não ter bases suficientes para o suportar. Como tal este fator, como em qualquer negócio, é deveras importante. No entanto, estimar custos em plataformas *cloud as a service* torna-se numa tarefa complexa, isto porque, o custo é calculado pela taxa de uso. Como tal, para efeitos de comparação, foi fixado um escalão computação semelhante em todas as plataformas. Este é o escalão básico com 2 vCPU e 4 GiB de memória, [99], [100], [101].

Através da análise da Tabela 2, onde se encontram os custos das plataformas, é facilmente percetível que o Azure ML possui um custo menor. Para além do menor custo direto, este ainda pode ser diminuído em cerca de 62%, para 0,016€/hora, se o serviço for reservado por três anos [99].

Tabela 2 - Comparação de Custos de Plataformas

Plataforma	Custo	Desconto de reserva
Azure ML	0,041€/hora	Reserva de 1 ano – 42% Reserva de 3 ano – 62%
Google Cloud AI Platform	0.183€/hora	Não
AWS SageMaker	0.0548€/hora	Não

Sobre a capacidade de integração com *frameworks* e serviços de terceiros, este critério é importante, mas não essencial, para a comparação. Isto porque não se torna imperativo a utilização de *framework* externas, se a plataforma já possuir capacidade autónoma suficiente. Através da análise da Tabela 3, é possível perceber que o AWS SageMaker é mais competente no que diz respeito à integração com terceiros.

Tabela 3 - Comparação entre integração de *frameworks* de terceiros

Plataforma	*Framework* e/ou serviços de terceiros
Azure ML	*TensorFlow, scikit-learn, Microsoft Cognitive Toolkit, SparkML, Jupyter Notebooks, R, Python*
Google Cloud AI Platform	*TensorFlow, scikit-learn, XGBoost, Keras, Jupyter Notebooks, R, Python*
AWS SageMaker	*TensorFlow, MXNet, Keras, Gluon, Pytorch, Caffe2, Chainer, Torch, Jupyter Notebooks, R, Python*

A integração contínua e *deployment* contínuo (CI/CD) é um aspeto importante a considerar na comparação e avaliação das plataformas. A capacidade de CI/CD, que em ML é designada de MLOps, permite automatizar e agilizar as tarefas pós desenvolvimento do modelo. Isto é, permite versionar os dados de treino e o modelo desenvolvido, treinar e validar o modelo a cada execução da pipeline, efetuar o *deploy* para um outro ambiente e/ou dispositivo e monitorizar o comportamento do modelo [102]. Com o conjunto de todas estas tarefas que o MLOps permite efetuar, é possível o modelo aprender como novos dados e comportamentos dos sistemas, como ilustrado na Figura 22.

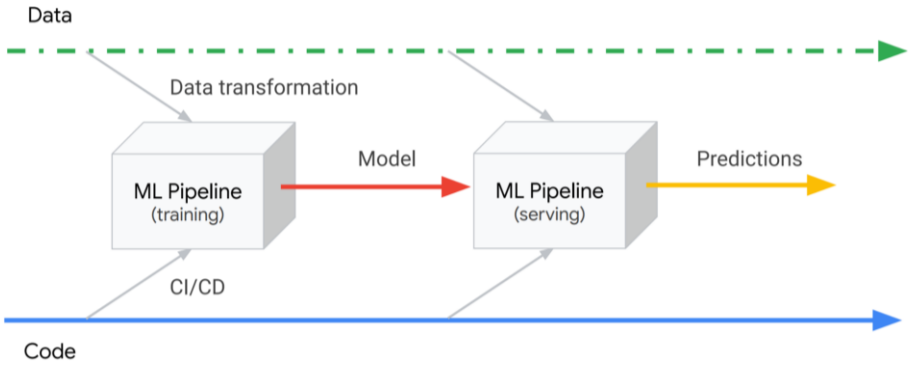


Figura 22 - Funcionamento do MLOps [102]

Através de uma análise à Tabela 4, é possível perceber que o Azure ML e o Google Cloud AI Platform possuem um maior número de integrações com repositórios, por outro lado o Google Cloud AI Platform possui um maior número de integrações com *pipelines* CI/CD [103], [104], [105], [106], [107].

Tabela 4 - Comparação entre CI/CD

Plataforma	*Pipelines*	Repositórios
Azure ML	Azure Pipelines, Bitbucket Pipelines	Repositórios Git, Azure DevOps
Google Cloud AI Platform	TFX, Cloud Build, Pipelines do Kubeflow	Repositórios Git, Google Source Repositories
AWS SageMaker	Amazon SageMaker Pipelines	Repositórios Git

O último critério de comparação e avaliação que foi tido em conta foi a adaptabilidade à atual infraestrutura do cliente. Este possui grande parte das suas operações *cloud* baseadas em Azure, tendo inclusive contrato de suporte *Premier* com suporte 24h por 7 dias da semana. O cliente utiliza também AWS, no entanto este serviço *cloud* encontra-se numa fase embrionária possuindo apenas pequenos testes e provas de conceito. O único serviço *cloud* com o qual o cliente não possui qualquer contacto é o Goolgle Cloud. Como tal, do ponto de vista de adaptabilidade, qualquer solução que utilize Azure terá maior grau de confiança e aceitação por parte do cliente.

Por fim, para comparar globalmente as três plataformas foi elaborada a Tabela 5. Nesta é possível aferir que no geral a plataforma Azure ML é a melhor a adotar no desenvolvimento da presente dissertação. Na secção 3.6 esta comparação é efetuada tendo em conta o método TOPSIS, no qual se chegou à mesma conclusão.

Tabela 5 - Comparação de Platafomas Tecnológicas ML

Plataforma	**Facilidade de Utilização**	**Custo**	**Integração com terceiros**	**Integração e *Deployment* Contínuo**	**Adaptabilidade**
AWS SageMaker	Bom	Bom	Ótimo	Razoável	Bom
Azure ML	Ótimo	Ótimo	Bom	Bom	Ótimo
Google Cloud AI Platform	Bom	Mau	Bom	Ótimo	Mau

2.5 Trabalhos Existentes

Nesta secção são abordados diversos trabalhos realizados no âmbito da previsão de falhas e manutenção preditiva. Para identificação destes trabalhos foram analisados diversos artigos publicados com as *keywords* "*Machine Learning*", "Previsão de Falhas", "Manutenção Preditiva", "Indústria 4.0" e "Serviços Preditivos para Equipamentos Industriais", nomeadamente [39], [40], [41], [42], [43], [44], [45], [46], [47], [108], [109], [110], [111], [112]. Após esta análise de literatura foram selecionados os artigos cujo trabalho incidiu sobre ambiente industrial, utilização de máquinas industriais, ambiente de Indústria 4.0., ou o problema era semelhante. Desta forma foram selecionados os seguintes trabalhos [40], [41], [42], [43], [44], [45], [46].

Com base na literatura analisada foi tomada a decisão de dividir a presente secção pelos algoritmos mais utilizados nas mesmas, ou seja, *Random Forest, Support Vector Machine* e Redes Neuronais.

2.5.1 Abordagens de Manutenção Preditiva utilizando Random Forest

Em 2017 um conjunto de investigadores, do Instituto de Tecnologia da Universidade de Deusto, redigiram um artigo no qual propõem uma solução *cloud* de manutenção preditiva em tempo

real [40]. Esta tem como objetivo a previsão de falhas em turbinas eólicas com uma grande velocidade de processamento e alta escalabilidade. A solução apresentada é composta por três módulos principais: 1) O módulo gerador de modelos preditivos para cada turbina eólica; 2) O módulo de monitorização, que em cada 10 minutos gera previsões de falhas nas turbinas eólicas para a próxima hora; 3) O módulo de visualização das previsões. Para a implementação da solução a equipa de trabalho utilizou tecnologias como *Apache Spark, Apache Kafka, Apache Mesos* e *HDFS*. Neste artigo, após diversos testes, o algoritmo adotado para o módulo de geração de modelo foi o *Random Forest*. Com este trabalho conseguiu-se uma melhoria de 5,54%, no que diz respeito à *accuracy* preditiva, quando comparado com trabalhos anteriores [40].

No ano de 2018 um grupo de uma universidade de Taiwan, Yuan Ze University, propõe um sistema preditivo em tempo real, denominado de "HDPass" [41]. Este sistema tem como objetivo prever e detetar falhas em discos rígidos nos servidores de uma determinada *cloud.* Este consiste em duas fases: 1) A fase de treino em lote, onde são gerados e treinados modelos com base no algoritmo *Random Forest* e onde são utilizados dados históricos; 2) A fase de previsão em tempo real, onde são utilizados dados em tempo real e são aplicados os modelos previamente treinados. Esta abordagem consegue atingir uma *accuracy* de cerca de 85% nas previsões efetuadas [41].

Ainda em 2018, um grupo de estudantes do departamento de engenharia informática de uma universidade italiana, *Università Politecnica delle Marche*, em conjunto com uma estudante de uma universidade sueca, *Uppsala University*, utilizaram uma abordagem de manutenção preditiva numa máquina de corte [42], idêntica à da Figura 23. Esta abordagem foi realizada em ambiente experimental, tendo por base um exemplo de um grupo industrial real. O objetivo do estudo desenvolvido por este grupo seria o desenvolvimento de uma metodologia de ML na *cloud* para manutenção preditiva.

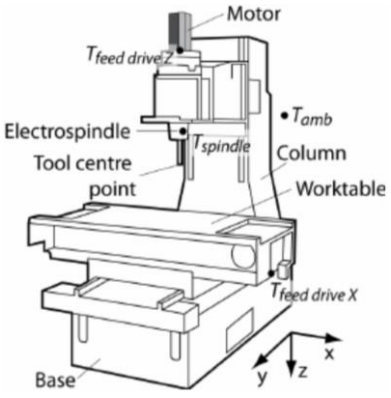


Figura 23 - Exemplo de máquina utilizada nesta abordagem [42]

Neste artigo os dados necessários para a implementação do projeto foram recolhidos de diversos sensores de máquinas, PLC's e protocolos de comunicação. O *dataset* recolhido continha 530731 linhas e foram extraídas 15 variáveis do mesmo. Numa primeira análise ao

problema, o grupo de trabalho encontrou diversos pontos de medição para avaliação da manutenção, nomeadamente:

- "As vibrações da máquina podem sinalizar deterioração do rolamento ou deformação de peças mecânicas" [42].
- A temperatura de um motor e da corrente do mesmo podem indicar um atrito anormal e um possível mau funcionamento mecânico que estão a degradar o funcionamento [42].
- A medição das partículas de um lubrificante poderá indicar a degradação das peças que fazem fricção. Com os devidos sensores é possível medir a composição do óleo lubrificante e verificar o estado da máquina [42].

Após esta identificação o grupo efetuou uma estimativa de parâmetros seguida de uma análise gráfica dos dados, para uma melhor compreensão dos mesmos. Para esta estimativa de parâmetros existem duas possibilidades, a previsão transversal ou a previsão de série temporal. A diferença entre ambas é que a análise transversal analisa os dados recolhidos num único espaço temporal, ao invés da análise de séries temporais que se refere à estimativa de parâmetros que mudam ao longo do tempo. Na análise de séries temporais os valores medidos são até o instante *t* e o valor a ser previsto está no instante *t + dt*. Segundo os autores as séries temporais permitem identificar:

- Tendências, ou seja, ou um aumento ou diminuição ao longo do tempo nos valores.
- "Os fenômenos sazonais, ou seja, os fenómenos que determinam mudanças nos valores ao longo de um período que sempre se repete na mesma duração" [42].
- "Fenómenos cíclicos que provocam aumentos e diminuições de valores com flutuações que nem sempre têm a mesma duração, ou seja, não são periódicas" [42].

Para uma solução de manutenção preditiva, segundo os autores, após a análise de dados deverão ser identificados nos dados os seguintes aspetos:

- Histórico de falhas – essencial para que os algoritmos preditivos aprendam com base nos erros e falhas anteriores. Para além dos erros é também essencial a presença de histórico de operação normal do sistema.
- Histórico de manutenções – necessário para correlacionar os erros que ocorreram no passado com as manutenções efetuadas para corrigir os mesmos.
- Condições da(s) máquina(s) – para que se possa estimar quanto tempo falta até à próxima falha, é fundamental existirem dados das condições da máquina ao longo do tempo. Correlacionando estes dados, com o histórico de falhas e manutenções é possível prever padrões de envelhecimento ou degradação.

Após uma investigação e análise de algoritmos de ML, nomeadamente Classificação Binária, Classificação Multiclasse e Modelos de Regressão, o grupo de trabalho acabou por utilizar o algoritmo *Random Forest*. A escolha sobre este algoritmo foi justificada pelo facto de

considerarem que "modelos *ensemble* geralmente fornecem melhor cobertura e *accuracy* do que árvores de decisão únicas" [42].

No que diz respeito à arquitetura, no presente artigo, foi utilizada uma arquitetura *cloud* híbrida. Nesta, existe uma máquina *on-premises* que hospeda a fase de análise de dados, ao passo que o pré processamento de dados, implementação, treino e avaliação do modelo foi realizado no serviço *cloud Azure Machine Learning* [42]. Importa ainda salientar que, 30% dos dados foram utilizados para treino e os restantes 70% utilizados para avaliação do modelo [42]. Por fim, os resultados obtidos nesta abordagem, encontram-se na Tabela 6.

Tabela 6 - Resultado obtidos na avaliação do modelo [42]

Métrica	Resultado
Overall Accuracy	0,95
Average Accuracy	0,92
Micro-Averaged Precision	0,94
Macro-Averaged Precision	0,93
Micro-Averaged Recall	0,95
Macro-Averaged Recall	0,94

2.5.2 Abordagens de Manutenção Preditiva utilizando Support Vector Machine

No ano de 2014, investigadores de uma universidade indiana, *Amrita Vishwa Vidyapeetham*, propõem um modelo baseado em *Support Vector Machine* (SVM) para deteção de falhas em caixas de transmissão [43]. Segundo estes, a caixa de transmissão "é um dispositivo essencial utilizado nas indústrias para variar a velocidade e as condições de carga de acordo com certos requisitos" [43] e "a falha em qualquer um dos componentes da caixa pode levar à perda de produção e aumentar o custo de manutenção" [43]. Para resolver o problema identificado foram utilizadas quatro caixas de transmissão. Estas foram testadas em duas velocidades e condições de carga diferentes. Em seguida, os dados dos sinais de vibração foram usados para treinar o modelo SVM. Neste estudo após diversas experimentações, treinos e comparações, o modelo desenvolvido permitiu classificar falhas nas caixas de transmissão com uma eficiência média de 97% [43].

Em 2017 um grupo de investigadores de várias universidades de Singapura propuseram um *kernel* de regressão modificado que utiliza o algoritmo SVR [44]. Este trabalho foi realizado no âmbito dos problemas de previsão, nomeadamente a previsão da *remaining useful life* (RUL). Segundo os mesmos, os métodos SVR tradicionais assumem um mapeamento entre as características de entrada e saída do modelo. No entanto, os problemas de previsão utilizam medições ao longo do tempo oriundas de vários sensores e os *datasets* consistem em vários ciclos relacionados. O modelo desenvolvido foi testado com dados de séries temporais simulados e simplificados. Os testes realizados por esta equipa de investigadores mostram que o modelo SVR proposto supera um modelo SVR padrão [44].

2.5.3 Abordagens de Manutenção Preditiva utilizando Redes Neuronais

Em 2018, investigadores do Instituto de Tecnologias de Informação do CERTH (*The Centre for Research & Technology*) apresentam uma metodologia para manutenção preditiva de equipamentos industriais no âmbito da produção de ânodos [45]. Nesta metodologia são utilizados dados provenientes de sensores existentes nas máquinas de produção, como sensores de temperatura e pressão do motor, do combustível e líquido de arrefecimento. O grupo de investigadores comparou algoritmos de Redes Neuronais com outras técnicas de ML, tendo chegado à conclusão de que o melhor algoritmo a aplicar neste caso seria o LSTM, um tipo de Redes Neuronais Recorrentes. Com esta metodologia conseguiram prever falhas com uma antecedência de 5 a 10 minutos [45].

Ainda em 2018, um grupo de investigadores alemães e holandeses propõe um sistema de monitorização da integridade de máquinas [46]. Este sistema tem como objetivo a previsão de indicadores de integridade de máquinas, com duas semanas de antecedência. Nesta abordagem é utilizada uma técnica de *deep learning* que combina camadas recorrentes bidirecionais com células GRU (*Gated Recurrent Units*) e camadas totalmente conectadas. Para além disso, são também utilizados métodos de *clustering K-Means* para incorporar o conhecimento prévio da distribuição da variável prevista no modelo. Segundo os investigadores os modelos de Redes Neuronais Recorrentes tipicamente “apenas usam dados do sensor e não incorporam informações categóricas” [46]. Este estudo tem como objetivo apresentar um método através do qual os “dados numéricos sequenciais podem ser combinados com entradas categóricas para obter previsões mais precisas do funcionamento da máquina” [46]. Adicionalmente, o modelo desenvolvido neste trabalho “é capaz de fornecer previsões superiores de integridade de uma máquina do que um modelo RNN tradicional” [46].

3 Análise de Valor

Neste capítulo é efetuada a análise de valor do presente projeto, sendo que esta irá elucidar e esclarecer qual o valor do projeto para o cliente. Nesta é identificado o cliente do presente projeto, seguido do enquadramento do processo de inovação, no qual é identificada e analisada a oportunidade. Esta oportunidade é analisada, recorrendo a uma análise de tendências tecnológicas e tendências de investimento em Indústria 4.0 e IA. Posto isto é enquadrado o valor para o cliente e analisado o valor percebido por este, sendo posteriormente efetuada uma proposta de valor. Por fim, é realizada uma análise *Function Analysis System Technique* (FAST) e uma análise *Technique for Order of Preference by Similarity to Ideal Solution* (TOPSIS).

3.1 Identificação do Cliente

O cliente do presente projeto é uma unidade de negócio da empresa Corticeira Amorim, do Grupo Amorim. O Grupo Amorim é uma das maiores multinacionais de origem portuguesa. Teve origem no negócio da cortiça, em 1870, sendo hoje líder destacado no setor a nível mundial. Por sua vez a Corticeira Amorim é a maior empresa mundial de produtos de cortiça liderando todo o setor. A unidade de negócio Amorim Cork (AC), , destina-se à produção de rolhas de cortiça para vinho, vinhos efervescentes e bebidas espirituosas. A AC está distribuída pelos principais países produtores de vinho, desde o continente europeu aos mercados da África do Sul, Austrália e América do Sul, possuindo cerca de 18 mil clientes ativos. Esta unidade de negócio tem apostado fortemente em Investigação e Desenvolvimento, o que resultou em projetos como o sistema Helix ®, que se refere a uma solução única de embalagens, ou como o NDtech, uma tecnologia de análise individualizada que permite detetar os níveis de TCA de cada rolha em segundos.

Importa ainda salientar que a equipa de desenvolvimento do presente projeto é uma equipa da empresa OSI. Esta é uma empresa interna do Grupo Amorim destinada ao apoio e desenvolvimento tecnológico do mesmo

3.2 Processo de Inovação

A rápida e crescente evolução tecnológica faz emergir constantemente novos produtos e serviços para o mercado [113]. Estes, por sua vez, colmatam as necessidades dos seus clientes e geram outras novas necessidades. Desta forma, a inovação tende a ser constante e torna-se necessário a existência de um processo, ilustrado na Figura 24. Este processo de inovação é composto por três fases, *Fuzzy Front End, New Product Development e Commercialization*.

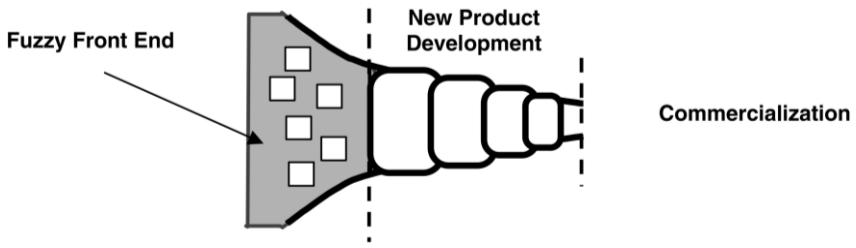


Figura 24 - Processo de Inovação [114]

A fase inicial do processo de inovação é denominada de *Fuzzy Front End*. Nesta, é identificado o problema, são geradas ideias e oportunidades e as atividades desenvolvidas são tipicamente "caóticas, imprevisíveis e não estruturadas" [115].

O *New Product Development* é a fase que se segue no processo de inovação. Esta é caracterizada por atividades mais previsíveis, disciplinadas, com maior grau de certeza e nas quais existe um plano de projeto. [114]

A fase final do processo é fase de *Commercialization*, na qual o produto desenvolvido na anterior fase já está maduro e pronto a ser lançado para o mercado. É nesta fase que as necessidades e desejos dos clientes são satisfeitas.

No presente projeto existe uma forte componente exploratória. Apesar do produto final do mesmo ser um serviço preditivo, cujas atividades tipicamente são disciplinadas, há uma alguma incerteza quanto ao módulo preditivo do mesmo. Isto é, o desenvolvimento deste módulo requer um processo de investigação de algoritmos e abordagens, exploração dos dados das máquinas para perceber o valor dos mesmos e experimentação de modelos. Este tipo de atividades retratam a fase do *Fuzzy Front End* e como tal o projeto insere-se nesta.

3.2.1 New Concept Development

Com o intuito de conseguir uma linguagem comum e uma definição dos vários processos constituintes do *Fuzzy Front End*, Koen [115] propôs o modelo *New Concept Development*. Este encontra-se ilustrado na Figura 25 e é constituído por três componentes, o motor, as cinco atividades chave e os fatores de influência.

O componente do motor é o agente impulsionador das cinco atividades chave e é representado pela estratégia, cultura e liderança praticada pela empresa. As cinco atividades representadas na área interna do modelo, são as atividades chave controláveis pela empresa, sendo elas a Identificação da Oportunidade, Análise da Oportunidade, Geração e Enriquecimento de Ideias, Seleção de Ideias e Definição do Conceito. A área exterior do modelo refere-se aos fatores de influência que não são controláveis pela empresa e afetam o processo de inovação até à comercialização [115].

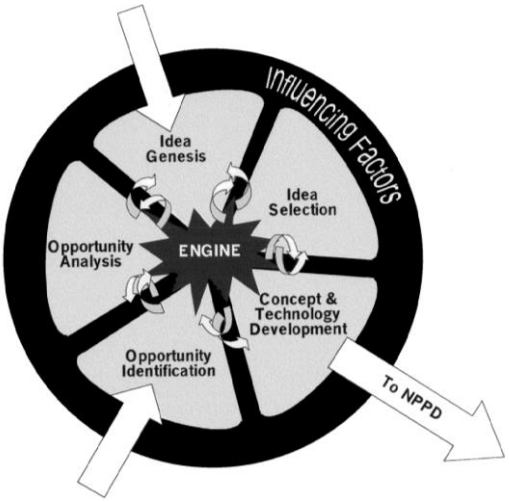


Figura 25 - Modelo New Concept Development (NCD) [115]

3.2.1.1 Identificação e Análise de Oportunidade

A identificação e análise da oportunidade é realizada de duas formas. Primeiramente é analisada a oportunidade no âmbito da Indústria 4.0, identificando as suas vantagens, investimentos e retornos dos mesmos. De seguida esta é analisada sobre o ponto de vista da IA, analisando as tendências tecnológicas. Isto porque a presente dissertação se insere no âmbito da Indústria 4.0, mais concretamente numa tecnologia constituinte da mesma, a IA. Sendo assim, é de todo pertinente identificar a analisar a oportunidade nestes dois níveis.

Como abordado na secção 1.1.1, a Indústria 4.0 é designada como a quarta revolução industrial. Esta é composta por um conjunto de tecnologias chave como sistemas ciber físicos, *Internet of Things* (IoT), *big data*, computação *cloud*, computação descentralizada, redes *wireless*, Inteligência Artificial, ciber segurança, entre outros. Os principais benefícios que esta revolução apresenta para as indústrias são o aumento da eficiência da produção, manutenção e administração, a redução dos custos, o aumento da qualidade de produto e o aumento do bem-estar dos funcionários, como ilustrado na Figura 26 [116].

Segundo um estudo da pwc [116], que abordou cerca de 26 empresas belgas de diversos setores, cerca de 73% das empresas abordadas afirmam que o principal fator motivador para realizar uma transformação digital é a pressão sobre seus custos. Por sua vez, 87% das empresas afirmam que já obtiveram retorno dos investimentos efetuados na transformação digital, principalmente em termos de eficiência produtiva e redução de custos. Sendo que, “nos últimos dois anos, os líderes digitais investiram em média 8% da sua faturação total por ano na transformação digital” [116]. Este estudo vai mais longe afirmando que, os líderes digitais “planeiam continuar a investir cerca de 10% da faturação ao ano nos próximos cinco anos” [116].

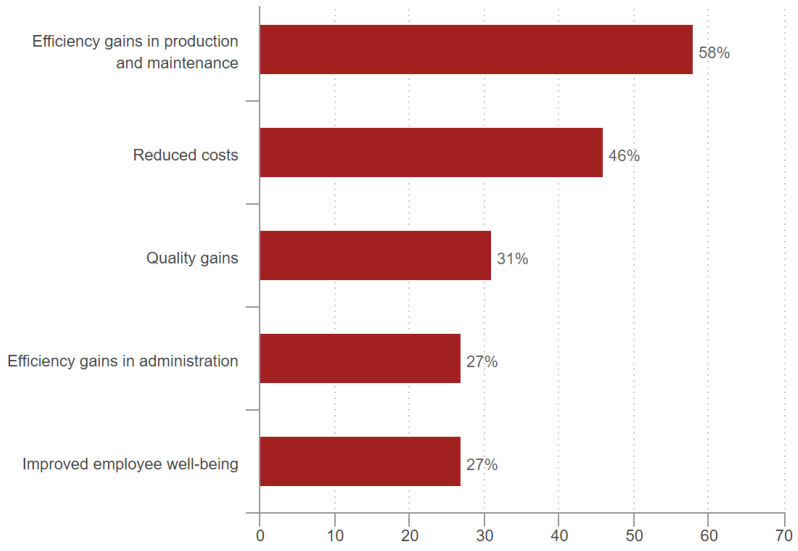


Figura 26 - Os 5 Principais Benefícios da Indústria 4.0 [116]

Segundo a Forbes [117], existem cinco fases para uma determinada empresa de manufatura atingir o estatuto de "fábrica inteligente". São elas:

1. A fase da monitorização remota na qual a empresa consegue monitorizar certos aspetos das fábricas remotamente, não existindo, no entanto, qualquer tipo de sensores ou sistema de recolha automática de dados [117].
2. A fase de monitorização em tempo real, na qual a fábrica já possui sensores de *IoT* interligados e sistemas de recolha de dados, sobre o quais é possível efetuar combinações e monitorizar os mesmos em tempo real, através de ferramentas de BI [117].
3. A fase de deteção de anomalias em tempo real, na qual as fábricas já possuem no mínimo um ano de histórico de dados provenientes dos sensores e dispositivos IoT e sobre os quais aplicam regras e modelos de ML. Estes modelos permitem "detetar anomalias nos processos de produção e disparar alarmes e alertas em tempo real" [117].
4. A fase de manutenção preditiva em tempo real, na qual a fábrica já reuniu um volume considerável de dados históricos, investiu na estruturação dos dados e melhorou os modelos preditivos anteriormente desenvolvidos para que sejam mais precisos. Com isto é possível gerar alertas em caso de deteção e previsão de anomalias. Nesta fase a fábrica já deverá ter um tempo de inatividade mínimo, tornando o negócio mais eficiente e menos dispendioso [117].
5. A fase de decisão autónoma em tempo real, na qual a fábrica possui diversos sistemas preditivos baseados em IA. Estes sistemas estão de tal forma precisos e interligados que alguns parâmetros podem ser alterados automaticamente sem intervenção humana. Nesta fase, a intervenção humana está principalmente ligada à definição dos objetivos dos sistemas de IA [117].

Posto isto, é possível identificar que o cliente se encontra na fase dois, isto é, na fase de monitorização em tempo real. Por forma a conseguir os benefícios totais da Indústria 4.0 e alcançar a quinta fase, este deverá ainda passar pelas anteriores. O resultado da presente dissertação colocará o cliente na terceira e quarta fase e irá contribuir ativamente para a melhoria contínua do modelo. Assim, será possível num futuro próximo o cliente alcançar a fase cinco de decisão autónoma em tempo real.

Numa análise de tendências tecnológicas e de investimento em tecnologia por parte das empresas é percetível que IA e ML ao longo dos últimos anos fixaram-se no topo [24]. A Gartner, uma empresa de pesquisa e consultoria, todos os anos elabora um relatório de tendências tecnológicas para os anos seguintes, bem como representa graficamente as expectativas para determinadas tecnologias. No fim do ano de 2020 as expectativas em torno dos termos *Machine Learning, AI Cloud Services, Deep Learning, Decision Intelligence* eram elevadas e o seu ponto alto deverá ser atingido num espaço de dois a cinco anos, como ilustrado na Figura 27. Segundo esta consultora, assiste-se atualmente a uma democratização da IA de tal forma que já não é um assunto apenas de especialistas, "agora, as organizações querem alcançar o próximo nível, entregando o valor da IA a mais pessoas" [25].

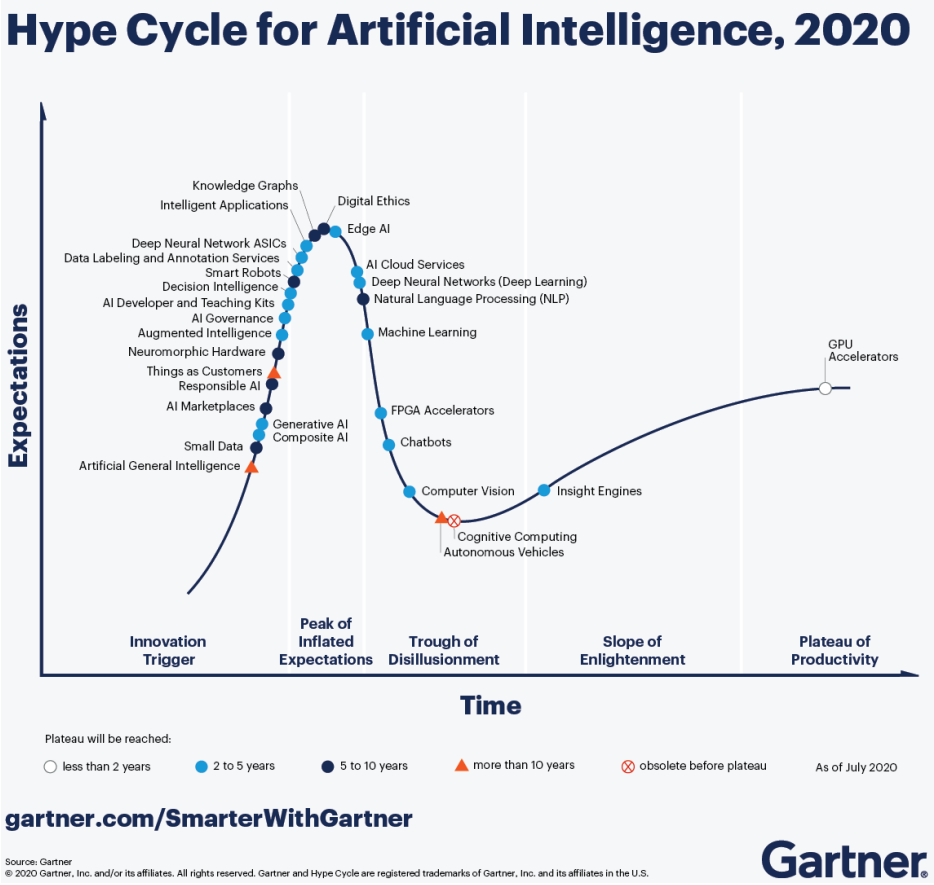


Figura 27 - Ciclo de expectativas da Inteligência Artificial em 2020 [25]

No presente ano estas tecnologias continuam no topo das tendências, como ilustrado na Figura 28. A partir desta é possível perceber que algumas das tendências para o ano de 2021 coincidem

com as tecnologias que compõem a Indústria 4.0, nomeadamente *Distributed Cloud, Cybersecurity, Hyperautomation* e *AI Engineering*.

A Gartner [26] e a Forbes [27] elegem IA no seu *ranking* de tendências para 2021. A Forbes vai mais longe e inclui a IA e o ML numa lista de tecnologias que irão definir a próxima década [28]. Importa salientar que a IA é uma tendência de tecnológica desde meados de 2016 e ganhou ainda maior escala com a pandemia vivida no ano de 2020. Como exemplo e justificação deste facto temos o caso da empresa chinesa Cheetah Mobile Inc que, como indicado anteriormente na secção 2.1.1, utilizou robôs com Inteligência Artificial para auxiliar no tratamento e mitigação da pandemia do Covid 19.

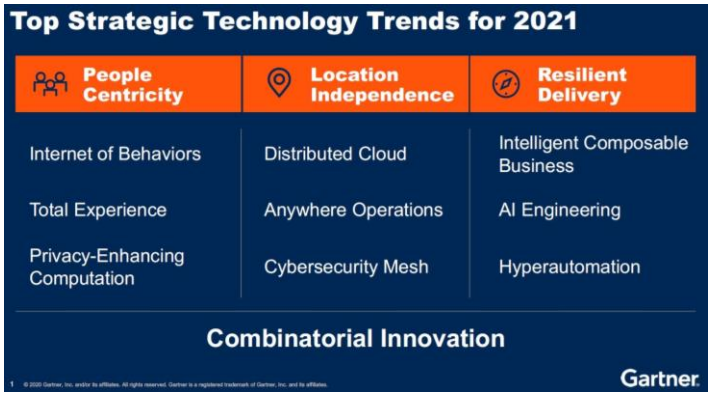


Figura 28 - Tendências Tecnológicas segundo a Gartner para 2021 [29]

A par das tendências tecnológicas, também o investimento por parte das empresas tem aumentado nos últimos anos. A empresa AI Multiple dedica-se a analisar o mercado da Inteligência Artificial. Esta recolheu informação de diversas fontes, nomeadamente de consultoras como a Forbes, PwC, Delloite e a Gartner e elaborou um estudo estatístico sobre a adoção e investimento em IA nas empresas [30]. Este estudo, resumido na Figura 29, indica que o investimento em Inteligência Artificial chegará aos 57,6 biliões de dólares em 2021 e que terá um retorno de 118,6 biliões de dólares por ano até 2025. Até 2030, o mercado de IA poderá valer mais de 15 triliões de dólares. Por outro lado, no que à adoção de soluções de IA diz respeito, 37% das empresas utilizam IA nos seus locais de trabalho, sendo que, estes números cresceram 270% entre 2015 e 2019. Por tudo isto, 83% das empresas indicam que IA é uma das prioridades estratégicas no presente e futuro [30].

Por outro lado, no seguimento do programa Capacitar i4.0 [31], que integra as iniciativas nacionais Indústria 4.0 e INCoDe.2030, o cliente está a apostar fortemente na digitalização do seu negócio tendo investido na sensorização de todas as máquinas que compõem o chão de fábrica de todas as suas unidades de negócio. Esta transformação digital tem em vista a Indústria 4.0 e claramente colher os benefícios da mesma. Assim, segundo a Forbes [117] este encontra-se na fase de monitorização em tempo real, na qual já possui tecnologias e mecanismos de BI (*Business Intelligence*) para análise de dados históricos com o objetivo de explicar factos passados. Existe, no entanto, uma vertente em falta para fechar o ciclo de analítica de dados, a análise preditiva utilizando tecnologias e mecanismos de IA e ML. Com isto

seria possível atingir a fase de previsão de anomalias em tempo real e aproximar o cliente cada vez mais da Indústria 4.0. Chegada a esta fase e segundo a pwc [116], o cliente conseguiria uma redução de custos e um aumento de eficiência no processo de produção e manutenção.

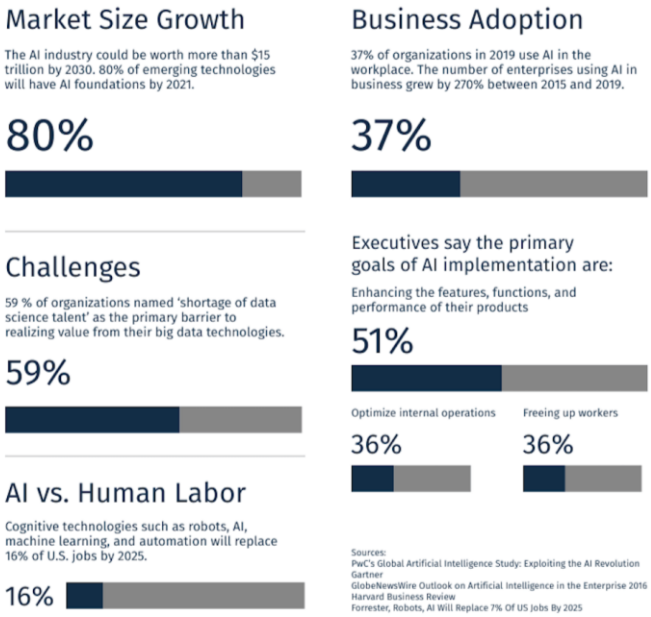


Figura 29 - Estudo sobre a Indústria de Inteligência Artificial [30]

Em suma, a oportunidade identificada trata-se em convergir tudo o que foi analisado anteriormente. Isto é, convergir uma tendência tecnológica em crescendo nos últimos anos e com perspetivas de futuro, com uma oportunidade de acompanhar os investimentos feitos na Indústria 4.0, com a necessidade do cliente em reduzir custos e aumentar a eficiência da produção, uma vez que todos estes factos estabelecem uma relação de sinergia.

3.3 Valor

O termo valor "tem sido definido em diferentes contextos teóricos como necessidade, desejo, interesse, standards/crenças, atitudes e preferências" [118]. Valor é assim um termo abstrato e ambíguo ao qual cada pessoa possui uma definição distinta, dependendo das suas necessidades e desejos. Nesta secção são apresentados e detalhados os termos valor para o cliente e valor percebido por este.

3.3.1 Valor para o Cliente

Segundo Shanker [119], o valor para o cliente possui dois aspetos diferentes, o valor desejado e o valor percebido. "O valor desejado refere-se ao que os clientes desejam num produto ou serviço. O valor percebido é o benefício que um cliente acredita que recebeu de um produto depois de o comprar." Assim, o valor para o cliente é uma razão entre o que este deseja e necessita e aquilo que realmente recebe [120].

3.3.2 Valor Percebido

O termo valor percebido foi definido por Zeithaml como "a avaliação geral do consumidor da utilidade de um produto com base nas perceções do que é recebido e do que é dado" [121]. Ainda, segundo Gutiérrez [122], valor percebido é a avaliação que o cliente faz de um produto, ponderando os benefícios e sacrifícios que este lhe traz.

A Tabela 7 pretende esquematizar os benefícios e sacrifícios que o presente projeto providencia ao cliente. Estes encontram-se divididos pelo domínio do produto resultante do projeto, serviço que é prestado ao longo do desenvolvimento do mesmo e pelo domínio da relação criada entre o cliente e a equipa de desenvolvimento.

Tabela 7 - Benefícios e Sacrifícios para o cliente

	Produto	**Serviço**	**Relação**
Benefícios	Customização de Produto	Responsividade	Confiança
	Qualidade do Produto	Flexibilidade	Solidariedade
	Soluções Alternativas	Competência Técnica	Imagem
	Preço		
Sacrifícios			Tempo/Esforço

No que diz respeito ao domínio do produto, o presente projeto é realizado por uma equipa interna da empresa OSI, no âmbito de uma dissertação de mestrado. Sendo assim, o cliente beneficia com este facto, uma vez que o conhecimento do projeto fica internalizado no seio do Grupo Amorim e haverá sempre espaço para a melhoria contínua, quer a nível de performance quer a nível confiabilidade do produto, assegurando a qualidade do mesmo. No seguimento deste facto, um outro benefício para o cliente é a componente preço. Isto porque, uma vez tratando-se de um projeto de dissertação de mestrado, os custos de análise, arquitetura e desenvolvimento do mesmo não serão assegurados pelo cliente. Este terá também como benefício a customização do produto, uma vez que todas as suas necessidades e requisitos serão satisfeitos e todos os critérios de decisão, ao longo do projeto, serão baseados no seu *feedback*. Antes de qualquer desenvolvimento será elaborado um estado de arte relativo à área na qual este projeto se insere, IA e ML. Sendo assim, o cliente terá sempre a oportunidade de avaliar soluções alternativas para a resolução do seu problema, revelando-se assim um outro benefício.

Ao nível do serviço prestado, dado que este projeto é realizado no seio do Grupo Amorim, este permite uma maior responsividade e flexibilidade para com o cliente. A equipa de projeto já se encontra perfeitamente entrosada no cliente, já conhece as pessoas chave envolvidas no negócio, bem como o modelo e as instalações do mesmo, revelando-se assim como outro benefício para o cliente.

Em termos de relação existem benefícios quer para o cliente quer para fornecedor, a empresa OSI representada pelo estudante. Este são a manutenção e a convalescença da relação entre

ambas as partes, aumentado a confiança e a solidariedade. Com este projeto o cliente verá a sua reputação ser aumentada, uma vez que irá reduzir o desperdício de dados de negócio e reaproveitá-los para potenciar a diminuição de custos e aumento de performance e eficiência com o conhecimento extraído.

Por fim, como principal sacrifício o cliente terá um consumo de tempo e esforço. As variáveis tempo e esforço serão usadas nas sessões de elaboração de requisitos, na discussão dos pormenores do projeto, no apoio ao esclarecimento de dúvidas e particularidades do negócio, nas sucessivas avaliações das entregas de funcionalidades ao longo do tempo e na testagem do produto.

3.4 Proposta de Valor

Segundo Alexander Osterwalder em "*Value Proposition Design: How to Create Products and Services Customers Want*" [32], proposta de valor "é uma visão geral do conjunto de produtos e serviços de uma empresa que são valiosos para o cliente" [32]. Ou seja, a proposta de valor define o valor que uma empresa pretender entregar aos clientes e tenta responder, de forma simples e clara, a questões sobre qual é o produto, qual ou quais são os clientes alvo do mesmo, qual o valor do produto e o que é que o torna único.

A *Value Proposition Canvas,* ilustrada na Figura 30, é uma ferramenta desenvolvida por Alexander Osterwalder que procura garantir que um determinado produto ou serviço está posicionado em torno dos valores do cliente [33]. Este grafismo permite combinar o produto com as necessidades e desejos do cliente, identificando claramente a que tipo de clientes o produto se adequa e por sua vez efetuar uma caracterização do valor do produto e do perfil de clientes [34]. A *Value Proposition Canvas* divide-se assim em dois segmentos, o do produto e o do cliente. Seguidamente será detalhada a Value Proposition Canvas nestes segmentos e consequentemente serão identificados os geradores de ganho, aliviadores de dor em relação ao produto e os ganhos e dores em relação ao cliente.

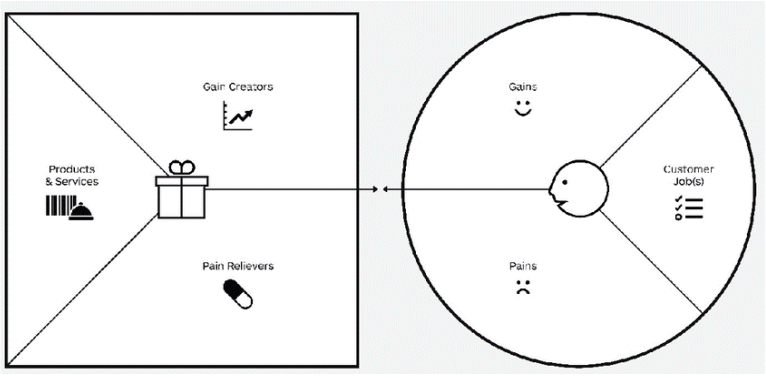


Figura 30 - Value Proposition Canvas [35]

Cliente

- **Ganhos** – detalha as expectativas do cliente em relação aos ganhos, isto é o que este deseja e necessita [33].
    - Obter conhecimento a partir de dados gerados;
    - Complementar o conjunto tecnológico existente com técnicas de ML;
    - Modernização do controlo de processo;
    - Planificar intervenções a manutenções.
    - Competitividade face à concorrência.
- **Dores** – detalha as experiências negativas, emoções e riscos que o cliente experiência [33].
    - Reatividade a anomalias (falha e/ou paragens) das máquinas;
    - Manutenção reativa e duradoura;
    - Não possuir informação clara e coesa sobre as falhas que ocorrem e que irão ocorrer;
    - Diferentes falhas com diferentes causas e dificuldade em encontrar padrões;
- **Trabalhos do Cliente** – detalha as tarefas que o cliente procura realizar, os eventuais problemas para os quais não possuem solução e as necessidades que deseja satisfazer [33].
    - Aproximação do conceito de Indústria 4.0;
    - Previsão de anomalias;
    - Manutenção Preditiva.

Produto

- **Produtos e Serviços** – descreve produtos e serviços que geram ganho e aliviam a dor, sustentando a criação de valor para o cliente [33].
    - Serviço preditivo para Limpeza e Tratamento de Rolhas de Cortiça.
- **Geradores de Ganho** – descreve a forma que como o produto ou serviço gera ganhos para o cliente [33].
    - Reutilização de dados gerados pelo negócio;
    - Obtenção de conhecimento;
    - Identificação de padrões de comportamento;
    - Melhoria contínua do processo de limpeza e tratamento;
    - Previsão de anomalias;

- Possibilidade de escalar serviço a mais máquinas do mesmo domínio.

- **Aliviadores de Dor** – descreve como o produto ou serviço alivia as dores e dificuldades do cliente [33].
  - Reação à mudança de comportamento das máquinas, com a aprendizagem contínua;
  - Manutenção preditiva

3.5 Function Analysis System Technique (FAST)

Charles Bytheway, em 1964, introduziu uma abordagem para a análise de produtos e processos que providencia uma representação gráfica da análise funcional dos mesmos. Esta representação denomina-se de diagrama FAST e organiza as funções de um determinado produto, processo ou serviço em estudo sobre a forma de questões "Como?" e "Porquê?" [36].

Com efeito, este modelo auxilia o processo de pensar no problema de forma mais objetiva e clara identificando todas as funcionalidades e promove a criatividade, para assim ser possível encontrar diferentes abordagens às funcionalidades [36].

Na Figura 31 é possível visualizar o modelo FAST aplicado ao presente projeto. Este trata-se de um projeto exploratório, em que o objetivo é desenvolver um serviço preditivo tendo por base a análise dos dados existentes e a análise de soluções para o problema. Posto isto, as funcionalidades foram descritas através das tarefas e etapas que irão culminar no serviço.

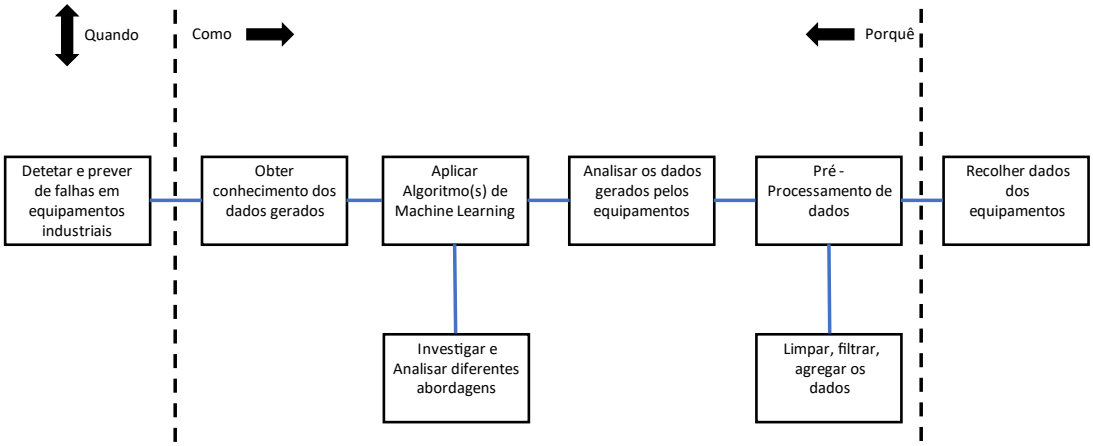


Figura 31 - Modelo FAST

3.6 Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS)

A presença de diversas opções para resolver um determinado problema, com múltiplos objetivos a satisfazer, torna o processo de decisão complexo. Para auxiliar nesta questão e para aumentar a confiabilidade e credibilidade da solução escolhida, existem métodos de apoio à decisão multicritério [37].

O método de apoio à decisão multicritério utilizado na presente dissertação para decidir qual das plataformas tecnológicas utilizar no desenvolvimento da mesma é o TOPSIS. Este método, introduzido por Hwang and Yoon em 1981, procura selecionar a alternativa com a distância euclidiana mais próxima da solução ideal e mais longe da solução ideal negativa [38].

Posto isto, existem como opções as plataformas AWS SageMaker, Azure ML e Google Cloud AI Platform, detalhadas na secção 2.4. Para os critérios de decisão foram elencados os seguintes:

- Facilidade de utilização – foi considerada a capacidade de possuir mecanismos de *drag and drop*, organização da interface e algoritmos pré-existentes para acelerar o desenvolvimento. Tendo em conta estes factos e visto que este é um ponto essencial para a futura manutenção e continuidade do sistema, foi atribuído um peso de 25%;
- Custo – visto que todas as plataformas são *cloud* e possuem o método de pagamento "pay as you go" torna-se difícil esta análise. No entanto foi fixado um nível de computação equivalente para todas as plataformas, a fim de se conseguir uma análise justa e equilibrada. Este critério tem elevada importância para o cliente e como tal foi atribuído um peso de 25%.
- Integração com terceiros – neste critério inclui-se a capacidade de integração com *frameworks* e librarias de terceiros. Este critério não é muito relevante para o cliente e como tal foi atribuído um peso de 10%;
- Integração e *deployment* contínuo – onde se inclui a capacidade ligação a repositórios de controlo de versões e a capacidade de *deployment*, a fim de implementar mecanismos de Integração Contínua e *Deployment* Contínuo. O cliente considera este facto interessante, mas não muito importante, como tal foi atribuído um peso de 15%;
- Adaptabilidade – neste critério insere-se a capacidade de adaptabilidade à infraestrutura atual do cliente, ou seja, quão fácil é colocar em funcionamento o presente serviço com a atual infraestrutura tecnológica. Este critério é essencial para o sucesso e futura escalabilidade do presente serviço, como tal foi atribuído um peso de 25%.

Na secção 2.4.4 são comparadas as várias plataformas e atribuídas classificações entre "Ótimo", "Bom", "Razoável" e "Mau". Assim, sabidas as opções, os critérios de decisão e os pesos dos mesmos, estas informações foram organizadas na Tabela 8 com a pontuação atribuída a cada par opção/critério. As pontuações foram inferidas das classificações previamente efetuadas na

secção 2.4.4. Como temos três alternativas (M) e cinco critérios (N), a Tabela 8 reflete a matriz de decisão multicritério (M x N).

Tabela 8 – TOPSIS Matriz de Decisão Multicritério

Pesos	**0,25**	**0,25**	**0,1**	**0,15**	**0,25**
	Facilidade de Utilização	Custo	Integração com terceiros	Integração e *Deployment* Contínuo	Adaptabilidade
AWS SageMaker	7	7	9	5	7
Azure ML	9	3	8	7	9
Google Cloud AI Platform	6	9	7	9	3

Seguidamente é necessário construir a matriz normalizada, ilustrada na Tabela 9. Sendo r_{ij} o valor normalizado presente em cada célula da tabela e x_{ij} o valor atribuído a cada par opção/critério, para construir esta matriz é necessário aplicar a fórmula:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\Sigma({x^2}_{ij})}}$$

Tabela 9 - TOPSIS Matriz Normalizada

Pesos	**0,25**	**0,25**	**0,1**	**0,15**	**0,25**
	Facilidade de Utilização	Custo	Integração com terceiros	Integração e *Deployment* Contínuo	Adaptabilidade
AWS SageMaker	0,543305368	0,593732251	0,646162343	0,401609664	0,593732251
Azure ML	0,698535473	0,254456679	0,574366527	0,56225353	0,763370037
Google Cloud AI Platform	0,465690315	0,763370037	0,502570711	0,722897396	0,254456679

Depois disto foi construída a matriz normalizada pesada, que consiste em multiplicar cada valor normalizado r_{ij} pelo peso do devido critério, tendo chegado à matriz da Tabela 10.

Tabela 10 - TOPSIS Matriz Normalizada Pesada

Pesos	**0,25**	**0,25**	**0,1**	**0,15**	**0,25**
	Facilidade de Utilização	Custo	Integração com terceiros	Integração e *Deployment* Contínuo	Adaptabilidade
AWS SageMaker	0,135826342	0,148433063	0,064616234	0,06024145	0,14843306
Azure ML	0,174633868	0,06361417	0,057436653	0,08433803	0,19084250
Google Cloud AI Platform	0,116422579	0,190842509	0,050257071	0,108434609	0,06361417

Analisando esta matriz normalizada pesada é possível obter o vetor da solução ideal positiva e o vetor da solução ideal negativa. Assumindo que o vetor da solução ideal positiva é dado por $A*$, o vetor da solução ideal negativa é dado por A', J é o conjunto de critérios positivos e que J' é o conjunto de critérios negativos, estes vetores são obtidos pelas fórmulas:

$$A* = \{v*_1, \dots, v*_n\}, onde\ v*_j = \{max\ (v_{ij})\ se\ j \in J;\ min\ (v_{ij})\ se\ j \in J'\}$$

$$A' = \{v'_1, \dots, v'_n\}, onde\ v'_j = \{min\ (v_{ij})\ se\ j \in J;\ max\ (v_{ij})\ se\ j \in J'\}$$

Assim, temos os seguintes vetores:

$$A* = 0{,}174633868;\ 0{,}06361417;\ 0{,}064616234;\ 0{,}108434609;\ 0{,}190842509$$

$$A' = 0{,}116422579;\ 0{,}190842509;\ 0{,}050257071;\ 0{,}06024145;\ 0{,}06361417$$

Assumindo que distância à solução ideal positiva é dada por $S*_i$ e distancia à solução ideal negativa é dada por S'_i, estas distâncias, ilustradas na Tabela 11 e Tabela 12 respetivamente, foram calculadas através das fórmulas:

$$S*_i = \left[\Sigma(v*_j - v_{ij})^2\right]^{1/2}$$

$$S'_i = \left[\Sigma(v''_j - v_{ij})^2\right]^{1/2}$$

Tabela 11 - TOPSIS Distância da Solução Ideal Positiva

Plataforma	$S*_i$
AWS SageMaker	0,113231667
Azure ML	0,02514342
Google Cloud AI Platform	0,189654529

Tabela 12 - TOPSIS Distância da Solução Ideal Negativa

Plataforma	S'_i
AWS SageMaker	0,09785447
Azure ML	0,190774334
Google Cloud AI Platform	0,04819316

Por fim, foi calculada a proximidade relativa à solução ideal ilustrada na Tabela 13, através da fórmula:

$$C*_i = \frac{S'_i}{S'_i + S*_i}$$

Como é possível perceber, através da análise dos valores obtidos na Tabela 13, o Azure ML tem uma distância de 0,88 à solução ideal para o problema em questão. Assim, esta plataforma é a eleita, através da técnica TOPSIS, como a mais vantajosa face às opções previamente estudadas.

Tabela 13 - TOPSIS Proximidade Relativa à Solução Ideal

Plataforma	Proximidade
AWS SageMaker	0,463576014
Azure ML	0,883550939
Google Cloud AI Platform	0,202621938

4 Desenho da Solução

O presente capítulo pretende detalhar a solução para o problema, numa primeira fase detalhando a disciplina de Engenharia de Requisitos e numa segunda fase desenhando a solução.

Na Engenharia de Requisitos são identificadas as partes interessadas do projeto e os intervenientes do sistema. Procede-se ainda ao levantamento dos requisitos funcionais, através de casos de uso, e dos requisitos não funcionais através do FURPS+.

Na secção de Arquitetura são apresentadas duas propostas de solução, sendo explicitadas e comparadas, com o objetivo de selecionar a que melhor se adequa ao problema. Selecionada a arquitetura, esta é detalhada através de um diagrama de camadas e diagrama de implantação.

4.1 Engenharia de Requisitos

Na presente secção é detalhada disciplina de Engenharia de Requisitos. São identificadas as partes interessadas da presente dissertação e os intervenientes do serviço a desenvolver no âmbito da mesma. Procede-se ainda ao levantamento dos requisitos funcionais, através de funcionalidades, e dos requisitos não funcionais através do FURPS+.

4.1.1 Partes Interessadas

Os negócios possuem diferentes tipos de partes interessadas, sejam internas ou externas, com diferentes interesses e prioridades. Entende-se por parte interessada, uma pessoa, grupo, entidade ou organização que tenha interesse num determinado negócio/projeto. Uma vez que, cada parte interessada é afetada, diretamente ou indiretamente, pelo projeto a realizar torna-se pertinente definir quem são [123].

Após uma breve reflexão foram identificadas as seguintes partes interessadas:

- Equipa de Manutenção dos equipamentos MLT – grupo de pessoas responsável pela manutenção dos equipamentos e que, à data, efetua uma manutenção reativa.
- Equipa de Controlo dos equipamentos MLT – grupo de pessoas responsável pelo correto funcionamento dos equipamentos MLT, que procura garantir a máxima performance dos equipamentos e qualidade de limpeza do produto
- Sistema Externo – sistema ou conjunto de sistemas externos que irão receber as previsões do serviço.
- Empresa – entidade máxima que lucra com a previsão de anomalias nos seus equipamentos, uma vez que isto se traduz numa diminuição de perdas monetárias.

4.1.2 Intervenientes do sistema

Os intervenientes do sistema (atores) são as partes interessadas que interagem com o serviço, fazendo uso das suas funcionalidades. No entanto, nem todas as partes interessadas são necessariamente intervenientes do serviço. Após uma conversa com o cliente, uma análise da mesma e visto que se trata de um serviço a ser consumido por terceiros, foram identificados como atores principais o próprio sistema e a equipa de controlo.

4.1.3 Requisitos Funcionais

Os requisitos funcionais de um sistema descrevem o que ele deve fazer, sendo que estes dependem do tipo de software a ser desenvolvido e de quem são os possíveis utilizadores do mesmo. Requisitos mal especificados podem ser a causa de problemas futuros de engenharia de software, uma vez que é possível interpretar um requisito ambíguo de uma forma que simplifique a sua implementação. Por este motivo, nesta secção são identificados os requisitos funcionais do serviço a desenvolver.

Um caso de uso descreve as interações com o sistema para se atingir determinado objetivo, detalhando a mesma interação. Ao conjunto de todos os casos de uso atribui-se a designação de modelo de casos de uso. Nesta secção será apresentado, através de um diagrama, o modelo de casos e posteriormente serão apresentadas descrições informais para cada caso de uso. Salienta-se que as descrições apresentadas nos casos de uso adotam um formato informal, cujo único propósito é proporcionar ao leitor uma compreensão dos objetivos respetivos.

Na Figura 32 apresentam-se os casos de uso relativos ao ator sistema:

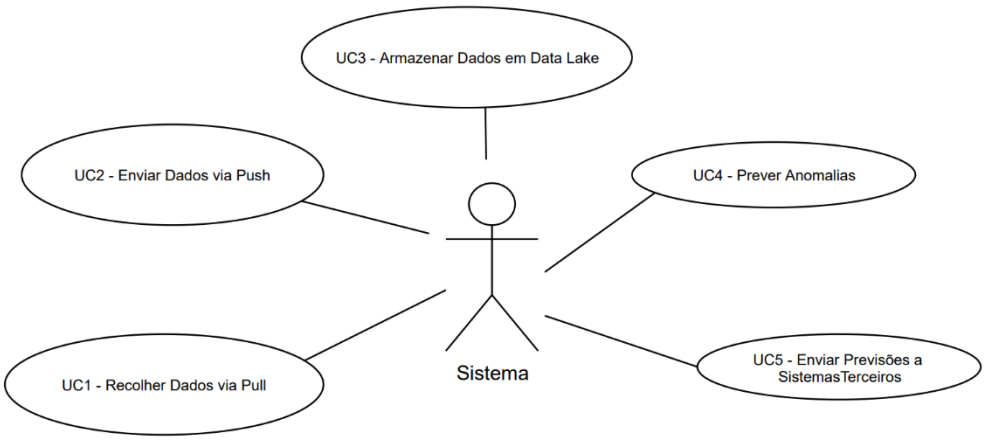


Figura 32 - Diagrama de Casos de Uso (Sistema)

UC1 Recolher Dados via *pull*

O sistema deve permitir recolher dados autonomamente. Este é um requisito de extrema importância, uma vez que a origem de dados SCADA não possui nenhum mecanismo para enviar

dados para o sistema. Deve ser possível efetuar recolha de dados várias vezes por dia, com um limite nunca inferior a 30 segundos entre cada recolha.

UC2 Enviar Dados para o Sistema via *push*

O sistema deve permitir o envio de dados para o mesmo. O SCADA, sistema que aloja os dados das máquinas, neste momento não se encontra dotado de mecanismos para enviar dados. No entanto, é algo que no futuro pode mudar. Sendo assim, numa ótica de tornar o presente sistema robusto e prevenir manutenção futura, esta funcionalidade torna-se necessária.

UC3 Armazenar Dados em *Data Lake*

Após a recolha e/ou receção de dados, o sistema deve armazená-los num *Data Lake*. Com o objetivo de centralizar todas as fontes de dados e tendo em vista a analítica avançada, o ator empresa possui na sua infraestrutura um *Data Lake* em *Azure*. Desta forma torna-se imperativo que os dados que circulam no sistema sejam armazenados no mesmo.

UC4 Previsão de Anomalias

O sistema deverá utilizar o um modelo preditivo, devidamente treinado e validado, para efetuar previsão de anomalias. Este é o requisito com maior relevo, uma vez que é o que providencia maior valor.

UC5 Envio de Previsões para Terceiros

O sistema deve enviar as falhas e paragens previstas para sistemas e/ou canais terceiros. Estes, deverão ser parametrizados no módulo de notificações e apenas deverão receber resultados positivos do modelo preditivo.

Na Figura 33 é apresentado o caso de uso relativo ao ator equipa de controlo:

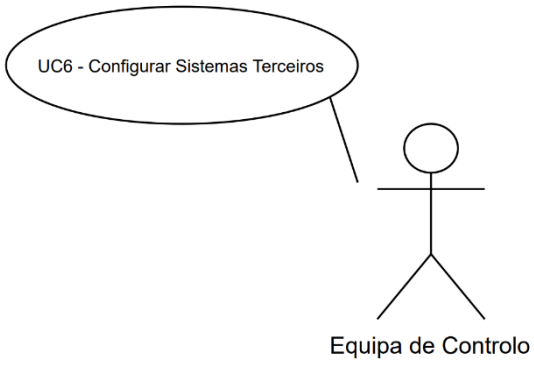


Figura 33 - Diagrama de Casos de Uso (Equipa de Controlo)

UC6 Configurar Sistemas Terceiros

Deve ser possível configurar no módulo de notificações os *endpoints* e/ou canais de sistemas terceiros que irão receber as previsões.

4.1.4 Requisitos Não Funcionais

Nesta secção são identificados os requisitos não funcionais, requisitos estes, que não estão diretamente ligados com o que o sistema faz, mas sim como o sistema deve funcionar. Estes são regularmente referidos como requisitos que impõem restrições ao sistema. O levantamento de requisitos é efetuado segundo a classificação FURPS+, que define estes requisitos como qualidades do sistema agrupados por categorias: Funcionalidade, Usabilidade, Confiabilidade, Desempenho, Suportabilidade, Restrições de Design, Restrições de Implementação, Restrições de Interface e Restrições Físicas [124].

Funcionalidade

- Alta *accuracy* global (>70%)

Usabilidade

-

Confiabilidade

- A taxa de disponibilidade do sistema deve ser muito elevada;
- As comunicações de dados dentro e fora da rede deverão seguir protocolos de segurança, nomeadamente *HyperText Transport Protocol Secure* (HTTPS).

Desempenho

- O modelo preditivo deve ser rápido a efetuar previsões, bastante próximo do tempo real;
- As previsões devem ter uma antecedência de 1 hora;
- O ciclo de recolha de dados não pode ser inferior a 30 segundos;
- A recolha de dados não deve criar constrangimentos na fonte de dados.

Suportabilidade

- O serviço preditivo deve ser desenvolvido de forma modular, tendo em vista a manutenção do mesmo;
- Devem ser adotadas as convenções de nomes utilizadas nos projetos anteriores do cliente;
- Não há qualquer restrição ao tipo de infraestrutura do projeto, *cloud* ou *on-prem*, ficando essa decisão a cargo da equipa de desenvolvimento. No entanto, caso seja selecionada a opção da *cloud*, todos os componentes criados deverão ser devidamente classificados com as *tags* corretas, para efeitos de rastreabilidade de custos;
- Idealmente os dados que servem de entrada ao modelo preditivo devem ser persistidos no *Data Lake* do cliente.

+ (Outros)

Restrições de Implementação

- Caso a infraestrutura do serviço preditivo seja *cloud*, apenas poderão ser utilizadas fornecedores *cloud* que o cliente já utiliza, como é o caso do *Azure* e *AWS*.

4.2 Arquitetura

A arquitetura de software corresponde a um conjunto de decisões importantes sobre a organização do sistema. Alguma dessas decisões passam pela seleção dos elementos estruturais, incluindo as interfaces pelas quais o sistema é composto e pela definição do comportamento entre esses elementos [125].

Nesta secção são apresentadas duas propostas de arquitetura do sistema e indicada a proposta selecionada. As propostas são apresentadas através de um diagrama de componentes, onde é mostrada a estrutura do sistema de software através de componentes, interfaces e as suas dependências. Posteriormente a proposta selecionada é detalha recorrendo a três diagramas UML, que providenciam várias visões do sistema.

Ambas as propostas têm em atenção os requisitos funcionais e não funcionais levantados na secção 4.1.3 e 4.1.4, bem como a tecnologia selecionada para o desenvolvimento do modelo preditivo na secção 2.4.4. Um outro detalhe que foi tido em conta no desenho das mesmas foi o facto do cliente possuir um *Data Lake* na *cloud.* Assim, por forma a agilizar as comunicações para dentro e fora da *cloud* e evitar problemas de latência nas mesmas, foi tomada a decisão de utilizar uma infraestrutura em Azure.

4.2.1 Proposta Cloud Pull

Como primeira proposta de arquitetura foi pensado um sistema *cloud* modular, ilustrado através de um diagrama de componentes na Figura 34. Neste, os dados das máquinas seriam recolhidos através do *Azure Data Factory*, através de uma pipeline. No entanto, para permitir a passagem dos dados para a *cloud* é necessário adicionar um *gateway*, o *Hybrid Gateway*. Este é responsável pelas comunicações, utilizando uma ligação HTTPS, entre a infraestrutura *cloud* e *on-prem*.

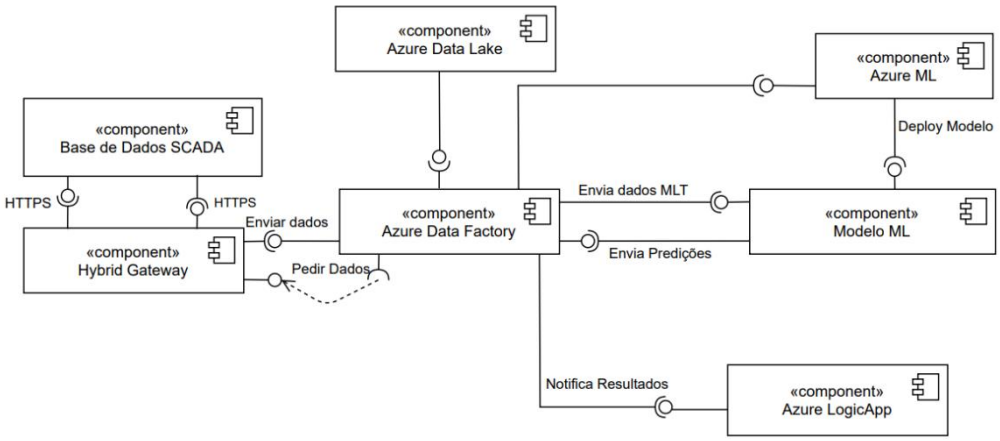


Figura 34 - Proposta de Arquitetura *Cloud Pull*

O cliente atualmente possui um *Data Lake* em Azure, com o objetivo de centralizar todos os dados importantes do negócio. Este permite um armazenamento escalável de dados estruturados, semiestruturados ou não estruturados [126]. Assim, uma vez que a escrita dos dados no *Data Lake* é uma das ambições expressas nos requisitos não funcionais, o *Azure Data Factory* é responsável por escrever os dados recolhidos no mesmo.

Posteriormente a pipeline *Azure Data Factory* envia os dados para o *Azure ML* e para o modelo preditivo. O *Azure ML* treina, valida e publica o modelo e este encontra-se a correr num *Azure ML Compute Cluster.* Os resultados preditivos são depois comunicados de volta à pipeline, que posteriormente os envia para uma *Azure Logic App*. Este, interpreta os resultados e comunica-os para diversos canais e sistemas terceiros que pretendam consumir o serviço preditivo. Neste último componente é possível efetuar uma configuração de *endpoints* de sistemas terceiros, bem como adicionar uma panóplia de canais de comunicação

A grande desvantagem desta arquitetura é o facto de apenas existir uma forma de interação com o serviço, via *pull de dados*.

4.2.2 Proposta Cloud Pull e Push

Com o objetivo colmatar a grande desvantagem da arquitetura proposta na secção acima, o facto de apenas possuir a recolha de dados (via *pull*), foi desenhada uma outra abordagem. Nesta, foi adicionada a possibilidade de o sistema SCADA enviar os dados para a pipeline *Data Factory* e consequentemente para o modelo preditivo. Embora o SCADA atualmente não possua mecanismos para enviar os dados (via *push*), esta arquitetura encontra-se apta para essa possibilidade e garante uma maior escalabilidade do que a anterior apresentada.

Assim, foi adicionado o componente *Azure Logic App* que funciona como uma REST API para receber os dados. A *Logic App* recebe os dados através de uma mensagem *json* pelo método POST. No entanto, para possibilitar a comunicação entre as duas infraestruturas, *cloud* e *on-prem*, é necessário adicionar um Gateway [127]. Este garante que as comunicações entre as

duas infraestruturas são realizadas numa ligação HTTPS. Posto isto, a *Logic App* invoca uma pipeline *Azure Data Factory* que envia os dados para o *Azure Data Lake* armazenando os mesmos em estado bruto.

O restante funcionamento é em tudo semelhante à arquitetura anteriormente proposta, ou seja, a pipeline *Azure Data Factory* envia os dados recebidos (via *push*) ou recolhidos (via *pull*) para o *Azure M*L com o objetivo de treinar o modelo. A *pipeline* também é responsável por enviar os dados mais recentes para o modelo preditivo, que após efetuar previsões retorna o resultado das mesmas para a *pipline*. Por fim, esta envia os resultados para uma *Azure Logic App* de notificação que envia a notificação preditiva para os canais e *endpoints* configurados nesta.

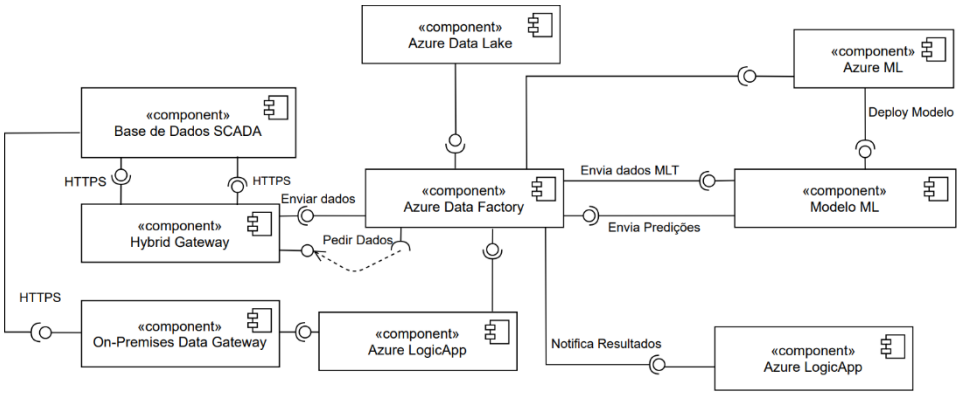


Figura 35 - Proposta de Arquitetura *Cloud Pull* e *Push*

A presente arquitetura possui a capacidade de aquisição de dados via *pull* e *push* em simultâneo e desta forma é possível cobrir a desvantagem apontada na proposta 4.2.1. Para além disso, importa salientar que é uma arquitetura modular, uma vez que é constituída por um aglomerado de componentes e serviços Azure, e tem como componente central de orquestração o *Azure Data Factory*. Sendo assim, esta é a arquitetura selecionada e pelo qual o desenvolvimento da presente dissertação se rege.

Numa perspetiva diferente, na Figura 36 é apresentado o diagrama de camadas da presente arquitetura. Este pretende demonstrar uma visão lógica da arquitetura do sistema. Nesta visão é possível perceber que existem cinco camadas, a camada de dados, camada de integração, camada de modelação, camada preditiva e a camada de comunicação.

A camada de dados contém todos os dados das máquinas MLT alojados, quer na base de dados do SCADA, quer no *Data Lake*. A única responsabilidade desta camada é armazenar estes dados e/ou fornecer os dados das máquinas à camada de integração.

A camada de modelação é responsável por tratar e limpar os dados, treinar, validar e publicar o modelo preditivo. Este é publicado na camada preditiva, cuja sua responsabilidade é efetuar previsões. Após as previsões, estas são comunicadas aos sistemas que pretendem consumir o serviço na camada de comunicação e configuração.

Os dados prevenientes da camada de dados, bem como os resultados gerados pela camada preditiva são integrados pelo *Azure Data Factory*, na camada de integração. Ou seja, esta camada é responsável por toda a orquestração e integração de dados nas diferentes camadas.

Por fim, a camada de configuração e comunicação é responsável por receber configurações de *endpoints* e canais de comunicação, para os quais deve enviar as previsões.

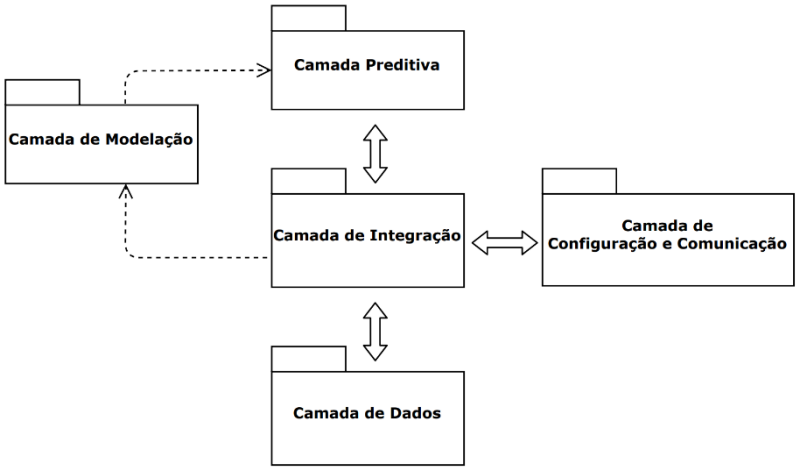


Figura 36 - Diagrama de camadas (proposta *cloud pull* e *push*)

Para finalizar a exposição da arquitetura do sistema, a Figura 37 ilustra o diagrama de implantação do mesmo. O nó do servidor local representa a origem dados, ou seja, a base de dados do SCADA que detém os dados das máquinas. Este comunica com a *cloud* através do nó *gateway* utilizando um dos dois componentes constituintes deste, dependendo se é via *push* ou *pull*. Esta comunicação entre o *gateway*, o servidor local e a *cloud* é sempre efetuada através de uma ligação HTTPS.

O componente *Azure Data Lake* representa o repositório dos dados na *cloud* e possui uma relação de dependência com o *Azure Data Factory*, que por sua vez depende da *Logic App* de *push*. Estes representam os mecanismos de captura de dados, via *pull* e *push* respetivamente.

O modelo preditivo estabelece uma relação de dependência com o *Azure ML* e com o *Azure Data Factory*, uma vez que este necessita da publicação de um modelo treinado e validado e de uma integração de dados para prever falhas. Por último a *Logic App* de notificação depende do *Azure Data Factory* e consequentemente do resultado do modelo preditivo.

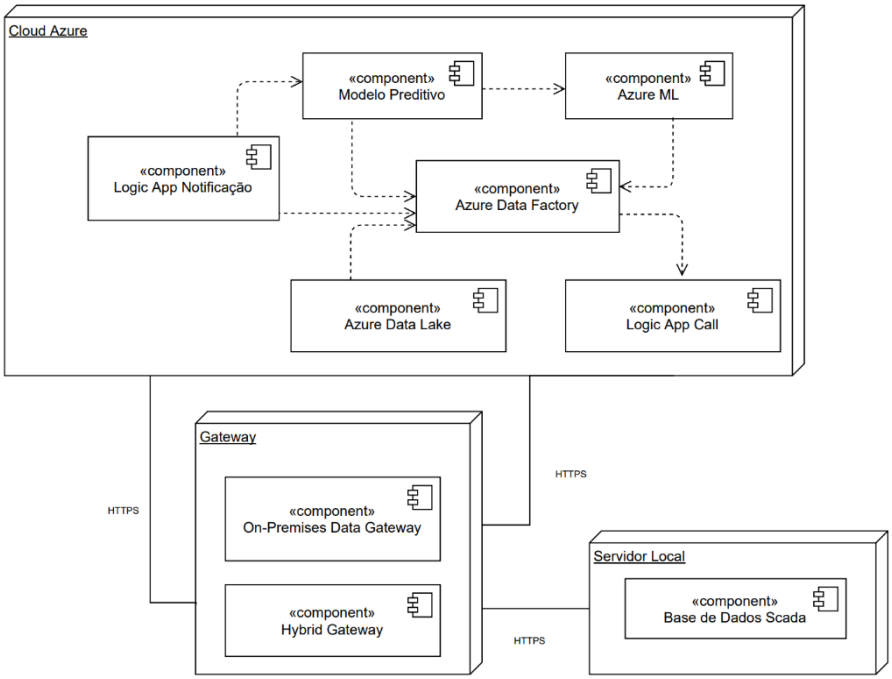


Figura 37 - Diagrama de Implantação (proposta *cloud pull* e *push*)

5 Implementação

O presente capítulo tem como objetivo demonstrar, explicar e justificar todo o processo de implementação do projeto. Assim, este encontra-se dividido sob a forma de secções tendo em conta a metodologia utilizada e descrita na secção 1.5, o CRISP-DM.

Na primeira secção é abordada a compreensão dos dados, onde primeiramente são descritas as diversas fontes de dados disponíveis. Posteriormente, é realizada uma caracterização e exploração do *dataset* construído através dessas mesmas fontes.

A segunda secção descreve e justifica todas as tarefas de preparação de dados realizadas no presente projeto.

A terceira secção aborda as diferentes pipelines criadas para o presente projeto, ou seja, a pipeline de treino, pipeline preditiva e a pipeline de orquestração em *Azure Data Factory*. Esta secção é particularmente importante, uma vez que é nesta que é dada visibilidade às tarefas abordadas na secção anterior.

Por fim, são explicados os testes realizados e demonstrados os seus resultados.

5.1 Compreensão dos Dados

Esta secção pretende expor os dados para que seja possível compreender melhor os mesmos. Para tal, são descritas as fontes de dados existentes e selecionadas as mais adequadas para o projeto. Posto isto, é explicada a construção do *dataset* através das diversas fontes de dados e é realizada uma análise, caracterização e exploração do mesmo.

5.1.1 Fontes de Dados

Para a elaboração do presente trabalho foi efetuado um levantamento de todas as fontes de dados que proliferam em torno das máquinas MLT. Sendo assim, foram encontradas as seguintes fontes:

- Sistema SCADA, que contém dados do processo como:
    - Ciclos de tratamento, etapa de tratamento, turno, ID da MLT, temperatura da cortiça, temperatura da câmara, pressão da bomba, pressão da câmara;
- Sistema SAP, que contém dados de avarias como:
    - ID da máquina, descrição de avaria, duração da avaria, data/hora de início da avaria, data/hora do fim da avaria, campo indicativo de paragem de máquina por avaria, duração da paragem;

- Ficheiro Excel, que contém dados não estruturados de avarias como:
    - ID da máquina, descrição da avaria e dia da avaria.

Tendo como base as fontes de dados elencadas na secção acima, estas foram analisadas com o objetivo de construir o *dataset* que servirá para treinar e validar o modelo.

5.1.1.1 Dados do Processo - SCADA

Os dados persistidos no sistema SCADA, exemplificados na Figura 38, são dados referentes ao processo de tratamento, isto é, dados de temperatura e pressão, ciclo e etapa de tratamento, turno, máquina MLT e data hora de cada registo. No total existem 18 máquinas MLT e a cadência temporal dos dados é de minuto a minuto por máquina. Ou seja, numa situação normal, em cada minuto existem 18 registos correspondentes às 18 máquinas existentes.

Machine	Cycle	Shift	Step	CorkTemp	ChamberTemp	ChamberPress	PumpPress	TimeStamp
MLT18	MLT18_120	T1	Unload2NextTreat	141	160	0.44278163	0.35627323	2021-04-29T17:27:00.000+0000
MLT13	MLT13_722	T1	Treatment	108	156	4.1371274	4.325492	2021-04-29T17:36:00.000+0000
MLT05	MLT05_620	T1	Treatment	131	155	21.247435	16.16015	2021-04-29T17:37:00.000+0000
MLT12	MLT12_626	T1	Treatment	141	155	21.136429	24.733028	2021-04-29T17:43:00.000+0000
MLT05	MLT05_620	T1	Treatment	132	155	22.156708	16.170744	2021-04-29T17:45:00.000+0000
MLT17	MLT17_120	T1	Treatment	119	161	1.2104905	0.6767883	2021-04-29T17:45:00.000+0000
MLT12	MLT12_710	T1	Treatment	139	155	25.273254	28.903738	2021-04-29T17:46:00.000+0000
MLT14	MLT14_692	T1	Treatment	101	155	3.8673236	4.0460463	2021-04-29T17:46:00.000+0000
MLT13	MLT13_636	T1	Treatment	114	156	3.0934608	3.3071122	2021-04-29T17:48:00.000+0000
MLT14	MLT14_606	T1	Treatment	99	156	3.7821815	4.230263	2021-04-29T17:49:00.000+0000
MLT07	MLT07_640	T1	Unload	151	151	1293.1384	1.654244	2021-04-29T17:50:00.000+0000

Figura 38 - Exemplo dos dados do processo

5.1.1.2 Dados de Avarias - SAP

Os dados das avarias persistidas em SAP encontravam-se divididos em duas tabelas distintas, uma tabela da Ordens de Manutenção e uma tabela de Equipamentos, como se verifica nas Figura 39 e Figura 40, respetivamente.

AvariaID	Descricao	Paragem	OrdemManutencao	InicioAvariaData	InicioAvariaHora	FimAvariaData	FimAvariaHora	DuracaoParagem	EquipamentoSAP	Empresa
200087865	EX_reparação bomba roots VSR06	0	200130480	2021-05-19	12:04:12	NULL	00:00:00	0	1007457	1400
10024515	EX_Manutenção prevent Bomba Busch VSR4	0	200119938	2021-04-13	12:28:45	2021-04-21	12:17:51	0	1007455	1400
10024418	EX_Alterar ventilador exaustão VSRs	0	200119581	NULL	00:00:00	NULL	00:00:00	0	1007469	1400
200081013	RV_Fuga de ar comprimir	0	200119535	2021-04-12	03:32:15	2021-04-13	14:06:46	0	2002394	1400
200079609	BC_VSR15 parado, dispara quadro elétrico	1	200117431	2021-03-31	22:34:45	2021-04-01	04:11:52	5,62	2002401	1400
10023990	VSR15	0	200117438	2021-03-31	19:54:35	2021-04-01	02:40:07	0	2002401	1400
200075577	TO_Resistência	1	200110763	2021-03-08	20:44:32	2021-03-08	23:28:36	2,73	2002393	1400
200073639	RV_Fuga de óleo no filtro	1	200108236	2021-02-24	08:19:40	2021-02-24	10:49:01	2,49	2002389	1400
10021559	RV_Correias	0	200099953	2021-02-12	09:04:45	2021-02-12	09:05:28	0	2002391	1400
10021511	Fuga ar	0	200099771	NULL	00:00:00	2021-02-11	12:01:37	0	2002394	1400
200071080	RV_Bomba roots avaria	1	200098959	2021-02-08	12:09:53	2021-03-02	10:05:58	525,93	2002393	1400
200068175	BC_Correia do ventilador arrebentou	1	200094500	2021-01-21	02:44:25	2021-01-21	06:00:00	3,26	2002389	1400
200066884	BC_Avaria no ventilador	1	200092238	2021-01-13	02:53:31	2021-01-13	07:30:00	4,61	2002397	1400

Figura 39 - Exemplo da Tabela de Ordens de Manutenção SAP

EquipamentoID	MLT	EquipamentoSAP	Descritivo
1	MLT01	2002388	MáquinaMLT
2	MLT02	2002389	MáquinaMLT
3	MLT03	2002390	MáquinaMLT
4	MLT04	2002391	MáquinaMLT
5	MLT05	2002392	MáquinaMLT
6	MLT06	2002393	MáquinaMLT
7	MLT05	1007472	Condensador
8	MLT06	1007472	Condensador
9	MLT03	1007471	Condensador
10	MLT04	1007471	Condensador
11	MLT01	1007470	Condensador

Figura 40 - Exemplo da Tabela de Equipamentos SAP

Tornou-se então necessário efetuar uma junção das duas tabelas para ser possível conjugar estes dados com os dados do processo, dando origem a uma vista única de avarias intitulada de "MLT_Avarias", como exemplificado na Figura 41.

MLT	AvariaID	Descricao	DataInicioAvaria	DataFimAvaria	DuracaoAvariaMinutos	DuracaoParagemMinutos	Paragem
MLT14	200065675	GV_Rolamento do ventilador estragado	2021-01-05T20:54:48.000+0000	2021-01-05T21:50:25.000+0000	55.61666666666667	55.800000000000004	true
MLT06	200066375	BC_Avaria motor ventilação	2021-01-10T22:07:50.000+0000	2021-01-11T03:25:00.000+0000	317.1666666666667	317.4	true
MLT06	200066696	EX_€Equip está indicação de falta fluido	2021-01-11T09:03:44.000+0000	2021-01-11T10:00:00.000+0000	56.266666666666666	0	false
MLT05	200066696	EX_€Equip está indicação de falta fluido	2021-01-11T09:03:44.000+0000	2021-01-11T10:00:00.000+0000	56.266666666666666	0	false
MLT06	200066460	TO_Curto circuito ventilador	2021-01-11T10:39:27.000+0000	2021-01-11T00:00:00.000+0000	-639.45	60	true
MLT10	200066884	BC_Avaria no ventilador	2021-01-13T02:53:31.000+0000	2021-01-13T07:30:00.000+0000	276.48333333333335	276.6	true
MLT02	200068175	BC_Correia do ventilador arrebentou	2021-01-21T02:44:25.000+0000	2021-01-21T06:00:00.000+0000	195.58333333333334	195.6	true
MLT06	200071080	RV_Bomba roots avaria	2021-02-08T12:09:53.000+0000	2021-03-02T10:05:58.000+0000	31556.083333333332	31555.799999999996	true
MLT07	10021511	Fuga ar	2021-02-11T00:00:00.000+0000	2021-02-11T12:01:37.000+0000	721.6166666666667	0	false
MLT04	10021559	RV_Correias	2021-02-12T09:04:45.000+0000	2021-02-12T09:05:28.000+0000	0.7166666666666667	0	false
MLT02	200073639	RV_Fuga de óleo no filtro	2021-02-24T08:19:40.000+0000	2021-02-24T10:49:01.000+0000	149.35	149.4	true
MLT06	200075577	TO_Resistência	2021-03-08T20:44:32.000+0000	2021-03-08T23:28:36.000+0000	164.06666666666666	163.8	true
MLT14	10023990	VSR15	2021-03-31T19:54:35.000+0000	2021-04-01T02:40:07.000+0000	405.53333333333336	0	false
MLT14	200079609	BC_VSR15 parado, dispara quadro elétrico	2021-03-31T22:34:45.000+0000	2021-04-01T04:11:52.000+0000	337.1166666666667	337.2	true

Figura 41 - Exemplo Vista consolidada de Avarias e Equipamentos SAP

5.1.1.3 Dados de Avarias - Excel

O presente Excel servia para ajudar a equipa de manutenção a registar as avarias enquanto as MLT's não estavam ligadas ao sistema SAP. Os colaboradores apenas registavam a máquina na qual ocorreu a avaria, o dia e o motivo. Uma vez que os dados do processo estão registados com uma cadência por minuto, possuir apenas o dia da avaria torna-se muito redutor. Ou seja, os dados presentes no Excel têm pouca qualidade uma vez que se traduzem em pouca informação, face ao que é possível extrair de SAP. Sendo assim, esta fonte de dados não foi tida em consideração para o presente projeto.

5.1.2 Dataset

Após a análise das diversas fontes de dados disponíveis e seleção das mesmas, estas foram migradas para o *Data Lake* do cliente com o objetivo de analisar os dados e construir o *dataset* no *Azure DataBricks*, utilizando *Spark SQL*. Como tal, para construir o *dataset* foi realizada uma junção entre a tabela de dados do processo e a vista consolidada "MLT_Avarias", cujo *script* se encontra no Excerto de Código 1.

```
SELECT s.Machine, s.Shift, s.Step, s. CorkTemp, s. ChamberTemp, s.ChamberPress,
s. PumpPress, s.TimeStamp, a.AvariaID as FailureID, a.Descricao as Description, a.
DataInicioAvaria as StartFailure, a.DataFimAvaria as EndFailure, a.DuracaoAvariaMinutos as
FailureDurationMinutes, a.Paragem as Stop, a.DuracaoParagemMinutos as
StopDurationMinutes
FROM DadosProcesso as s
LEFT JOIN MLT_Avarias as a ON s.Machine=a.MLT and
DATE_TRUNC('Minute',LocalTimeCol) BETWEEN DATE_TRUNC('Minute', a.DataInicioAvaria)
and  DATE_TRUNC('Minute',a.DataFimAvaria)
```

Excerto de Código 1 – *Script SparkSQL* de Construção do *Dataset*

Esta junção para além de renomear as colunas da vista "MLT_Avarias" para nomes mais coerentes no que à linguagem diz respeito, tem a particularidade de posicionar cada anomalia no seu *timestamp* correto. Isto é, cada registo cujo *timestamp* se encontre entre as colunas DataIniciaAvaria e DataFimAvaria é marcado como sendo uma anomalia com o devido FailureID e restantes campos da vista de avarias. Desta forma, cada registo do *dataset* caso tenha a variável *FailureID* preenchida indica a presença de uma anomalia. Assim, caso determinado registo esteja marcada como anomalia, é possível saber se é uma falha ou paragem, através da variável *Stop*.

Como resultado desta junção, obteve-se o *dataset* presente na Figura 42.

Cycle	Machine	Shift	Step	CorkTemp	ChamberTemp	ChamberPress	PumpPress	TimeStamp	FailureID	Description	StartFailure	EndFailure	FailureDurationMinutes	Stop	StopDurationMinutes
MLT07_586	MLT07	T2	Treatment	51	136	10.000271	4.1371274	12/04/2021 04:32	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	10.000271	4.1371274	12/04/2021 04:32	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	13.488478	2.8317602	12/04/2021 04:33	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	136	13.488478	2.8317602	12/04/2021 04:33	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	136	21.136429	3.4599485	12/04/2021 04:34	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	21.136429	3.4599485	12/04/2021 04:34	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	21.136429	3.377137	12/04/2021 04:35	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	136	21.136429	3.377137	12/04/2021 04:35	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	136	20.617031	3.368304	12/04/2021 04:36	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	20.617031	3.368304	12/04/2021 04:36	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	12.926471	4.325492	12/04/2021 04:37	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	136	12.926471	4.325492	12/04/2021 04:37	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	136	11.31762	4.0460463	12/04/2021 04:38	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	136	11.31762	4.0460463	12/04/2021 04:38	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	51	135	10.810613	4.422864	12/04/2021 04:39	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	51	135	10.810613	4.422864	12/04/2021 04:39	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	136	10.339847	4.325492	12/04/2021 04:40	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	136	10.339847	4.325492	12/04/2021 04:40	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	135	10.000271	3.6964922	12/04/2021 04:41	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	135	10.000271	3.6964922	12/04/2021 04:41	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	136	9.671858	3.7821815	12/04/2021 04:42	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	136	9.671858	3.7821815	12/04/2021 04:42	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	136	18.27699	3.2343035	12/04/2021 04:43	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	136	18.27699	3.2343035	12/04/2021 04:43	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	136	21.136429	3.3837757	12/04/2021 04:44	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	136	21.136429	3.3837757	12/04/2021 04:44	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	136	20.671099	3.242786	12/04/2021 04:45	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	136	20.671099	3.242786	12/04/2021 04:45	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_578	MLT07	T2	Treatment	52	135	17.28758	3.3071122	12/04/2021 04:46	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0
MLT07_586	MLT07	T2	Treatment	52	135	17.28758	3.3071122	12/04/2021 04:46	200081013	RV_Fuga de ar co	12/04/2021 04:32	13/04/2021 15:06	2074.516666666667	FALSO	0

Figura 42 – Exemplo do *Dataset* construído

5.1.2.1 Caracterização do Dataset

Ao todo existem 7 537 099 registos no *dataset*, com um total de 16 variáveis descritas na Tabela 14.

Tabela 14 – Descrição das colunas do *Dataset*

Nome Coluna	Descrição	Tipo de Dados
Machine	Nome da máquina MLT (ex: MLT01)	*String (not nullable)*
Cycle	Ciclo da máquina (cada conjunto de carga+tratamento+descarga representa um ciclo) EX: MLT01_142	*String (nullable)*
Shift	Turno da fábrica (T1 ou T2)	*String (not nullable)*
Step	Fase do ciclo (Carga, tratamento ou descarga)	*String (nullable)*
CorkTemp	Temperatura da Cortiça em ºCelsius	*Integer (nullable)*
ChamberTemp	Temperatura da Câmara em ºCelsius	*Intiger (nullable)*
ChamberPress	Pressão da Câmara em mBar	*Float (nullable)*
PumpPress	Pressão da Bomba em mBar	*Float (nullable)*
TimeStamp	Data e Hora em que foi registada a leitura das Pressões e Temperaturas no SCADA	*Datetime (not nullable)*
FailureID	ID da falha (caso exista)	*Int (nullable)*
Stop	*Flag* que indica se a falha deu origem a uma paragem (*true/false/null*)	*Bool (nullable)*
Description	Descrição da falha (caso exista)	*String (nullable)*
StartFailure	Data e Hora de início da falha (caso exista)	*Datetime (nullable)*
EndFailure	Data e Hora de fim da falha (caso exista)	*Datetime (nullable)*
FailureDurationMinutes	Duração da falha (caso exista)	*Float (nullable)*
StopDurationMinutes	Duração da paragem (caso exista)	*Float (nullable)*

Após uma breve descrição de todas as variáveis constituintes do *dataset*, foi realizada uma análise individual a cada uma delas. Nesta foram visualizados exemplos de cada variável, realizadas contagens de valores nulos e distintos, com o intuito de compreender melhor aos dados. Esta análise encontra-se exposta na Tabela 15 e permitiu conhecer melhor cada uma das variáveis.

Tabela 15 – Análise das colunas do *Dataset*

Coluna	Análise
Machine	• Nº de Registos com valor (sem nulos ou vazio) – 7 537 099 • Nº de Registos sem valor (nulo ou vazio) – 0 • Nº de Valores Distintos – 18 • Valores exemplo – MLT01, MLT02 (máquinas 1 e 2)
Cycle	• Nº de Registos com valor (sem nulos ou vazio) – 6 621 999 • Nº de Registos sem valor (nulo ou vazio) – 915100 • Nº de Valores Distintos – 10 647 • Valores exemplo: MLT01_10, MLT02_10 (máquinas 1 e 2 no ciclo 10)
Shift	• Nº de Registos com valor (sem nulos ou vazio) – 7 537 099 • Nº de Registos sem valor (nulo ou vazio) – 0 • Nº de Valores Distintos – 2 • Valores exemplo – T1, T2
Step	• Nº de Registos com valor (sem nulos) – 6 621 851 • Nº de Registos sem valor (nulo ou vazio) – 915 248

	• Nº de Valores Distintos – 4 • Valores exemplo – Treatment, Treat2Unload
CorkTemp	• Nº de Registos com valor (sem nulos) – 6 817 944 • Nº de Registos sem valor (nulo ou vazio) – 719 155 • Nº de Valores Distintos – 158 • Valores exemplo – 89; 63
ChamberTemp	• Nº de Registos com valor (sem nulos) – 6 817 948 • Nº de Registos sem valor (nulo ou vazio) – 719 151 • Nº de Valores Distintos – 190 • Valores exemplo – 12; 103
ChamberPress	• Nº de Registos com valor (sem nulos) – 6 817 941 • Nº de Registos sem valor (nulo ou vazio) – 719 158 • Nº de Valores Distintos – 13 320 • Valores exemplo – 890; 1 200
PumpPress	• Nº de Registos com valor (sem nulos) – 6 817 943 • Nº de Registos sem valor (nulo ou vazio) – 719 156 • Nº de Valores Distintos – 16 600 • Valores exemplo – 490; 1 400
TimeStamp	• Nº de Registos com valor (sem nulos) – 7 537 099 • Nº de Registos sem valor (nulo ou vazio) – 0 • Nº de Valores Distintos – 359 415 • Valores exemplo – 2021-03-25 16:02:00
FailureID	• Nº de Registos com valor (sem nulos) – 7 501 503 • Nº de Registos sem valor (nulo ou vazio) – 35596 • Nº de Valores Distintos – 15 • Valores exemplo – 200 075 577; 10 021 559
Stop	• Nº de Registos com valor (sem nulos) – 35 596 • Nº de Registos sem valor (nulo ou vazio) – 7 501 503 • Nº de Valores Distintos – 2 • Valores exemplo – 0; 1
Description	• Nº de Registos com valor (sem nulos) – 35 596 • Nº de Registos sem valor (nulo ou vazio) – 7 501 503 • Nº de Valores Distintos – 15 • Valores exemplo – “BC_Avaria motor ventilação”
StartFailure	• Nº de Registos com valor (sem nulos) – 35 596 • Nº de Registos sem valor (nulo ou vazio) – 7 501 503 • Nº de Valores Distintos – 15 • Valores exemplo – 2021-01-05 20:54:00

EndFailure	• Nº de Registos com valor (sem nulos) – 35 596 • Nº de Registos sem valor (nulo ou vazio) – 7 501 503 • Nº de Valores Distintos – 15 • Valores exemplo – 2021-01-05 21:50:00
FailureDurationMinutes	• Nº de Registos com valor (sem nulos) – 35 596 • Nº de Registos sem valor (nulo) – 7 501 503 • Nº de Valores Distintos – 15 • Valores exemplo – 317,16; 55,61
StopDurationMinutes	• Nº de Registos com valor (sem nulos) – 35 596 • Nº de Registos sem valor (nulo) – 7 501 503 • Nº de Valores Distintos – 8 • Valores exemplo – 55,80; 163,8

5.1.2.2 Exploração do Dataset

Uma vez efetuada a construção e caracterização do *dataset*, torna-se necessário realizar uma exploração do mesmo, com o objetivo de detetar e perceber padrões nos dados, bem como relações entre eles. Para tal, foi utilizada a ferramenta *Microsoft PowerBI*, onde foram construídos diversos gráficos com a finalidade de analisar, compreender e explorar os dados.

Primeiramente foi tomada a decisão de se aferir qual o espectro temporal do *dataset*. Assim, depois de calcular o mínimo e máximo da variável *Timestamp*, chegou-se à conclusão de que a dimensão temporal dos dados é de cerca de 11 meses, entre 2020-07-15 e 2021-05-25.

De seguida, foram analisadas as variáveis de temperatura e pressão do processo de limpeza e tratamento. Iniciando pela variável *CorkTemp*, foi construído o gráfico ilustrado na Figura 43. Este providencia informações sobre o máximo, mínimo, média e mediana da presente variável. Através da leitura deste gráfico é possível aferir que os dados desta variável apresentam uma assimetria negativa, dado que a mediana se encontra mais próxima do terceiro quartil. Adicionalmente, foram calculadas duas métricas que o presente gráfico não providenciava:

- Média – 56 270,33[3]
- Desvio Padrão – 22 089,84[3]

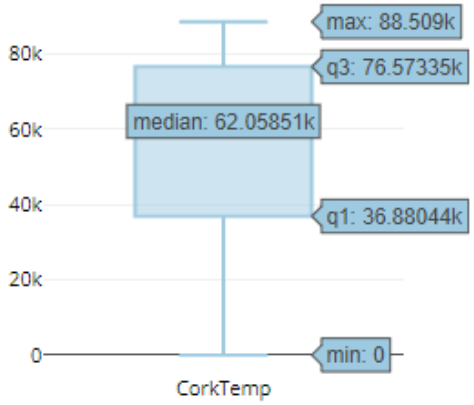


Figura 43 – *Box Plot* da Variável *CorkTemp*[3]

A segunda variável explorada foi a *ChamberTemp*, tendo sido adotada a mesma estratégia, isto é, foi construído o gráfico *box plot* representado na Figura 44. Através da leitura do presente gráfico é percetível que os valores máximos e mínimos se encontram muito distantes da mediana da população. Adicionalmente, foi também calculada a média e o desvio padrão:

- Média – 78 370,62[3]
- Desvio Padrão – 15 666,67[3]

Após analisar ao detalhe os valores mínimos e máximos, percebeu-se que estes se encontram 108 desvios padrão afastados da média. Estes aconteceram apenas na máquina MLT09 e durante um intervalo de tempo de 40 minutos em todo o *dataset*. Isto poderá indicar a presença de *outliers*.

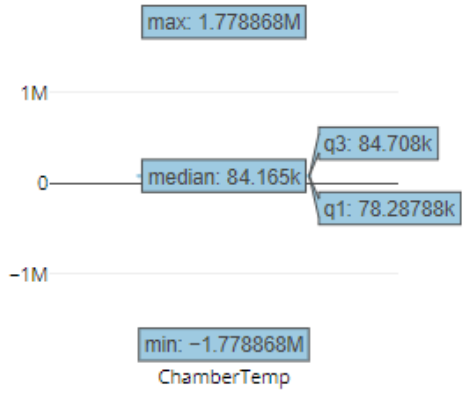


Figura 44 - *Box Plot* da Variável *ChamberTemp*[3]

De seguida, foi analisada a variável *ChamberPress*, tendo sido elaborado o gráfico da Figura 45. Através deste é possível depreender que a presente variável possui uma distribuição

[3] Por motivos de confidencialidade, os valores foram mascarados.

assimétrica positiva, dado que a mediana se encontra mais próxima do primeiro quartil. Foram adicionalmente, calculadas as seguintes métricas:

- Média – 228 039,24[3]
- Desvio Padrão – 311 620,65[3]

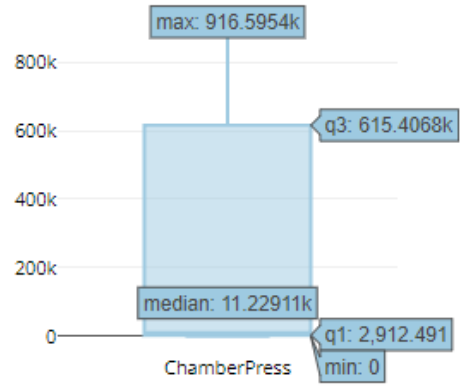


Figura 45 – *Box Plot* da Variável *ChamberPress*[3]

Em relação à variável *PumpPress*, foi construído o gráfico *box plot* da Figura 46. Através deste é percetível uma distribuição assimétrica positiva. Para além da construção do presente gráfico, foi também calculada a média e o desvio padrão:

- Média – 33 215,10[3]
- Desvio Padrão – 130 193,88[3]

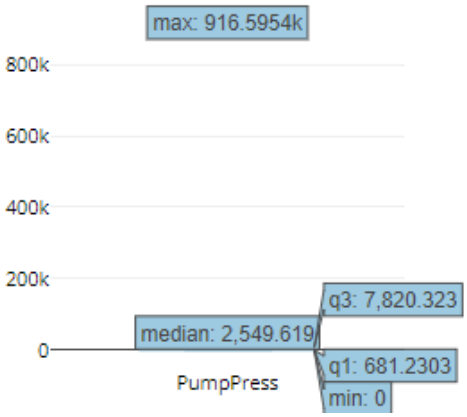


Figura 46 - *Box Plot* da Variável *PumpPress*[3]

Após analisar as variáveis de pressão e temperatura, foram aferidas quais os valores mais comuns das variáveis Shift e Step. Assim, através da Figura 47, é possível perceber que no que diz respeito à variável Shift, esta possui relativamente o mesmo volume de dados para ambos os turnos. Contudo o turno "T2", possui um volume ligeiramente superior de 3 771 249 registos, face ao "T1" com 3 765 850.

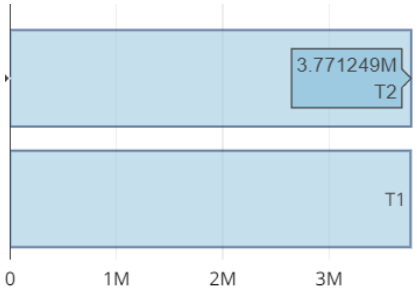


Figura 47 - Contagem da Variável *Shift*

Em relação à variável Step, a partir do gráfico da Figura 48 é percetível a existências de 4 etapas, existindo uma quinta, que se refere a valores nulos. A etapa mais comum com 4 854 695 registos é a etapa "*Treatment*". Segundo o cliente, esta refere-se ao funcionamento em pleno das máquinas de tratamento.

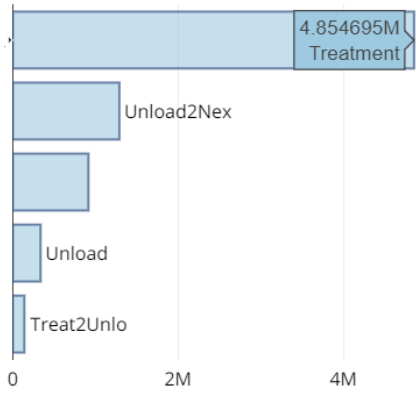


Figura 48 - Contagem da Variável *Step*

Posto isto, foi decidido abordar as variáveis que dizem respeito às anomalias, ou seja, as variáveis *FailureID, Stop, FailureDurationMinutes* e *StopDurationMinutes.* Para tal, foi elaborado o gráfico presente na figura Figura 49, onde é visível a existência de um total de 15 anomalias, 8 paragens e 7 falhas.

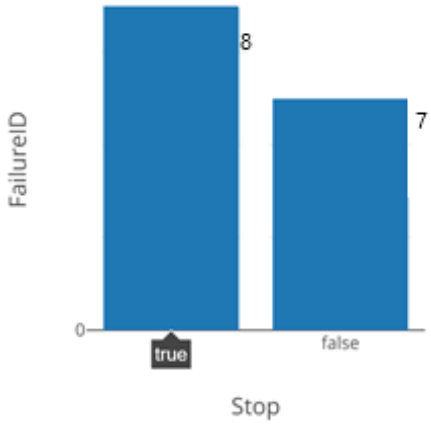


Figura 49 - Contagem de Anomalias por Paragem

Contudo, analisando o *dataset*, existem mais registos marcados com paragem, cerca de 30 215, face aos registos marcados com falha, cerca de 5 381. Dado que apenas existe mais uma paragem face às falhas, isto pode indicar que as paragens possuem uma duração maior. Desta forma, foi calculado o mínimo, máximo, média, mediana e desvio padrão da duração das falhas e paragens, representado na Tabela 16. Através desta, é possível aferir que as paragens possuem uma maior duração face às falhas. Confrontando o cliente com esta informação, este confirmou-a. Para além, disso o cliente justificou dizendo que a falha pode ser corrigida de imediato pelo operador da máquina, caso ele a detete, ao passo que a paragem implica intervenção da equipa de manutenção para avaliação, deteção e resolução do problema.

Tabela 16 - Comparação das Métricas de Falha e Paragem

Métrica	Falha	Paragem
Mínimo	0,72	55,8
Máximo	2 074,52	31 555,80
Média	1 727,14	30 153,65
Mediana	2 074,52	31 555,80
Desvio Padrão	646,30	6 473,61

Deste modo, foi construído o gráfico da Figura 50 para aferir qual a duração média de paragem por máquina. Através deste consegue-se saber que no presente *dataset* existiram falhas nas máquinas MLT02, MLT06, MLT10 e MLT14, sendo que a MLT06 possuiu a maior duração média de paragem, de cerca de 517 minutos.

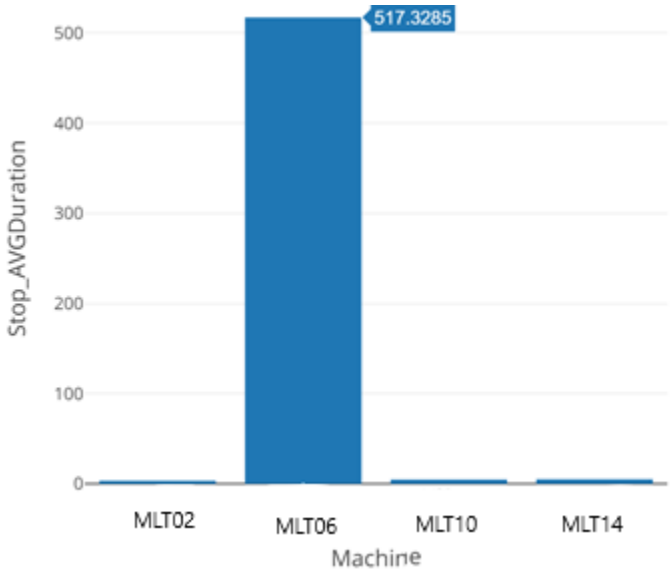


Figura 50 - Duração Média de Paragem por Máquina

No que diz respeito às falhas, foi elaborado o gráfico da Figura 51, para se realizar a análise da duração média de falha por máquina. Com este, é percetível que as máquinas MLT04, MLT05,

MLT06, MLT07 e MLT14 tiveram falhas. A máquina com maior duração média de falha é a MLT07, de cerca de 31 minutos.

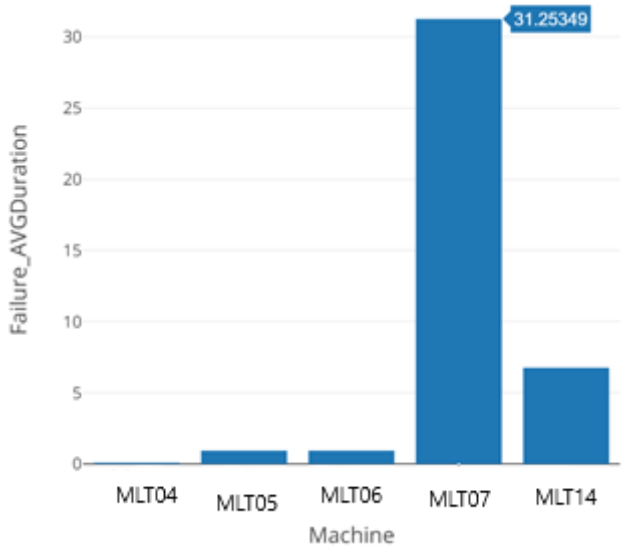


Figura 51 - Duração Média de Falha por Máquina

Posto isto, foi tomada a decisão de analisar a dispersão de temperaturas e pressões através da mediana. Isto porque, esta métrica é mais robusta do que a média a *outliers*, sendo mais fácil a sua deteção graficamente. Assim, foi construído um gráfico de dispersão entre as medianas das variáveis de temperatura da câmara e da cortiça, ilustrado na Figura 52. Este gráfico permitiu visualizar valores excessivamente altos e que fugiam aos valores padrão de temperatura para a variável *ChamberTemp*. Esta análise está em concordância com o que foi identificado previamente na Figura 44 e suporta a presença de *outliers* nesta variável.

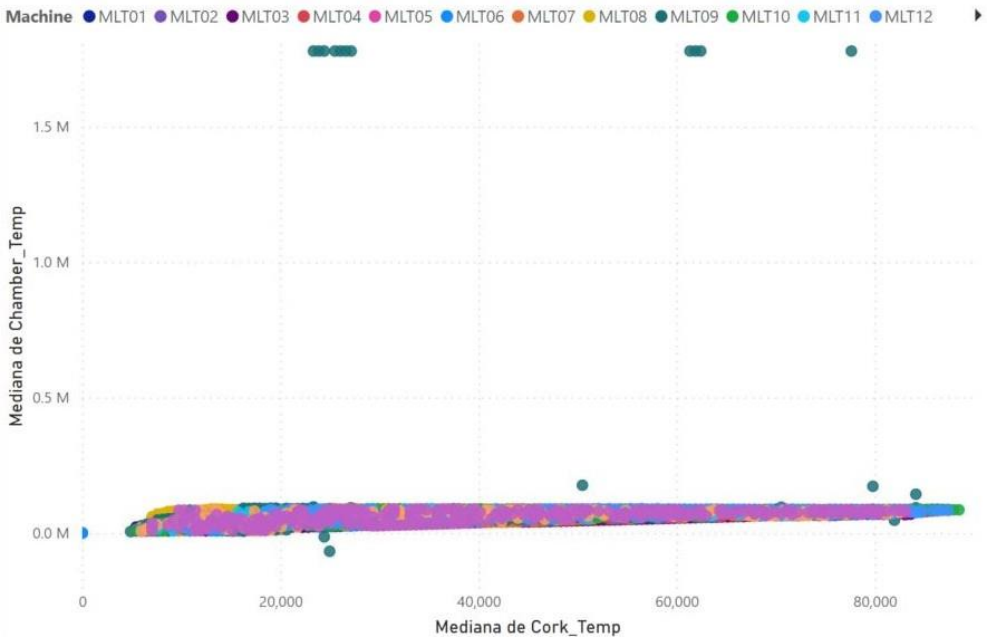


Figura 52 - Dispersão de Temperaturas (*Outliers*) [3]

Contudo, apesar de o gráfico da Figura 52 permitir visualizar os *outliers*, não permite identificar nenhum padrão, dado o elevado número de máquinas. Sendo assim, foram elaborados gráficos de dispersão individuais por máquina para perceber o padrão de cada uma e comparar. A Figura 53 e Figura 54 ilustram a comparação realizada. Através destas é possível comprovar que as diversas máquinas possuem padrões de temperatura idênticos. Ou seja, à medida que a temperatura da câmara aumenta, a temperatura da cortiça também aumenta. Assim que a temperatura da câmara estabiliza num determinado valor, a temperatura da cortiça continua a aumentar até que a temperatura da câmara começa a diminuir e progressivamente a temperatura da cortiça diminui também.

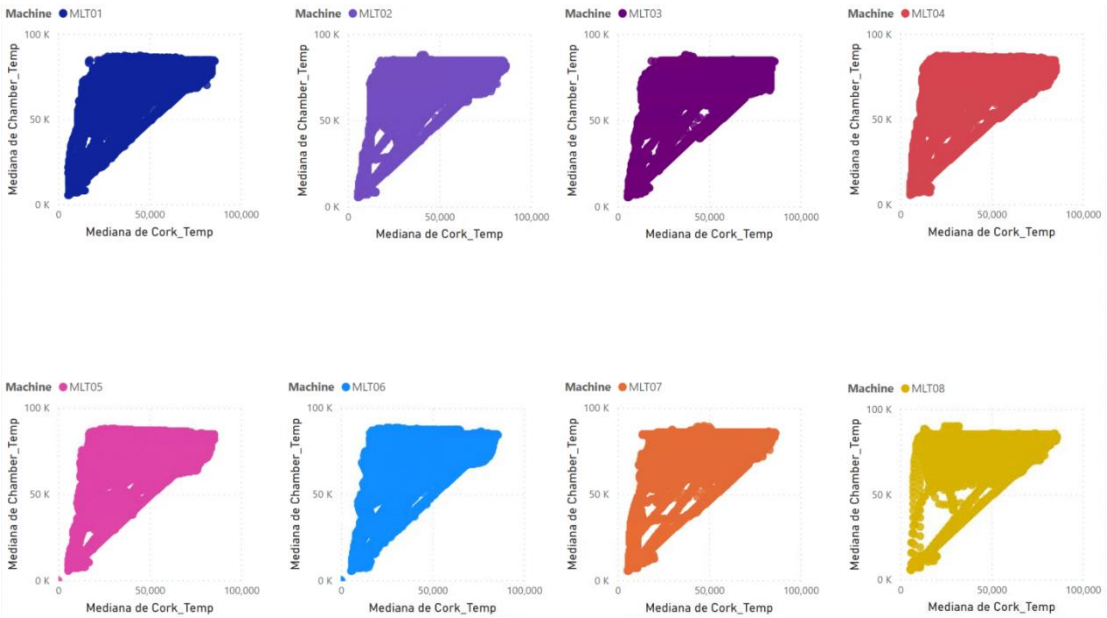


Figura 53 - Comparação de Temperaturas por máquina (MLT01 a MLT08) [3]

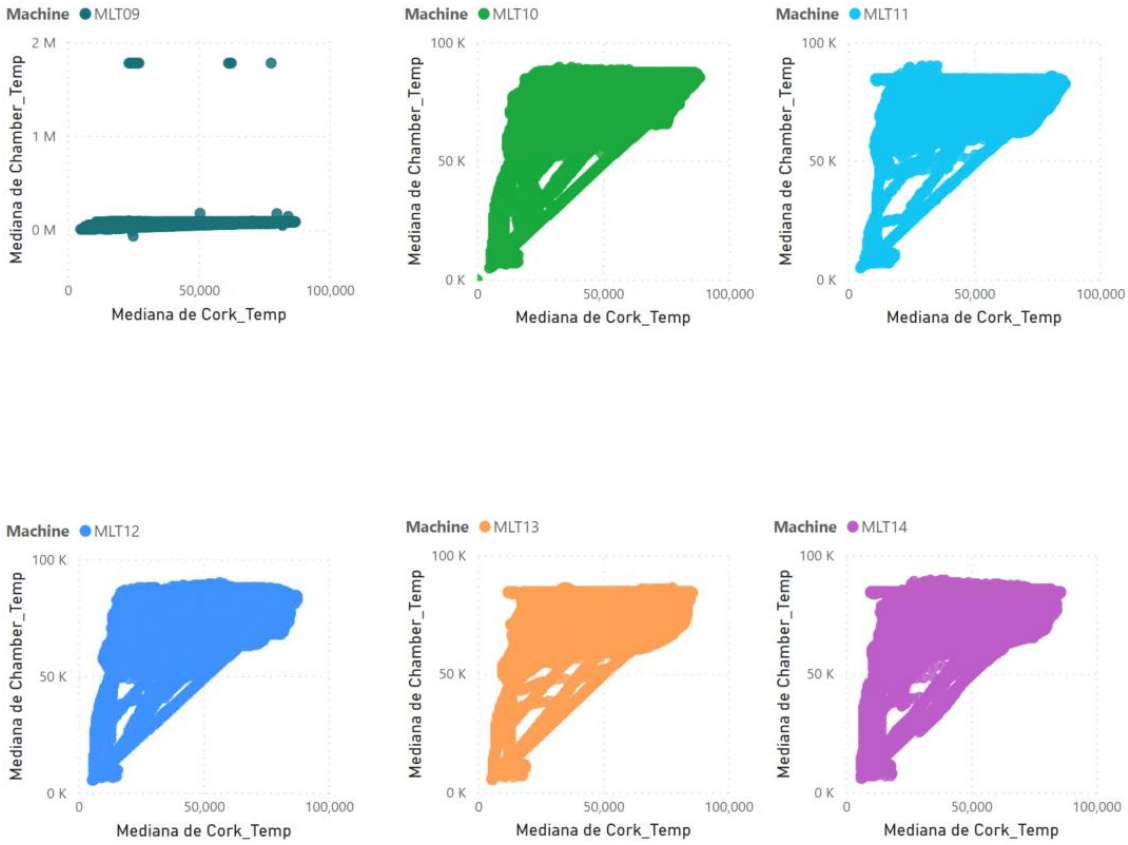


Figura 54 - Comparação de Temperaturas por máquina (MLT09 a MLT14) [3]

Uma outra análise que foi considerada foi a filtragem dos gráficos anteriores para perceber qual o padrão de relação de temperaturas na presença de uma anomalia (falha ou paragem). Foram então construídos dois conjuntos de gráficos que representam a relação nesta situação, representados na Figura 55 e Figura 56. Através da Figura 55 é percetível que o padrão de dispersão de temperaturas na presença de uma falha é diferente do padrão normal. Contudo, cada máquina possui o seu próprio padrão, não existindo um único que caracterize a falha.

Figura 55 - Dispersão de Temperaturas na presença de Falha[3]

Analisando a Figura 56, percebe-se que o padrão de dispersão de temperaturas é diferente de máquina para máquina. Sendo que este é também diferente face ao estado normal.

Ao analisar todos os gráficos da Figura 55 e da Figura 56, e tendo em conta os gráficos da Figura 53 e da Figura 54, é possível perceber que, em caso de anomalia, o padrão de correlação das temperaturas é completamente diferente de um estado normal. Comparando agora a correlação na presença de uma paragem com a presença de uma falha, o padrão também é diferente.

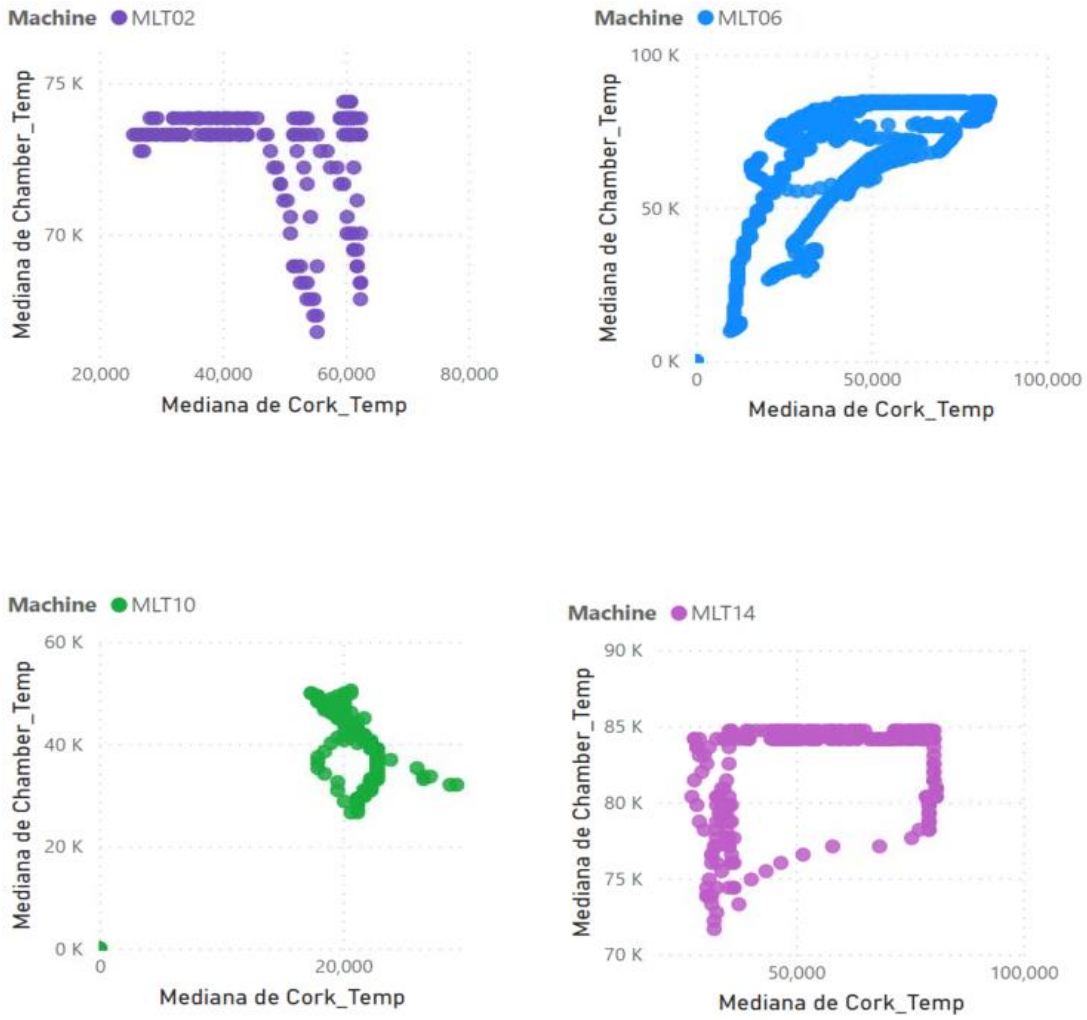


Figura 56 - Dispersão de Temperaturas na presença de Paragem[3]

Após a análise efetuada às temperaturas, as variáveis de pressão foram alvo de uma exploração semelhante. Isto é, foram construídos os gráficos presentes na Figura 57 e na Figura 58 para analisar a dispersão entre as medianas da pressão da bomba e da pressão da câmara. Foi novamente tomada a decisão de efetuar uma análise comparativa por máquina, dado o elevado ruido provocado pela presença das diversas máquinas no mesmo gráfico.

Na Figura 57 e Figura 58 podem observar-se os padrões de relação entre as variáveis de pressão nas diferentes máquinas. Desta comparação pode concluir-se que embora cada máquina tenha um padrão muito próprio, existe um aspeto que é semelhante, a tendência decrescente, isto é, quanto menor a pressão da bomba, menor a pressão na câmara.

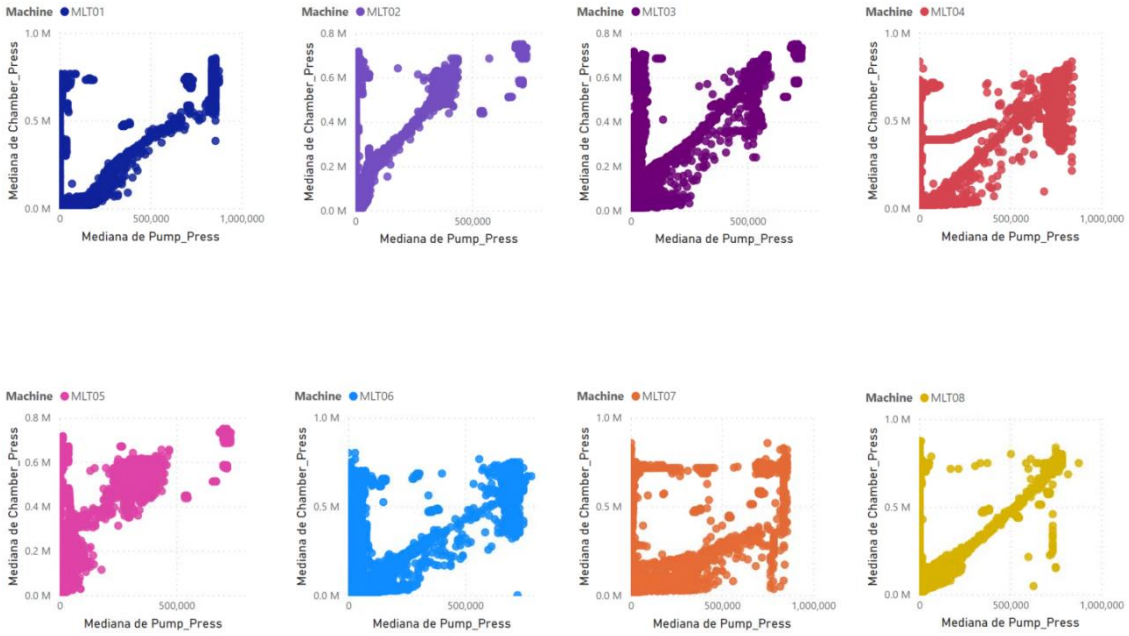


Figura 57 - Comparação de Pressões por máquina (MLT01 a MLT08)[3]

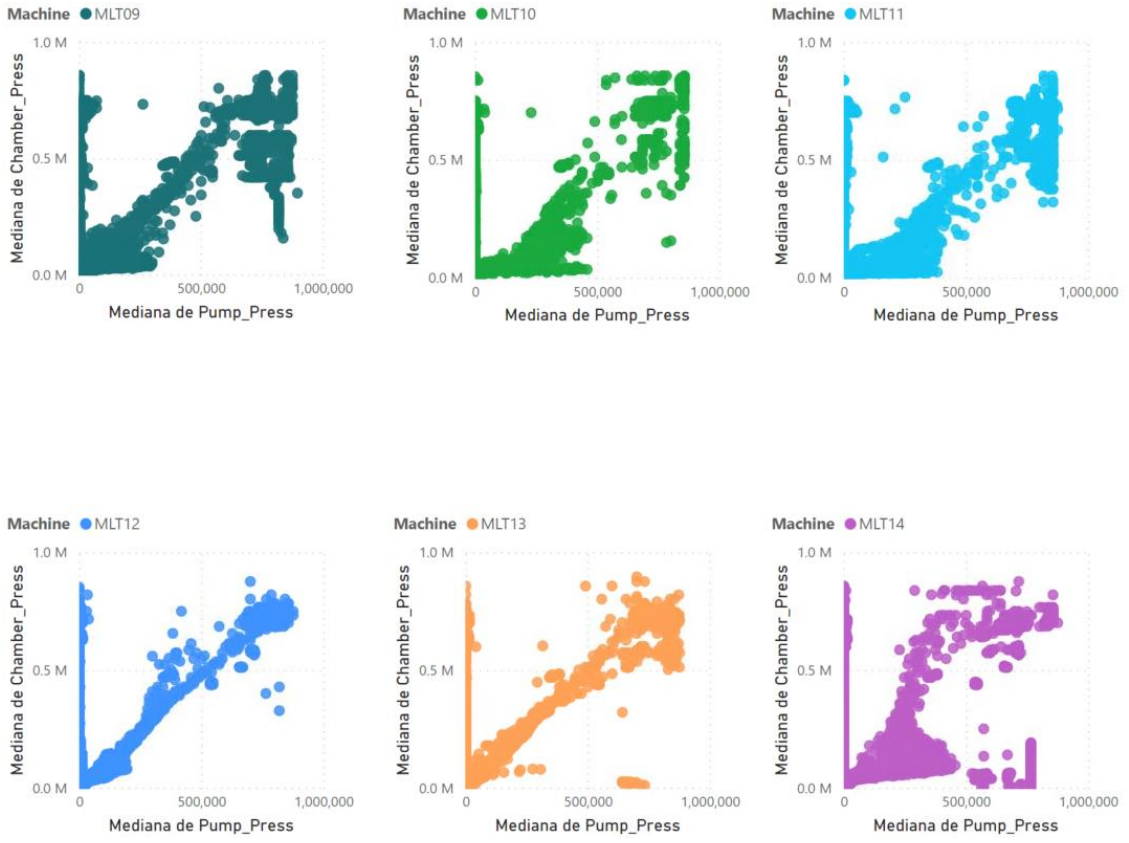


Figura 58 - Comparação de Pressões por máquina (MLT09 a MLT14)[3]

Depois de perceber que existem algumas semelhanças na relação entre as pressões, foram construídos gráficos onde é possível analisar esta relação em caso de falha e em caso de paragem, ilustrados na Figura 59 e Figura 60, respetivamente. Esta análise permite perceber

que os padrões são distintos do funcionamento normal das máquinas. Dentro das anomalias, os padrões de dispersão de pressões na presença de falha e paragem são também distintos, identificando um possível padrão de anomalia. Um aspeto pertinente identificado é que, no caso de falha na máquina MLT06, a temperatura e pressão da câmara mantêm-se constantes à medida que a temperatura da cortiça e pressão da bomba aumentam.

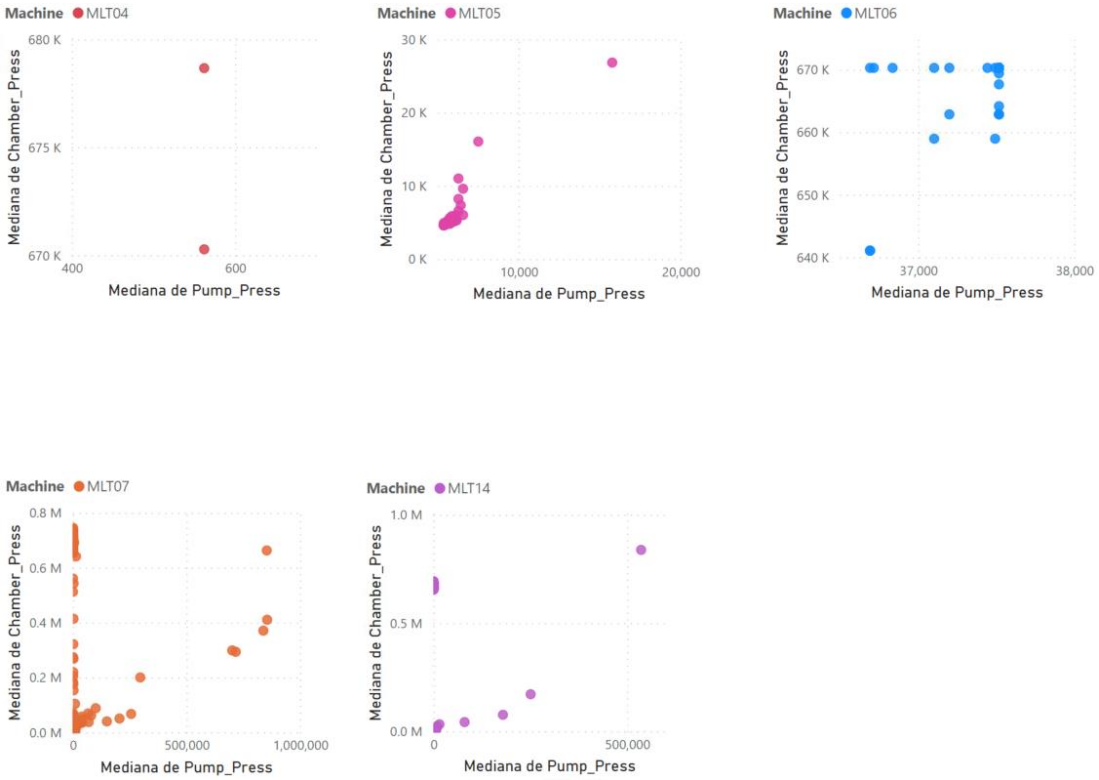


Figura 59 – Dispersão de Pressões na presença de Falha[3]

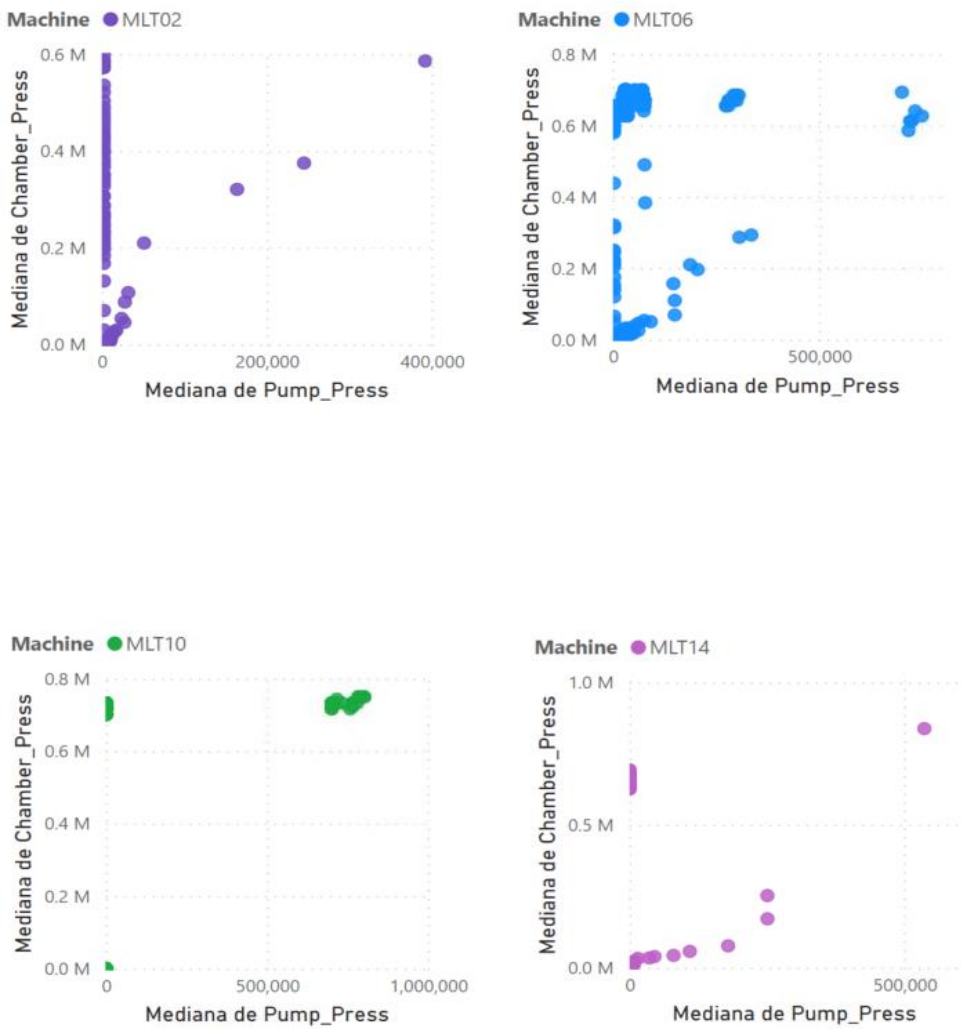


Figura 60 - Dispersão de Pressões na presença de Paragem[3]

Por fim, foi realizada uma análise das principais variáveis que influenciam numa anomalia nas máquinas MLT, através de uma correlação de *Pearson*. Na Figura 61 e Figura 62 encontram-se representados os principais influenciadores de falha e paragem respetivamente.

Em relação à análise no caso de falha, a variável que tem maior peso de influência sobre a a sua ocorrência é, claramente, a temperatura da câmara e as demais variáveis calculadas a partir desta. A variância e desvio padrão da temperatura são as variáveis com maior peso, com uma percentagem de 87,69%, ao passo que a média, mediana, mínimo e máximo da temperatura da câmara possuem um peso entre os 13,21% e 8,68%.

Figura 61 - Principais Influenciadores na Falha[4]

No que diz respeito à análise de paragem da máquina, através da análise da Figura 62 é possível perceber que existe uma forte relação entre as variáveis de pressão e uma paragem. Todas as variáveis de pressão e as suas derivadas possuem uma influência sobre a paragem da máquina acima dos 100%, entre 151,34% e 140,14%.

[4] Por motivos de confidencialidade, os dados foram ocultados.

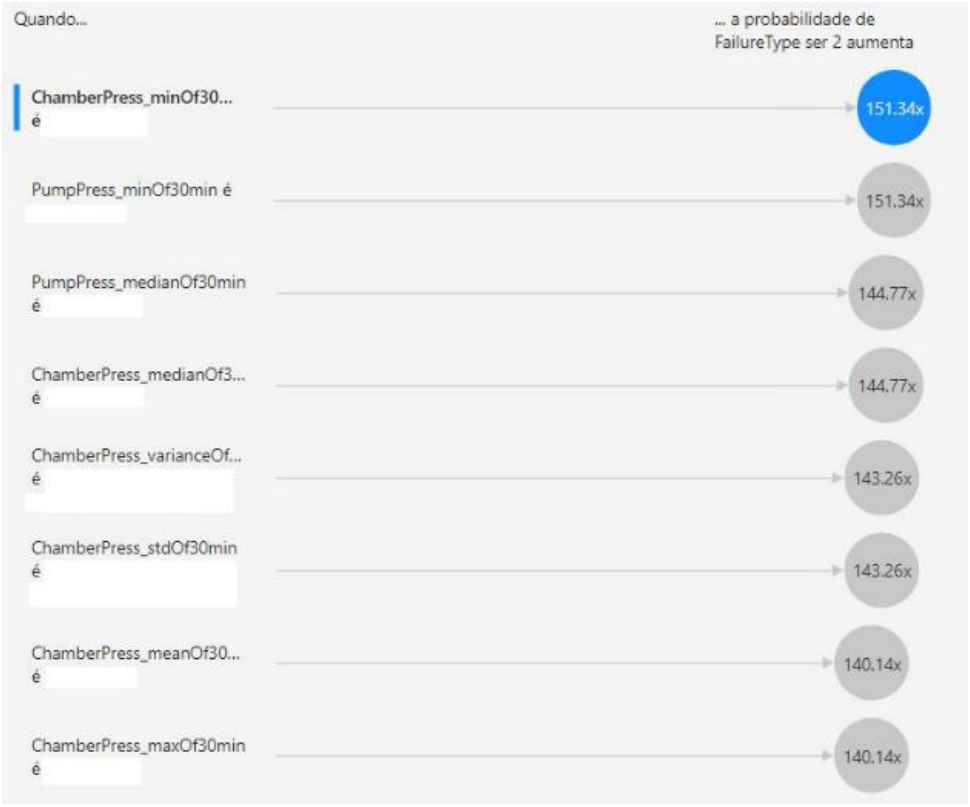


Figura 62 - Principais Influenciadores na Paragem[4]

Após efetuar a exploração de dados explanada nesta secção e tendo em conta a caracterização do *dataset* é possível concluir que:

- Existem *outliers*;
- Existem 15 anomalias, 8 paragens e 7 falhas;
- As paragens possuem maior duração do que as falha;
- Existiram paragens em 4 máquinas e existiram falhas em 5 máquinas;
- Existem padrões e relações entre variáveis;
- Existem padrões no funcionamento normal das máquinas, bem como nas anomalias;
- Cada máquina tem o seu próprio padrão de anomalia;
- As variáveis de pressão e temperatura têm influência nas anomalias;
- Existem variáveis com diferentes escalas.

5.2 Preparação dos Dados

Como constatado na secção 5.1.2.1, o *dataset* construído possui alguns defeitos no que concerne à qualidade dos dados, como por exemplos valores nulos, registos de erros, entre outros. Sendo assim, torna-se imperativo realizar algumas tarefas de preparação de dados. Na presente secção serão detalhadas as tarefas de preparação de dados realizadas, no sentido de melhorar a qualidade do *dataset*.

5.2.1 Limpeza e Tratamento de Dados

Nas várias variáveis que compõem o *dataset* foram encontradas diversas que serviram de apoio à construção da classe, outras que não eram pertinentes para projeto e ainda outras que necessitavam de limpeza.

Iniciando pela construção da classe, foram utilizadas as colunas *FailureID* e *Stop*. Primeiramente, foi aplicada uma transformação à coluna *FailureID*, representada no Excerto de Código 2. Esta permite que, quando o valor de *FailureID* é nulo, uma coluna adicional denominada de *Failure* fique com o valor booleano *false* e vice-versa. Desta forma, existe a clara noção da existência de falha.

```
CASE WHEN FailureID is null THEN false ELSE true END Failure
```

Excerto de Código 2 - Construção da variável *Failure*

Estando a variável auxiliar *Failure* construída, esta foi utilizada em conjunto com a variável Stop para desenvolver a classe, denominada de *FailureType*, representada no Excerto de Código 3. Esta é a classe que caracteriza a anomalia e contém 3 rótulos possíveis 0,1 ou 2, sendo que: 1) Rótulo 0 refere-se a "Sem Falha"; 2) Rótulo 1 refere-se a "Falha sem Paragem"; 3) Rótulo 2 refere-se a "Falha com Paragem".

```
CASE WHEN Failure=0 and Stop =0 THEN 0 WHEN Failure=1 and Stop =0 THEN 1 WHEN Failure=1 and
Stop =1 THEN 2 ELSE 3 END as FailureType
```

Excerto de Código 3 - Construção da variável *FailureType*

Posto isto, foi realizada uma análise de pertinência das demais variáveis para o problema. Sobre esta análise foi concluído o seguinte:

- *FailureID* – dado que apenas se refere a um ID sequencial de falhas, que não caracteriza uma falha e que foi utilizada como auxiliar para construir a classe, a presente variável foi excluída do *dataset*;
- *Description* – uma vez que se trata de uma descrição não normalizada e inserida manualmente pelo operador da máquina, não é representativa da falha. Sendo assim, esta variável foi excluída do *dataset*;

- *StartFailure* e *EndFailure* – ambas as variáveis referem-se ao início e fim da falha, tendo sido extremamente úteis na construção do *dataset*, dado que foram a chave do *join*. No entanto, a dimensão temporal dos dados é dada pela coluna *TimeStamp* e por este motivo estas variáveis foram excluídas do *dataset*;
- *Stop* – esta variável, que representa o nível de severidade da falha, foi de extrema relevância para a construção da classe. Uma vez construída, a presente variável deixa de ter importância no *dataset* e por este motivo foi excluída do mesmo;

No que diz respeito à existência de valores de temperatura e pressão nulos, após uma análise dos dados, foi percebido que estes valores apenas existiam aquando da presença de uma paragem. Após uma conversa com o cliente, este confirmou o comportamento observado. Assim sendo, e para não correr riscos de modificar valores erradamente, foi tomada a decisão de aplicar uma transformação ao *dataset* através do *script* SQL presente no Excerto de Código 4. Este *script* permite que quando os valores de temperatura e pressão são nulos e existe uma paragem estes passem ao valor 0.

Aquando da caracterização do *dataset* foram adicionalmente identificadas duas colunas com valores nulos ou vazios, as colunas *Cycle* e *Step*. Dado que estas são do tipo *string*, foi criado um novo valor para categorizar estes casos, ou seja, a categoria "*Unknown*".

```
select
Case When Cycle is null THEN 'Unknown' When Cycle = '' THEN 'Unknown' ELSE Cycle END as Cy-
cle,
Case When Step is null THEN 'Unknown' ELSE Step END as Step,
Machine, Shift,
CASE WHEN CorkTemp is null and Stop =1 THEN 0 ELSE CorkTemp END as CorkTemp,
CASE WHEN ChamberTemp is null and Stop =1 THEN 0 ELSE ChamberTemp END as ChamberTemp,
CASE WHEN ChamberPress is null and Stop =1 THEN 0 ELSE ChamberPress END as ChamberPress,
CASE WHEN PumpPress is null and Stop =1 THEN 0 ELSE PumpPress END as PumpPress,
TimeStamp,
FailureType
from dataset
```

Excerto de Código 4 - *Script* SQL de Limpeza de Valores Nulos

Para que um algoritmo tenha um bom desempenho preditivo, convém que os tipos de dados do *dataset* estejam corretamente definidos. Assim, após a limpeza de valores nulos é importante alinhar e editar os tipos de dados. Ou seja, todas as variáveis foram catalogadas com os tipos de dados corretos, inteiro, *double*, *time-stamp, string*, categórico, não categórico, *label* ou *feature*. Esta apesar de ser uma operação relativamente simples, é de extrema relevância. Por exemplo, se a variável de temperatura estivesse classificada como *string* podia não ser interpretada e utilizada por um modelo da mesma forma do que se estivessem corretamente classificados como inteiro.

5.2.2 Filtragem de Dados

O processo de limpeza de TCA através das MLT foi implementado pelo cliente em julho de 2020. Com isto, é expectável que exista uma curva de aprendizagem e que os primeiros meses de arranque do processo de limpeza se traduzam em erros e afinações das máquinas, do sistema SCADA e do processo em si. A isto, acresce o facto de no total das 18 máquinas existentes, apenas 14 terem sido instaladas desde essa data. As restantes 4 máquinas apenas foram instaladas em abril de 2021.

Analisando o *dataset* construído percebe-se que existem ao total 15 anomalias e que estas ocorreram entre o mês de janeiro e maio de 2021. Como dito anteriormente, os dados do processo das máquinas MLT estão compreendidos no intervalo temporal entre julho de 2020 e maio de 2021. Quer isto dizer que o *dataset* construído encontra-se desbalanceado, uma vez que existe um imenso volume de dados do processo contra um volume mais reduzido de anomalias.

Posto isto, para minimizar o problema do desbalanceamento do *dataset* e tendo em conta a curva de aprendizagem e a falta de volume de dados para 4 máquinas, foi tomada a decisão de filtrar o *dataset*. Este exclui as máquinas instaladas tardiamente e considera apenas dados a partir de 1 de dezembro de 2020 às 00:00H. Assim, como resultado final o *dataset* encontra-se compreendido no espaço temporal entre 01-12-2020 00:00:00 e 25-05-2021 13:02:00, para as máquinas MLT01, MLT02, MLT03, MLT04, MLT05, MLT06, MLT07, MLT08, MLT09, MLT10, MLT11, MLT12, MLT13, MLT14.

5.2.3 Substituição de Outliers

Durante a exploração de dados, na secção 5.1.2.2, uma das conclusões retiradas foi a existência de *outliers* nos valores de temperatura e pressão. A primeira impressão da equipa de trabalho é que os *outliers* identificados poderiam representar uma anomalia. Contudo, estes ocorreram num *timestamp* no qual não existiu qualquer registo de anomalia nas diversas fontes de dados. Sendo assim, foi tomada a decisão de conversar com o cliente no sentido de perceber se os valores detetados eram habituais. Durante a sessão ficou esclarecido que se tratavam de valores que não traduziam a realidade do funcionamento normal das máquinas. Adicionalmente, o cliente conferenciou quais os valores mínimos e máximos considerados expectáveis e aceitáveis para as variáveis de temperatura e pressão.

Findada a sessão com o cliente, os resultados da mesma foram aproveitados para realizar uma operação de substituição de *outliers*. Isto porque, estes se traduziam em ruido, podendo influenciar a performance do modelo e influenciar negativamente a precisão do mesmo [128].

Assim, foi utilizado um componente do Azure ML específico para o efeito, denominado de “Clip Values” [129]. Este componente recebe por parâmetro:

- As variáveis sobre as quais será aplicada a transformação;

- O tipo de limite, sendo que este pode ser uma constante ou um intervalo de percentil;
- Os limites constantes ou intervalos de percentil máximos e mínimos (consoante a configuração do tipo de limite);
- O método de substituição, sendo que este pode ser um valor específico, a média da variável, a mediana da variável ou um valor vazio.

Desta forma, este componente foi configurado como ilustrado na Figura 63. Ou seja, os valores indicados pelo cliente como mínimos e máximos foram utilizados para configurar os limites do componente "*Clip Values*". O método de substituição utilizado foi a mediana das variáveis de temperatura e pressão, sendo que esta é calculada antes da operação de substituição. A decisão recaiu sobre a mediana por ser mais resistente à presença de *outliers*.

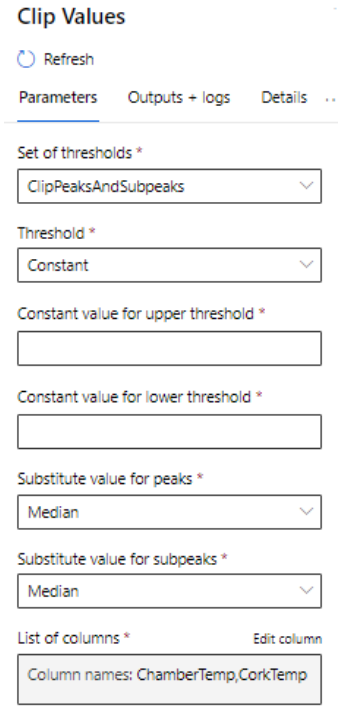


Figura 63 - Configuração de Substituição de *Outliers*[5]

5.2.4 Normalização

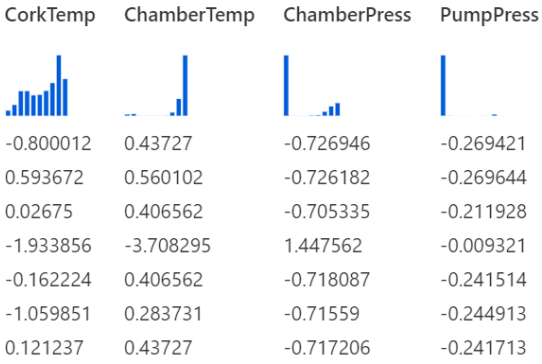
Aquando da caracterização do *dataset* e exploração do mesmo, foram detetadas diferentes escalas para as variáveis existentes. Por exemplo, as variáveis de temperatura possuem uma escala na ordem dos milhares, ao passo que, as variáveis de pressão possuem uma escala na ordem dos milhões. Assim sendo, foi realizada uma operação de normalização de todas as variáveis de pressão e temperatura para uma escala comum sem distorcer as diferenças nos intervalos de valores, utilizando a técnica *Z-Score*.

[5] Por motivos de confidencialidade, os valores mínimos e máximos foram ocultados.

A técnica de normalização *Z-Score*, recorre à fórmula indicada abaixo para quantificar o quão distante o desvio padrão de um dado registo está da média da população geral [130].

$$Z_i = \frac{d - mean(P)}{std(P)}, \qquad onde\ P = população, d = registo$$

Após a aplicação da normalização *Z-Score*, as variáveis de pressão e temperatura ficaram como exemplificado na Figura 64. Através desta, é possível perceber que ficaram todas na mesma escala e com valores positivos e negativos. Ou seja, apesar de o seu valor ter sido alterado ainda existe evidência dos seus intervalos de diferença.



CorkTemp	ChamberTemp	ChamberPress	PumpPress
-0.800012	0.43727	-0.726946	-0.269421
0.593672	0.560102	-0.726182	-0.269644
0.02675	0.406562	-0.705335	-0.211928
-1.933856	-3.708295	1.447562	-0.009321
-0.162224	0.406562	-0.718087	-0.241514
-1.059851	0.283731	-0.71559	-0.244913
0.121237	0.43727	-0.717206	-0.241713

Figura 64 - Resultado na Normalização *Z-Score*

5.2.5 Antecipação de Falhas

Na secção 5.2.1 encontra-se demonstrada a criação da classe *FailureType* com 3 rótulos possíveis, "Sem Falha", "Falha sem Paragem" e "Falha com Paragem". No entanto, após reunião com o cliente, foi indicado que as previsões de anomalias deveriam acontecer com uma antecedência de no mínimo 1 hora. Assim, para dar resposta a este requisito foi criada uma nova variável, denominada de *FailureType_Early_1hour.* Esta, através da variável *FailureType*, antecipa o estado de falha ou paragem 1 hora antes.

No Excerto de Código 5 encontra-se representada a lógica subjacente à construção desta variável. Sabendo que os registos se encontram numa cadência de minuto a minuto, tendo o *dataset* ordenado por data e tendo a garantia que não existiam datas em falta, foi realizado uma antecipação da anomalia através da contagem 61 linhas, agrupado por máquina e tipo de falha.

```
dataframe1['FailureType_Early_1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'],asce
nding=False).groupby('Machine')['FailureType'].rolling(61, min_periods = 1).max().reset_in
dex(drop=True, level=0)
```

Excerto de Código 5 - *Script Python* para Antecipação de Anomalias

Uma vez criada a nova variável, esta passou a ser considerada a classe e a variável *FailureType*, construída anteriormente, foi eliminada do *dataset*.

5.2.6 Derivação de Novas Features

Uma vez criada uma nova classe que antecipa a anomalia, torna-se pertinente a derivação de novas *features* que respeitem o intervalo de tempo de antecipação. Desta forma, foram derivadas, a partir das variáveis de temperatura e pressão, 12 novas *features* para cada uma delas. Isto é, tendo por base as principais variáveis influenciadoras detetadas aquando da exploração do *dataset*, foram calculados os máximos, mínimos, médias, medianas, desvio padrão, e variância de cada variável de temperatura e pressão nos últimos 30 minutos e na última hora para cada registo.

O Excerto de Código 6 demonstra a criação de algumas destas *features* para a última hora, no entanto a lógica é idêntica à que foi utilizada na criação da classe *FailureType_Early_1hour*. Ou seja, sabendo que os registos se encontram numa cadência de minuto a minuto e tendo o *dataset* ordenado, foi realizado um agrupamento dos registos por máquina e variável, para derivar as novas *features*. Todas estas foram criadas de forma similar à apresentada no Excerto de Código 6.

```
#CorkTemp Analytical Features
dataframe1['CorkTemp_maxOf1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'],
ascending=True).groupby('Machine')['CorkTemp'].rolling(61, min_periods = 1).max().reset_index(d
rop=True, level=0)

dataframe1['CorkTemp_minOf1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'], ascending=Tr
ue).groupby('Machine')['CorkTemp'].rolling(61, min_periods = 1).min().reset_index(drop=True, le
vel=0)

dataframe1['CorkTemp_meanOf1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'], ascending=T
rue).groupby('Machine')['CorkTemp'].rolling(61, min_periods = 1).mean().reset_index(drop=True,
level=0)

dataframe1['CorkTemp_medianOf1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'], ascending
=True).groupby('Machine')['CorkTemp'].rolling(61, min_periods = 1).median().reset_index(drop=Tr
ue, level=0)

dataframe1['CorkTemp_stdOf1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'], ascending=Tr
ue).groupby('Machine')['CorkTemp'].rolling(61, min_periods = 1).std().reset_index(drop=True, le
vel=0)

dataframe1['CorkTemp_varianceOf1hour']=dataframe1.sort_values(['Machine', 'TimeStamp'], ascendi
ng=True).groupby('Machine')['CorkTemp'].rolling(61, min_periods = 1).var().reset_index(drop=Tru
e, level=0)
```

Excerto de Código 6 - *Script Python* para Derivação de Novas *Features (CorkTemp)*

Para além da derivação de *features* tendo em conta as variáveis de temperatura e pressão, a variável *Timestamp* foi também decomposta em 6 novas *features*. Isto porque, existem diversas

informações a extrair do *Timestamp* que podem descrever melhor uma anomalia, como por exemplo o dia da semana. Assim, foi extraído desta variável as *features* apresentadas no Excerto de Código 7.

```
#Calendar Features
dataframe1['Year'] = dataframe1['TimeStamp'].dt.year
dataframe1['Month'] = dataframe1['TimeStamp'].dt.month
dataframe1['Day'] = dataframe1['TimeStamp'].dt.day
dataframe1['Hour'] = dataframe1['TimeStamp'].dt.hour
dataframe1['Minute'] = dataframe1['TimeStamp'].dt.minute
dataframe1['DayOfWeek'] = dataframe1['TimeStamp'].dt.dayofweek
```

Excerto de Código 7 - *Script Python* para Derivação de Novas *Feature*s (*Timestamp*)

Em jeito de resumo, o *dataset* ficou então constituído por 63 *features*, a saber:

- *Features* de temperatura e pressão originais, que caracterizam a temperatura e pressão num dado instante: *CorkTemp, ChamberTemp, ChamberPress, PumpPress*;
- *Features* que caracterizam os valores máximos, mínimos, médias, medianas, desvio padrão, e variância da temperatura e pressão, nos últimos 30 minutos e última hora. Cada uma destas *features* possui uma nomenclatura muito própria, com o seguinte formato: **[**Variável**]_[**Tipo_de_Cálculo**]**Of**[**Grau_de_Antecipação**]**, por exemplo: *ChamberPress_maxOf1hour* ou *ChamberPress_maxOf30min*;
- *Features* derivadas do *Timestamp*: *Year, Month, Day, Hour, Minute, DayOfWeek*;
- *Features* originais do *dataset*: *Machine, Cycle, Shift, Step, Timestamp*.

5.2.7 Balanceamento do Dataset

Na fase de filtragem do *dataset*, na secção 5.2.2, já tinha sido levantado e minimizado o problema de desbalanceamento do *dataset*. No entanto, os filtros aplicados não resolveram esta problemática. Após essa transformação o *dataset* ficou com 4 261 110 registos, dos quais 4 224 897 (99,15%) representam o estado normal das máquinas, 5 425 (0,13%) representam o estado de falha e 30 788 (0,72%) o estado de paragem.

Para lidar com este problema era necessário realizar um *oversampling* ou *undersampling* aos dados, ou seja, reajustar a distribuição das diferentes classes. Assim, foi tomada a decisão de aplicar uma técnica de *oversampling* denominada de *Synthetic Minority Oversampling Technique* (SMOTE) ao *dataset* de treino. O SMOTE tem como objetivo aumentar o volume das classes minoritárias através da criação de novos registos para as mesmas. Estes são criados "por interpolação entre várias instâncias da classe minoritária que estão dentro de uma vizinhança definida" [131]. Na Figura 65 encontra-se representado um exemplo do funcionamento do SMOTE, no qual uma instância de classe minoritária x_i é selecionada como base para criar novos registos sintéticos. Os vários vizinhos mais próximos da dessa classe (x_1 a x_4) são

escolhidos. Finalmente, uma interpolação aleatória é realizada com o objetivo de serem criadas as novas instâncias r_1 a r_4 [131].

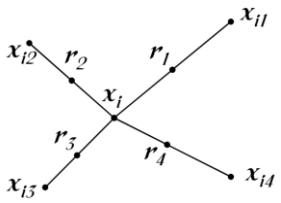


Figura 65 - Exemplo de funcionamento do SMOTE [131]

No Azure ML o componente de SMOTE recebe três parâmetros, a classe minoritária, o valor de *k* vizinhos e a quantidade de *oversampling*. Esta última é denominada de percentagem SMOTE, é expressa em múltiplos de 100 e refere-se à quantidade de novas instâncias geradas na vizinhança *k*, por cada registo.

Foram realizadas diversas execuções da técnica SMOTE. Ao longo destas, a percentagem SMOTE e o valor de *k* foram modificados até se atingir um resultado satisfatório no que ao balanceamento diz respeito. Na Tabela 17, encontram-se representados os resultados obtidos nas várias execuções.

Tabela 17 - Execuções Operação SMOTE

%SMOTE Falhas	*%SMOTE STOP*	*K Vizinhos*	*Nº Registos OK*	*Nº Registos Falha*	*Nº Registos STOP*	*RÁCIO OK*	*RÁCIO FALHA*	*RÁCIO STOP*
-	-	-	2 535 001	5 425	30 788	98,59%	0,21%	1,20%
2 600	1 600	4	2 535 001	146 475	523 396	79,10%	4,57%	16,33%
8 000	3 200	5	2 535 001	339 425	908 368	67,01%	8,97%	24,01%
25 000	8 000	8	2 535 001	861 675	1 693 828	49,80%	16,93%	33,27%
45 000	10 000	9	2 535 001	1 145 675	2 109 588	43,78%	19,79%	36,43%
55 000	13 000	10	2 535 001	1 810 035	2 407 780	37,54%	26,80%	35,56%

No final das diversas execuções do SMOTE foi concluído que a percentagem SMOTE para as falhas deveria ser de 55 000 e para as paragens 13 000, tendo sido estabelecido o valor de *k*=10. Após isto o *dataset* de treino ficou com um maior equilíbrio entre todas as classes.

5.2.8 Seleção de Features

Após a criação da nova classe e derivação de novas *features* foi percecionado um largo volume de informação, com um total de 63 *features*. No entanto, de acordo com Fernandes *et al* [132], este volume aumenta os requisitos computacionais da maioria dos algoritmos de ML, diminui o seu desempenho e aumenta o *overfitting*. Assim, torna-se necessário realizar uma seleção das *features* com maior capacidade preditiva. Para tal, foi utilizado um componente no Azure ML,

o "*Filter Based Feature Selection*", que utiliza o método de coeficiente de correlação *Pearson* para indicar quais as variáveis com maior capacidade preditiva sobre a classe *FailureType_Early_1Hour* [133]. Este método utiliza a equação indicada abaixo [133].

$$r(x,y) = \frac{\sum_{i=1}^{N_T}(x_i - \frac{1}{N_T}\sum_{j=1}^{N_T} x_j)(y_i - \frac{1}{N_T}\sum_{j=1}^{N_T} y_j)}{\sqrt{\sum_{i=1}^{N_T}(x_i - \frac{1}{N_T}\sum_{j=1}^{N_T} x_j)^2}\sqrt{\sum_{i=1}^{N_T}(y_i - \frac{1}{N_T}\sum_{j=1}^{N_T} y_j)^2}}$$

Sendo assim, foi realizada uma execução da pipeline de treino, representada na Tabela 18. Nesta execução o componente "*Filter Based Feature Selection*" foi configurado com o número máximo de *features* no *dataset* e com o método de correlação de *Pearson* para a pontuação das mesmas. Esta execução teve como objetivo aferir quais as *features* com maior capacidade preditiva para a classe, através do coeficiente de correlação *Pearson*.

Tabela 18 - Resultado da Seleção de *Features*

Feature	***FailureType_Early_1Hour Coeficiente Pearson***
ChamberTemp_maxOf1hour	0,333913323
ChamberTemp_maxOf30min	0,33078433
ChamberTemp_meanOf1hour	0,328149456
ChamberTemp_medianOf1hour	0,327993475
ChamberTemp_meanOf30min	0,327321394
ChamberTemp_medianOf30min	0,327301353
ChamberTemp	0,326766265
ChamberTemp_minOf30min	0,321451696
ChamberTemp_minOf1hour	0,316230133
Machine	0,271058555
CorkTemp_maxOf1hour	0,214344929
CorkTemp_maxOf30min	0,196804259
CorkTemp_meanOf1hour	0,187172596
Month	0,185301656
CorkTemp_meanOf30min	0,182083999
CorkTemp_medianOf1hour	0,180878357
CorkTemp_medianOf30min	0,17942855
CorkTemp	0,179038839
Step	0,164059473
CorkTemp_minOf30min	0,164047431
CorkTemp_minOf1hour	0,15281554
Day	0,058227074
CorkTemp_stdOf1hour	0,051085324
ChamberPress_maxOf1hour	0,040556772
CorkTemp_stdOf30min	0,040076162
ChamberTemp_stdOf1hour	0,039646659
ChamberTemp_stdOf30min	0,03602093
ChamberPress_maxOf30min	0,035861461
Year	0,033768432

ChamberPress_stdOf1hour	0,03331151
ChamberPress_meanOf1hour	0,030109424
ChamberPress_varianceOf1hour	0,028975242
PumpPress_maxOf1hour	0,028941433
ChamberPress_meanOf30min	0,028804917
ChamberPress_medianOf1hour	0,028001241
ChamberPress_medianOf30min	0,027768173
ChamberPress	0,027708318
CorkTemp_varianceOf1hour	0,025773547
PumpPress_maxOf30min	0,024435426
PumpPress_meanOf1hour	0,022064084
ChamberPress_stdOf30min	0,02203205
ChamberPress_minOf30min	0,021955425
PumpPress_meanOf30min	0,021858742
TimeStamp	0,021139493
PumpPress	0,021059356
PumpPress_medianOf1hour	0,020953344
PumpPress_medianOf30min	0,020936514
PumpPress_minOf30min	0,019850704
PumpPress_minOf1hour	0,019310866
PumpPress_stdOf1hour	0,019075028
ChamberPress_varianceOf30min	0,018355944
DayOfWeek	0,017726551
CorkTemp_varianceOf30min	0,017203103
ChamberPress_minOf1hour	0,015480778
ChamberTemp_varianceOf30min	0,014996424
PumpPress_stdOf30min	0,013301217
PumpPress_varianceOf1hour	0,011416819
PumpPress_varianceOf30min	0,00871964
Hour	0,00621046
Shift	0,00510775
ChamberTemp_varianceOf1hour	0,001823956
Minute	0,000311262
Cycle	0,000270986

Com base nos resultados obtidos, pode concluir-se que as *features* com maior coeficiente de correlação *Pearson* são maioritariamente as derivadas da temperatura da câmara. Uma outra conclusão é que a média dos resultados de correlação *Peason* para todas as *features* é de 0,098. Assim, foi tomada a decisão de selecionar as *features* com o coeficiente de correlação *Pearson* maior do que a média, por possuírem maior capacidade preditiva sobre a classe *FailureType_Early_1Hour*. No entanto a *feature month* apesar de possuir um coeficiente *Pearson* superior à média foi excluída. Isto porque, o *dataset* apenas contem metade dos meses do ano, podendo causar um elevado enviesamento no modelo e uma baixa capacidade de generalização para os restantes meses.

Após toda esta análise, foram selecionadas 20 *features*. Estas encontram-se presentes na seguinte lista: *ChamberTemp_maxOf1hour, ChamberTemp_maxOf30min, ChamberTemp_meanOf1hour, ChamberTemp_medianOf1hour, ChamberTemp_meanOf30min,*

ChamberTemp_medianOf30min, ChamberTemp, ChamberTemp_minOf30min, ChamberTemp_minOf1hour, Machine, CorkTemp_maxOf1hour, CorkTemp_maxOf30min, CorkTemp_meanOf1hour, CorkTemp_meanOf30min, CorkTemp_medianOf1hour, CorkTemp_medianOf30min, CorkTemp, Step, CorkTemp_minOf30min, CorkTemp_minOf1hour.

5.3 Pipelines e Orquestração de Dados

Na arquitetura abordada na secção 4.2.2 existe um componente que realiza toda a orquestração de dados, desde a extração de dados à notificação preditiva. O componente em questão é o *Azure Data Factory* e o desenvolvimento neste baseia-se em *pipelines*. Por outro lado, a codificação do modelo preditivo do presente serviço foi realizada através de uma *pipeline* de treino em *Azure ML*, que foi depois convertida numa *pipeline* preditiva. Assim, na presente secção serão apresentadas as *pipelines* de orquestração, de treino e a *pipeline* preditiva resultante.

5.3.1 Pipeline de Treino

Após a construção do *dataset*, as tarefas de preparação de dados abordadas na secção 5.2 foram realizadas numa *pipeline* de treino em Azure ML. Para melhor exposição e compreensão, esta pode-se dividir em 4 zonas de análise distintas: 1) Zona de Tratamento de Dados; 2) Zona de *SMOTE*; 3) Zona de Tipo de Dados e Seleção de *Features*; 4) Zona de Treino e Avaliação.

Na zona de tratamento de dados, presente na Figura 66, foram realizadas as operações de limpeza, filtragem, remoção de *outliers*, normalização, derivação de novas *features* e limpeza de registos com valores vazios. Estas correspondem ao abordado nas secções 5.2.2, 5.2.3, 5.2.4, 5.2.5, 5.2.6.

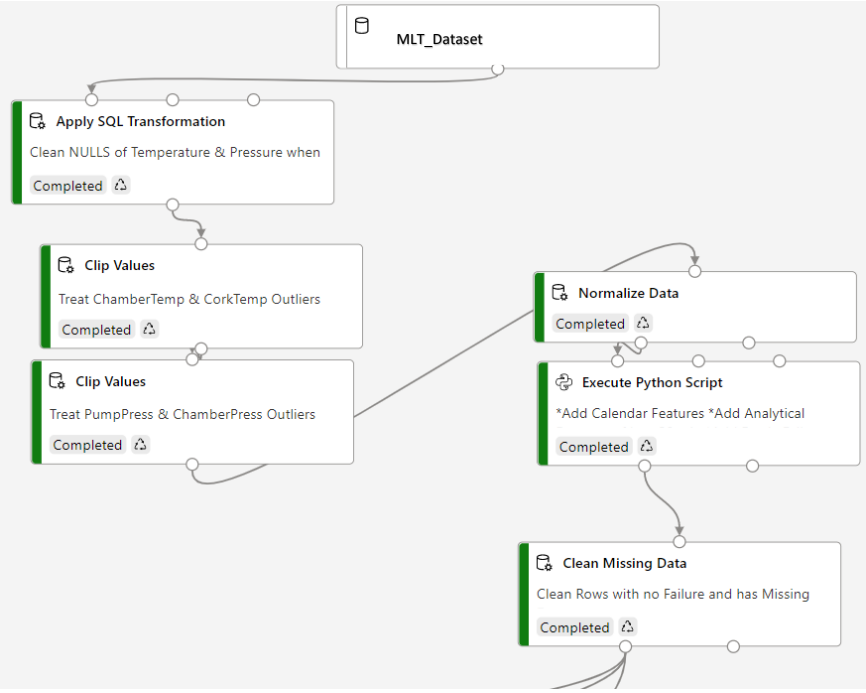


Figura 66 - Pipeline de Treino (Zona de Tratamento)

Na Figura 67 encontra-se representada a zona de *SMOTE* da *pipeline* de treino, onde é realizada a operação descrita na secção 5.2.7. Através desta é possível perceber que foram realizadas duas operações *SMOTE* e diversas separações de dados. Isto deve-se a uma limitação do *Azure ML* que apenas permite a operação *SMOTE* em classes binárias [134]. Assim sendo, o *dataset* foi separado de forma que o *SMOTE* receba uma de duas classes minoritárias e a classe maioritária. No final desta operação as classes minoritárias são isoladas para serem agregadas à classe maioritária e formar o *dataset* balanceado. Importa ainda salientar que, esta operação é apenas realizada para o *dataset* de treino, após a separação dos dados.

Figura 67 - *Pipeline* de Treino (Zona de *SMOTE*)

A zona de tipo de dados e seleção de *features*, ilustrada na Figura 68, é onde são configurados os tipos de dados das diversas *features* do modelo. Posto isto, é realizada uma seleção das diversas *features*. Esta zona corresponde à operação de tratamento de dados abordada na secção 5.2.7.

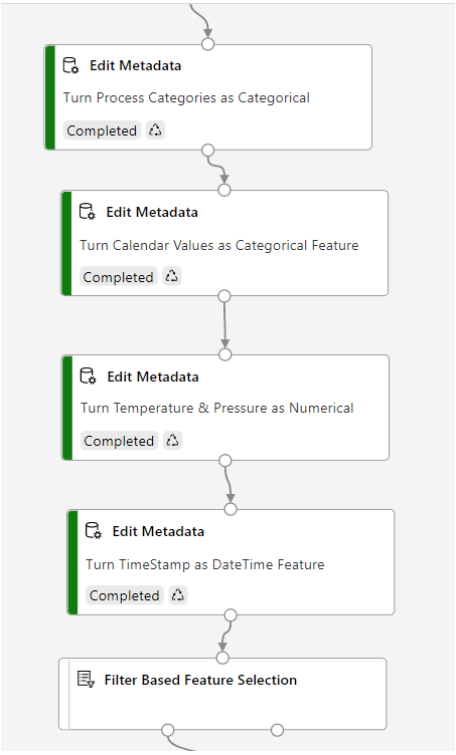


Figura 68 - *Pipeline* de Treino (Zona de Tipo de Dados e Seleção de *Features*)

Por fim, a última zona da presente *pipeline* é a de treino e avaliação, ilustrada na Figura 69. Esta reflete a estratégia de testes que será abordada na secção 6.2, bem como os testes realizados na secção 6.3.

Figura 69 - Pipeline de Treino (Zona de Treino e Avaliação com Hiperparametrização)

Ainda na zona de treino e avaliação existe uma outra variante. Isto é, como foram desenvolvidos diversos tipos de avaliações, essas refletem-se na pipeline de treino. Sendo assim, na Figura 70

encontra-se ilustrada a validação cruzada efetuada no âmbito do treino e avaliação do modelo. A versão final da pipeline de treino apenas inclui a variante selecionada na secção 6.3.4.

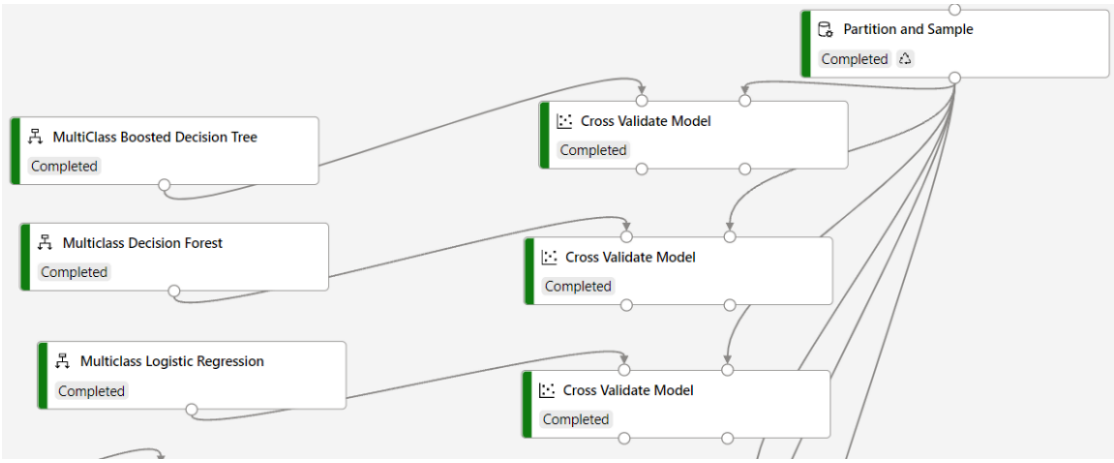


Figura 70 - Pipeline de Treino (Zona de Treino e Validação - Validação Cruzada)

5.3.2 Pipeline Preditiva

Após a implementação e consequente avaliação do modelo preditivo, abordada na secção 6.3, torna-se necessário efetuar o *deploy* do mesmo para consumo em tempo real. Assim, a *pipeline* de treino em *Azure ML* foi convertida numa *pipeline* preditiva, representada na Figura 71. Esta é em tudo semelhante à anterior, no entanto o *Azure M*L condensa algumas tarefas de treino e remove outras. Por exemplo, a tarefa de *SMOTE* é removida, uma vez que não se justifica aumentar o número de classes minoritárias numa pipeline de consumo em tempo real. Um outro exemplo é a normalização, que para aumentar a rapidez de resposta, esta é convertida num *join* entre um *dataset* pré-carregado com os valores de entrada na pipeline. Este *dataset* corresponde às *features* normalizadas durante o treino.

Todo o processo de implantação é realizado com base em configurações no *Azure ML*, sendo que a presente pipeline preditiva foi implantada num *Azure Container Instance* configurado com 40% de CPU e 60% de memória.

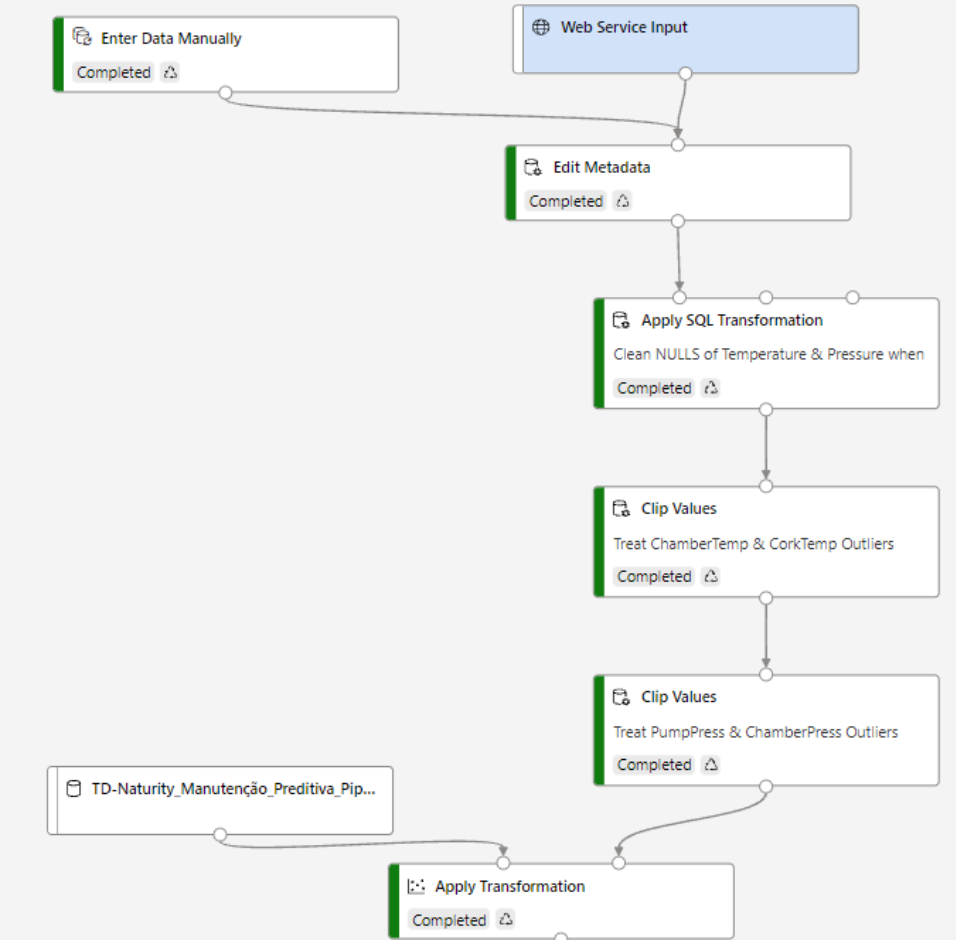


Figura 71 - Excerto *Pipeline* Preditiva em *Real Time*

Nesta pipeline existem ainda 2 componentes de entrada e saída adicionais, "*Web Service Input*" e "*Web Service Output*". O primeiro, refere-se ao *endpoint* REST de entrada na *pipeline* que recebe uma mensagem POST com um *json* idêntico ao apresentado no Excerto de Código 8.

```
{
   "Inputs":
        {
        "WebServiceInput0":
                [
                        {
                                "Cycle": "MLT06_01",
                                "Machine": "MLT06",
                                "Shift": "T2",
                                "Step": "Treatment",
                                "ChamberTemp": 170,
                                "CorkTemp": 180,
                                "PumpPress": 1800.45,
                                "ChamberPress": 1900.20,
                                "TimeStamp": "2021-09-06 04:55:00"
                        }
                ]
        }
}
```


Excerto de Código 8 - *Json* de Entrada na *Pipeline* Preditiva

Ou seja, a *pipeline* recebe uma lista de objetos, sendo que cada objeto representa os dados que se encontram na Tabela 19. O componente "*Web Service Output*" refere-se à resposta de saída da *pipeline* que indica a previsão da anomalia (falha ou paragem) para a próxima hora.

Tabela 19 - Caracterização da mensagem de entrada do modelo

Campo Mensagem	Significado	Tipo de Dados
Machine	MLT	*String*
Cycle	Ciclo de Tratamento	*String*
Shift	Turno de Trabalho	*String*
Step	Etapa de Tratamento	*String*
ChamberTemp	Temperatura da Câmara	*Integer*
CorkTemp	Temperatura da Cortiça	*Integer*
PumpPress	Pressão da Bomba	*Double*
ChamberPress	Pressão da Câmara	*Double*
TimeStamp	Data e hora de leitura	*Timestamp*

5.3.3 Orquestração Azure Data Factory

Para a orquestração de dados, no qual se inclui aquisição de dados da base de dados do SCADA, inserção dos dados em *Azure Data Lake*, envio de dados para o modelo ML e posterior notificação das previsões a potenciais *stakeholders*, foi desenvolvida uma pipeline em *Azure Data Factory*, ilustrada de forma macro na Figura 72. Esta é constituída por dois componentes,

um que efetua a extração de dados para o *Data Lake* e outro que contém a lógica de invocação do modelo preditivo. O primeiro não irá ser abordado por questões de confidencialidade.

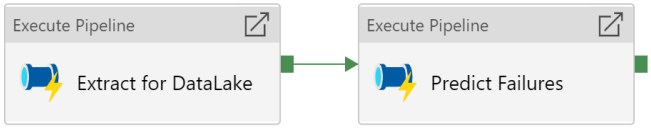


Figura 72 - *Pipeline* Macro *Azure Data Factory*

O componente preditivo da *pipeline*, ilustrado na Figura 73, recebe por parâmetro uma lista de máquinas e para cada uma destas é executado um outro componente que detém a lógica preditiva, como explanado na Figura 74.

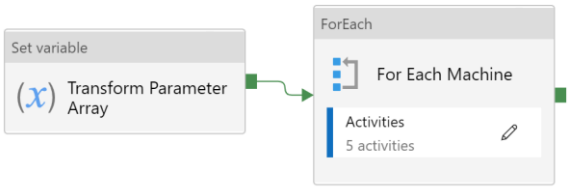


Figura 73 - Parametrização e Ciclo da *Pipeline* Preditiva

Por sua vez, este componente executa uma *query* ao *Data Lake,* através de um *notebook Azure Data Bricks.* Esta retorna as variáveis de temperatura e pressão das últimas duas horas, de uma dada máquina, num formato *json*. Após isto, é realizado um *parse* ao *json* para ficar de acordo com o *schema* que o modelo preditivo recebe. De seguida é enviada, para o modelo, uma mensagem *json* com os dados.

Figura 74 - Lógica da *Pipeline* Preditiva

A resposta recebida pelo modelo é enviada para uma *Azure Logic App* desenvolvida, representada na Figura 75. Esta interpreta o resultado recebido, para perceber se se trata de uma previsão de um estado normal, falha ou paragem e publica o resultado num *Microsoft Team* específico do cliente para alarmística de sistemas.

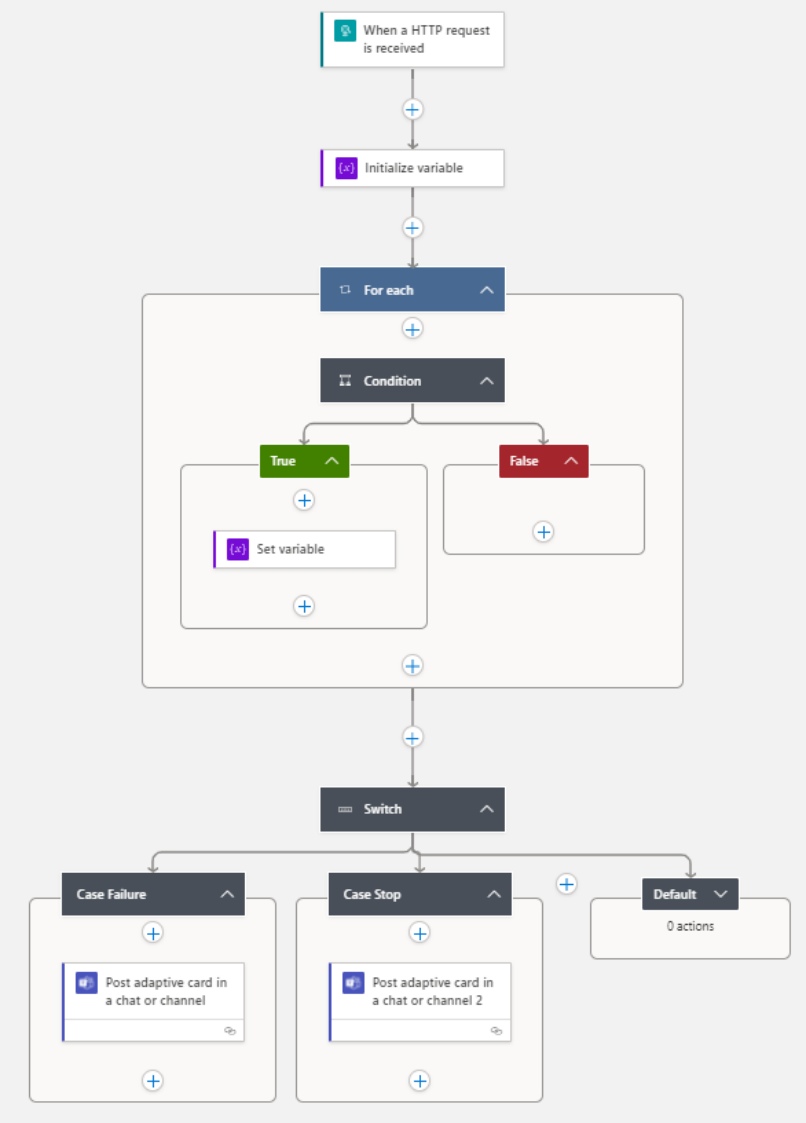


Figura 75 - *Azure Logic App* para Notificação

A utilização da *Logic App* prende-se com o facto de ter integração nativa com todos mecanismos de alarmística que o cliente utiliza e de ser altamente escalável, seja para *Microsoft Teams, e-mail, WhatsApp* ou até mesmo para o envio de uma mensagem para uma API. De momento apenas foi configurado o envio de notificações para o *Microsoft Teams*, representado na Figura 76, contudo no futuro poderão ser adicionados outros canais de alarmística.

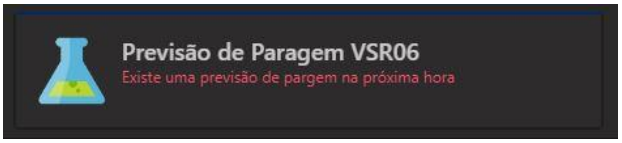


Figura 76 - Exemplo de Notificação de Falha

5.4 Testes

Os testes de software são uma disciplina do RUP para verificar se os resultados esperados correspondem aos resultados obtidos e assim garantir que o sistema esteja livre de defeitos. Esta disciplina é de elevada importância, uma vez que permite a identificação de erros e requisitos em falta. Erros estes que poderão acarretar elevados custos a nível monetário, uma vez que, a identificação dos mesmos num nível avançado do projeto e sem recurso aos testes poderá levar bastante tempo e consequentemente dinheiro [135].

Assim sendo, foram implementados três tipos de testes:

- Testes de validação de dados, que permitem atestar a qualidade e coesão dos mesmos;
- Testes funcionais, que permitem assegurar que as funcionalidades implementadas correspondem ao esperado;
- Testes de aceitação, que permitem ao cliente validar e atestar a qualidade do software em geral.

No presente projeto não existiu escrita de código numa linguagem específica, tendo sido utilizados diversos componentes com uma configuração muito própria e pontualmente escrita de código residual em *SQL Lite, Python* ou *SparkSQL*. Para além disso, todos os componentes utilizados não possibilitam o desenvolvimento de testes unitários. Sendo que também não faz sentido efetuar testes unitários aos componentes macro como *Data Lake, Data Factory*, *Data Bricks* ou *Logic Apps*, uma vez que são componentes *Platform as a Service (PaS)* desenvolvidos por outrem. Assim sendo, os testes unitários não foram considerados nos testes a realizar.

Para os testes de validação de dados foram aplicados os seguintes passos:

- Validação de todo o processo de integração ponta a ponta. Foi verificado se todo o processo de integração correu sem erros, desde extração para *Data Lake*, passagem para o modelo preditivo e notificação de predições;
- Validação da extração de dados. Foi verificada a passagem dos dados da base de dados original para o *Azure Data Lake*;
- Validação do volume, conteúdo e veracidade de dados presentes no *Data Lake* em confronto com os dados presentes na base de dados original. Isto é, foi verificado se o número de registos presentes em *Data Lake* coincidia com a base de dados original do SCADA, bem como, foram marcados alguns registos como exemplo, para se verificar após a integração se estes se mantinham coerentes;
- Validação dos dados enviados para o modelo, ou seja, foi averiguado se era possível, após a extração de dados, comunicar uma lista dos mesmos para o modelo preditivo, tentando cobrir todos os casos de exceção;
- Validação da resposta retornada pelo modelo e consequente envio para a *Logic App*, para aferir se os formatos e formatações da mesma se encontravam corretos e para

cobrir todos as potenciais exceções. Isto é, *time-outs* na resposta, resposta vazia e falha momentânea na autenticação.

Os testes funcionais avaliam se o desempenho do serviço se encontra em conformidade com os requisitos funcionais. Desta forma, para realizar os testes funcionais, os resultados esperados de cada caso de uso foram confrontados com os resultados obtidos. Assim, é possível perceber se a implementação passou nos testes ou não. Analisando a Tabela 20, que contempla o resultado dos referentes testes, é possível afirmar que o serviço passou nos testes funcionais.

Tabela 20 - Testes Funcionais

Teste	Resultado Esperado	Resultado Obtido
UC1 – Recolher dados via *pull*	O serviço deverá ser capaz de autonomamente recolher dados da base de dados origem.	O serviço recolhe dados através da pipeline Azure *Data Factory*; **Passou.**
UC2 – Envio dados via *push*	O serviço deverá ser capaz de receber dados do processo, via *push*.	O serviço tem um *endpoint* que recebe dados para despoletar todo o processo preditivo; Passou.
UC3 – Armazenar dados em *Data Lake*	Cada conjunto de dados recebido, seja via *push* ou *pull*, deverá ser armazenado em *Data Lake*.	O serviço armazena todos os dados em *Data Lake*; **Passou.**
UC4 – Previsão de Anomalias	O serviço deverá prever falhar e paragens das máquinas MLT.	O serviço tem um componente preditivo que prevê falhas e paragens das máquinas MLT; Passou.
UC5 – Envio de notificações para terceiros	Por cada previsão de falhas ou paragens, o sistema deverá notificar sistemas terceiros interessados em receber as notificações.	O serviço envia notificações das previsões; **Passou.**
UC6 – Configurar sistemas terceiros	Deverá ser possível configurar diferentes sistemas terceiros para envio de notificações.	O sistema de notificações está desenvolvido sobre *Logic Apps*. Este componente é configurável e compatível com diversos componentes de notificação; **Passou.**

Depois de todos os testes realizados, o serviço foi ainda submetido a testes de aceitação. Foram agendadas reuniões com as partes interessadas do presente projeto, onde foram apontados e discutidos alguns pontos menos satisfatórios para o sucesso dos testes de aceitação. Após essas mesmas reuniões foram reajustados os pontos menos positivos e no final, o serviço passou nos testes de aceitação.

6 Experimentação e Avaliação

Após a realização de diversas operações de tratamento de dados, cujo objetivo passa por preparar os mesmos para um modelo preditivo, eis que chega a fase de treino, avaliação e seleção do mesmo. Assim, no presente capítulo serão abordadas as experiências realizadas com o intuito de treinar e avaliar inúmeros modelos e selecionar o que obteve melhor performance. A estratégia adotada passou por utilizar diversos algoritmos que o *Azure ML* dispunha, treinar o modelo, pontuá-lo e avaliá-lo de acordo com algumas estratégias, como ilustrada na Figura 77. Nesta secção serão abordados todos os algoritmos utilizados, as diferentes estratégias e os resultados obtidos em cada uma.

Na secção 5.1.1 foi efetuado um levantamento das fontes de dados. Posto isto, foi concebido um *dataset* mediante as fontes disponíveis, tendo este sido caracterizado e explorado na secção 5.1.2. Desta feita, e tendo como base o trabalho realizado nessas secções, é possível afirmar que a tipologia de aprendizagem que deverá ser escolhida é a aprendizagem supervisionada. Isto prende-se com o facto de, no *dataset*, existirem duas colunas *FailureID* e *Stop* que juntas apresentam evidências históricas de falhas e paragens.

O resultado preditivo alvo do presente serviço é a existência de anomalia ou não. Assim, a abordagem seguida e descrita na secção 5.2.1 passou por criar uma classe com 3 rótulos, sendo cada um deles a tradução direta da previsão do estado normal, falha ou paragem. Isto significa então que se trata de um problema de classificação multiclasse.

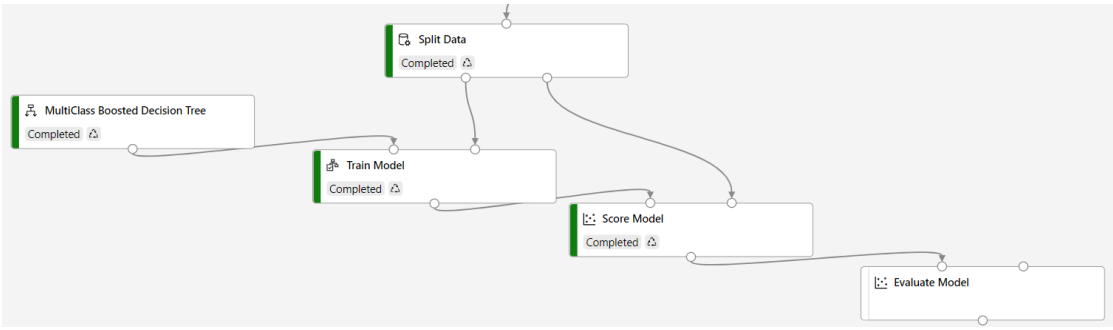


Figura 77 - Exemplo de Treino e Validação em Azure ML

6.1 Métricas de Avaliação

Os resultados preditivos do presente projeto apresentam um alto custo aquando da existência de falsos negativos. Ou seja, na existência de uma previsão de um falso negativo irá efetivamente ocorrer uma anomalia que o modelo não previu e consequentemente a equipa de manutenção não será alertada para o problema. Isto terá naturalmente muito mais impacto do que uma previsão de um falso positivo, em que a equipa de manutenção é alertada para algo que na realidade não irá acontecer.

Assim sendo, e tendo em conta que a métrica *recall* mede a fração de padrões positivos que são classificados corretamente, esta foi inicialmente eleita a principal métrica de avaliação [79]. No entanto, torna-se importante realizar um balanceamento entre *recall* e *precision*, isto porque, apesar do custo de um falso positivo ser menor, este também não é desejável. Então foi considerada também a métrica *F1-Score*, que é uma média harmónica entre o *recall* e a *precision* [136].

Contudo, na classificação multiclasse, para o cálculo da métrica *F1-Score* existe a necessidade de envolver todas as classes, sendo necessário calcular previamente uma métrica multiclasse do *recall* e da *precision*. Ou seja, é necessário calcular a média macro e micro *recall*, bem como a média macro e micro *precision*, para se chegar às métricas micro *F1-Score* e macro *F1-Score*. Na média macro a métrica é calculada de forma independente para cada classe sendo depois realizada a média, ou seja, todas as classes são tratadas de igual forma e todas possuem o mesmo peso. Já no cálculo de uma micro média são somados todos os verdadeiros positivos de todas as classes e dividido pela soma de todos os verdadeiros positivos com a soma de todos os falsos positivos. Ou seja, uma micro média agrega as contribuições de todas as classes para calcular a métrica [80]. Por fim, métricas micro e macro *F1-Score* são calculadas pelas fórmulas:

$$Micro\ F1-Score\ = 2\ * \frac{Micro\ Precision\ *\ Micro\ Recall}{Micro\ Precision\ +\ Micro\ Recall}$$

$$Macro\ F1-Score\ = 2\ * \frac{Macro\ Precision\ *\ Macro\ Recall}{Macro\ Precision\ +\ Macro\ Recall}$$

Assim sendo e tendo em conta que o presente *dataset* se encontra desbalanceado, foi tomada a decisão de avaliar a performance dos modelos preditivos através das médias macro. Isto porque, estas métricas atribuem o mesmo pesa a cada classe. Em suma, foram utilizadas as métricas macro *recall* e macro *F1-Score*.

6.2 Estratégia de Avaliação

A problemática do presente projeto é a classificação multiclasse, sendo utilizados maioritariamente algoritmos de classificação multiclasse. Contudo, para estes problemas também é possível utilizar algoritmos de classificação binária, recorrendo a estratégias de *One vs. All ou One vs. One*. Nestas estratégias, sendo *N* o número de rótulos de uma dada classe, existiriam *N* classificadores binários para a primeira e *N*(N-1)/2* classificadores binários na segunda estratégia. Tendo em consideração a classe *FailureType_Early_1Hour*, que possui 3 rótulos, em ambas as estratégias existiriam ao total 3 classificadores binários. Logo era indiferente optar por uma ou por outra, tendo sido escolhida a estratégia *One vs. All* [137].

Sendo assim, a estratégia de avaliação abordada passou por utilizar na *pipeline* de treino todos os algoritmos que o *Azure ML* dispunha para a problemática de classificação, sendo que os de classificação binária foram agregados à estratégia anteriormente indicada. A *pipeline* foi executada diversas vezes com diferentes ajustes e configurações com o objetivo de recolher métricas de performance. No final foram comparados os resultados obtidos em todos os testes

com o intuito de selecionar o melhor algoritmo e consequentemente obter o melhor modelo preditivo.

Os algoritmos de classificação multiclasse utilizados foram o *Multiclass Bosted Decision Tree, Multiclass Decision Forest* e *Multiclass Logistic Regression*, como representado na Figura 78.

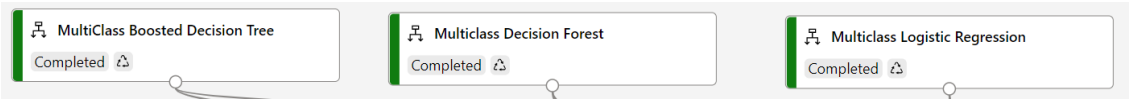


Figura 78 - Algoritmos de Classificação Multiclasse

No que diz respeito aos algoritmos de classificação binária utilizando a estratégia *One vs. All*, foram utilizados o *Two-Class Decision Forest, Two-Class Boosted Decision Forest* e o *Two-Class Support Vector Machine*, como ilustrado na Figura 79.

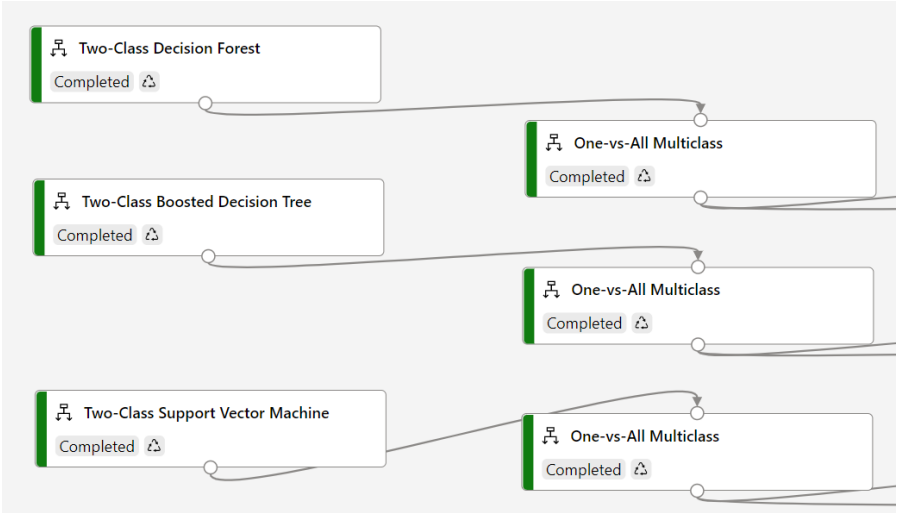


Figura 79 - Algoritmos de Classificação Binária

6.3 Testes Realizados

Depois de tomada a decisão sobre qual seria a estratégia de avaliação é necessário efetuar um conjunto de testes para efetivamente avaliar os diferentes algoritmos e selecionar o melhor. Para tal, a pipeline de treino, representada através de um excerto na Figura 80, foi executada diversas vezes com um tempo médio de execução de cerca de 20 horas. Cada corrida foi executada com todos os algoritmos abordados na secção 6.2 e com diferentes parametrizações para se aferir com qual, ou quais, parametrizações o modelo obtém melhor performance.

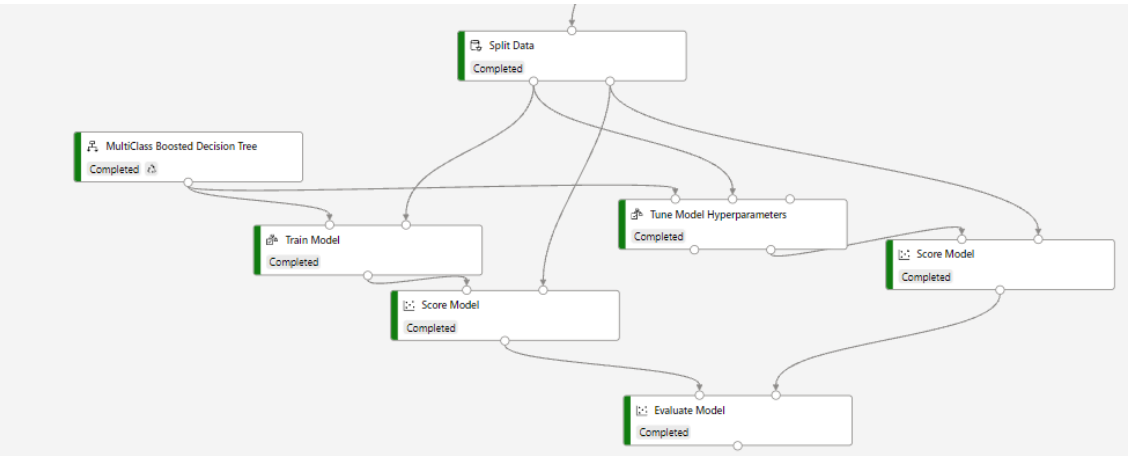


Figura 80 – Excerto da *Pipeline* de treino

Relativamente aos testes efetuados, estes assumem as seguintes tipologias:

- Modificação das condições de separação entre *dataset* de treino e teste;
- Modificação dos hiperparâmetros dos algoritmos;
- *Cross Validation Stratified K-Fold*.

Importa ainda salientar que, para os testes realizados no presente projeto foi aplicada uma operação de *oversampling* ao *dataset* de treino, através do SMOTE, como indicado na secção 5.2.7. Esta operação tem como objetivo combater o desbalanceamento observado na *dataset* e não foi aplicada nos testes de validação cruzada. Tal não foi possível, porque neste tipo de testes o componente de validação cruzada do Azure ML recebe o *dataset* inicial sem separação de treino e teste. Assim, por limitação do Azure ML, não foi possível aplicar SMOTE ao *dataset* de treino e como tal este tipo de testes utilizaram o *dataset* desbalanceado.

6.3.1 Condições de separação dataset treino e teste

Na presente tipologia de testes as condições de separação dos *datasets* de treino e teste foram alteradas para se obter os resultados de avaliação e comparar quais eram as condições mais favoráveis.

Num primeiro teste, a condição de separação foi 70% dos registos para o *dataset* de treino e 30% para o *dataset* de teste. Os resultados obtidos encontram-se representados na Tabela 21. Neste teste os 3 algoritmos com melhor performance são o *Two Class Decision Forest*, o *Multiclass Decision Forest* e o *Multiclass Boosted Decision Tree*, com um Macro *F1 Score* de 0,9999835, 0,999944501 e 0,999913003, respetivamente. Sendo assim, no presente teste o algoritmo escolhido seria o *Two Class Decision Forest*.

Tabela 21 - Resultados obtidos Separação *dataset* 70/30

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999995	0,999995	0,999994	0,999995	0,999995	0,999985	0,9999835
Multiclass Decision Forest	0,999983	0,999983	0,999979	0,999983	0,999984	0,999949001	0,999944501
Multiclass Boosted Decision Tree	0,999971	0,999971	0,999968	0,999971	0,999974	0,999913003	0,999913003
Two Class Boosted Decision Tree	0,999919	0,999919	0,999914	0,999919	0,999928	0,99975702	0,999763019
Multiclass Logistic Regression	0,985781	0,985781	0,985172	0,985781	0,987182	0,957946665	0,959099593
Two Class Support Vector Machine	0,981498	0,981498	0,980372	0,981498	0,983256	0,945514638	0,946424094

No segundo teste realizado, no que diz respeito às condições de separação do *dataset*, a condição foi 60% para o *dataset* de treino e 40% para o *dataset* de teste. Os resultados obtidos encontram-se espelhados na Tabela 22, tendo sido ligeiramente inferiores para todos os algoritmos face ao teste anterior. Os algoritmos com melhor performance foram o *Multiclass Boosted Decision Tree*, *Two Class Decision Forest* e o *Multiclass Decision Forest*, com um Macro *F1 Score* de 0,994376638, 0,990494295 e 0,989153368, respetivamente. Portanto neste teste o algoritmo selecionado com melhor performance seria o *Multiclass Boosted Decision Tree*.

Tabela 22 - Resultados obtidos Separação *dataset* 60/40

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Multiclass Decision Forest	0,999994	0,999994	0,999993	0,999994	0,999994	0,999982	0,9999805
Two Class Boosted Decision Tree	0,999993	0,999993	0,999992	0,999993	0,999994	0,999979	0,999979
Two Class Decision Forest	0,999942	0,999942	0,999938	0,999942	0,999948	0,99982601	0,99982901
Multiclass Logistic Regression	0,999922	0,999922	0,999915	0,999922	0,999931	0,999766018	0,999769018
Multiclass Boosted Decision Tree	0,985446	0,985446	0,984805	0,985446	0,986883	0,956970374	0,958128212

Two Class Support Vector Machine	0,981503	0,981503	0,980382	0,981503	0,983262	0,945529088	0,946447241

Adicionalmente, foi realizado um teste de condição de separação, tendo em conta dispersão temporal do *dataset* inicial. Quer isto dizer que se procedeu à análise do número de anomalias por mês, presente na Tabela 23, para se encontrar uma divisão do *dataset* tendo em conta estes dados. De notar que, o número de anomalias unitárias não reflete o número de registos, porque uma anomalia pode traduzir-se em vários registos ao longo do tempo, dependo da duração da mesma.

Tabela 23 - Número de Anomalias por mês

Mês	Nº Falhas	Nº Paragens
12	0	0
1	1	4
2	1	2
3	2	2
4	2	0
5	1	0

Assim, neste último teste realizado o *dataset* de treino contemplou os meses 12, 1, e 2, já o *dataset* de teste incluiu meses 3, 4 e 5. Os resultados obtidos encontram-se espelhados na Tabela 22, tendo sido bastante inferiores para todos os algoritmos face aos testes anteriores. Mesmo assim, os algoritmos com melhor performance foram o *Two Class Decision Forest*, *Multiclass Decision Forest* e o *Two Class Boosted Decision Tree,* com um Macro *F1 Score* de 0,122818276, 0,118532293 e 0,05368411, respetivamente. Portanto neste teste o algoritmo selecionado com melhor performance seria o *Two Class Decision Forest*.

Tabela 24 - Resultados obtidos Separação *dataset* por mês

Algoritmo	***Accuracy***	***Micro Precision***	***Macro Precision***	***Micro Recall***	***Macro Recall***	***Micro F1 Score***	***Macro F1 Score***
Two Class Decision Forest	0,997198	0,997198	0,643600	0,997198	0,391789	0,991617532	0,122818276
Multiclass Decision Forest	0,997173	0,997173	0,627279	0,997173	0,391780	0,991542953	0,118532293
Two Class Boosted Decision Tree	0,995257	0,995257	0,364512	0,995257	0,390563	0,985838381	0,05368411
Two Class Support Vector Machin	0,995496	0,995496	0,332336	0,995496	0,332826	0,986548767	0,036786785
Multiclass Logistic Regression	0,993094	0,993094	0,332334	0,993094	0,332023	0,979424749	0,036653409

Multiclass Boosted Decision Tree	0,983566	0,983566	0,332361	0,983566	0,329068	0,951503791	0,03616914

Para além de auxiliar no processo de seleção de algoritmos e modelos, este conjunto de testes permitiu perceber que a melhor condição de separação de *dataset* seria 70% para o *dataset* de treino e 30% para validação. Nos restantes testes esta foi a condição utilizada. Uma outra conclusão a reter destes testes é que o melhor algoritmo foi o *Two Class Decision Forest*, com um Macro *F1-Score* de 0,9999835.

6.3.2 Hiperparametrização

Na operação de hiperparametrização existem dois tipos de procura dos melhores parâmetros, *Random Sweep* e Entire Grid Sweep. No primeiro tipo são realizadas combinações aleatórias e os melhores resultados servem de guia para a seleção das próximas combinações. Este tipo de procura possui um menor esforço computacional, no entanto, pelo facto de os resultados servirem de guia para as próximas combinações, pode levar a mínimos locais e não a mínimos globais. Já no segundo tipo de procura são exploradas exaustivamente todas as combinações possíveis de hiperparâmetros, contudo este possui um maior esforço computacional e associado a isso um maior volume de tempo [138].

No presente projeto, como foi utilizado o Azure ML, tornou-se possível a alocação temporária de servidores dedicados com grande capacidade, sendo o esforço computacional considerado desprezível. Sendo assim, foi tomada a decisão de se optar pelo tipo de procura *Entire Grid Sweep*, para garantir os melhores hiperparâmetros possíveis.

O Azure ML possui um componente que efetua a afinação dos hiperparâmetros de forma automática. Para tal, basta que todos os algoritmos estejam configurados com um intervalo de valores para cada parâmetro. De seguida, encontram-se identificados os intervalos de valores para cada hiperparâmetro de cada algoritmo [139].

- *Multiclass Boosted Decision Tree*
    - Número Máximo de Folhas por Arvore – 2; 8; 32; 128
    - Número Mínimo de Amostras por Nó Folha – 1; 10; 50
    - Taxa de Aprendizagem – 0,025; 0,05; 0,1; 0,2; 0,4
    - Número de Árvores Construídas – 20; 100; 500
- *Multiclass Decision Forest*
    - Número de Árvores de Decisão – 1; 8; 32
    - Profundidade Máxima das Árvores de Decisão – 1; 16; 64
    - Número Mínimo de Amostras por Nó Folha – 1; 4; 16

- *Multiclass Logistic Regression*
    - Tolerância de otimização – 0,00001; 0,00000001
    - Peso de regularização L2 – 0,01; 0,1; 1,0
- *Two-Class Decision Forest*
    - Número de Árvores de Decisão – 1; 8; 32
    - Profundidade Máxima das Árvores de Decisão – 1; 16; 64
    - Número Mínimo de Amostras por Nó Folha – 1; 4; 16
- *Two Class Boosted Decision Tree*
    - Número Máximo de Folhas por Arvore – 2; 8; 32; 128
    - Número Mínimo de Amostras por Nó Folha – 1; 10; 50
    - Taxa de Aprendizagem – 0,025; 0,05; 0,1; 0,2; 0,4
    - Número de Árvores Construídas – 20; 100; 500
- *Two Class Support Vector Machine*
    - Número de iterações – 1; 10; 100
    - Lambda – 0,00001; 0,0001; 0,001; 0,01; 0,1

Após a configuração do tipo de procura e dos intervalos de valores dos hiperparâmetros, foi necessário decidir quais as métricas objetivo para a hiperparametrização. Como indicado anteriormente as métricas selecionadas para avaliar os modelos foram o Macro *Recall* e o Macro *F1 Score*. No entanto, foram realizados testes individuais cujas métricas configuradas para medir a performance no modelo, nos testes de hiperparametrização, foram o *F1-Score*, *Recall* e a área debaixo da curva ROC (AUC).

No teste de hiperparametrização em que a métrica objetivo foi o *F1-Score*, obtiveram-se os resultados descritos na Tabela 25. Através da análise da tabela, é possível perceber que os 3 melhores algoritmos são o T*wo Class Boosted Decision Tree, Multiclass Boosted Decision Tree e o Two Class Decision Forest,* tendo obtido um Macro *F1-Score* de 0,995645834, 0,995207638 e 0,988826368, respetivamente. Assim sendo, neste teste o algoritmo selecionado foi o *Two Class Boosted Decision Tree*.

Tabela 25 - Resultados obtidos Hiperparametrização (*F1-Score*)

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Boosted Decision Tree	0,999976	0,999976	0,998522	0,999976	0,998571	0,999928002	0,995645834
Multiclass Boosted Decision Tree	0,999975	0,999975	0,998538	0,999975	0,998262	0,999925002	0,995207638

Two Class Decision Forest	0,999955	0,999955	0,998198	0,999955	0,99433	0,999865006	0,988826368
Multiclass Decision Forest	0,999943	0,999943	0,997894	0,999943	0,99332	0,99982901	0,986868396
Multiclass Logistic Regression	0,960003	0,960003	0,435263	0,960003	0,977836	0,884744294	0,256385201
Two Class Support Vector Machin	0,954404	0,954404	0,42884	0,954404	0,974793	0,869354192	0,248995675

No teste de hiperparametrização em que a métrica objetivo foi o *Recall*, os resultados obtidos encontram-se representados na Tabela 26. Estes foram exatamente iguais aos obtidos no teste anterior e por esse motivo o algoritmo selecionado foi o mesmo, o *Two Class Boosted Decision Tree.*

Tabela 26 - Resultados obtidos Hiperparametrização (*Recall*)

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Boosted Decision Tree	0,999976	0,999976	0,998522	0,999976	0,998571	0,999928002	0,995645834
Multiclass Boosted Decision Tree	0,999975	0,999975	0,998538	0,999975	0,998262	0,999925002	0,995207638
Two Class Decision Forest	0,999955	0,999955	0,998198	0,999955	0,99433	0,999865006	0,988826368
Multiclass Decision Forest	0,999943	0,999943	0,997894	0,999943	0,99332	0,99982901	0,986868396
Multiclass Logistic Regression	0,960003	0,960003	0,435263	0,960003	0,977836	0,884744294	0,256385201
Two Class Support Vector Machin	0,954404	0,954404	0,42884	0,954404	0,974793	0,869354192	0,248995675

Por fim, foi realizado mais um teste de hiperparametrização. Neste, a métrica objetivo foi o AUC, e os resultados encontram-se na Tabela 27. Os algoritmos com melhor Macro *F1-Score* foram o *Multiclass Boosted Decision Tree, Two Class Boosted Decision Tree e o Multiclass Decision Forest,* com um respetivo valor de 0,99724703, 0,997055379 e 0,9969042. Assim sendo, neste teste o algoritmo selecionado foi novamente o *Multiclass Boosted Decision Tree.*

Tabela 27 - Resultados obtidos Hiperparametrização (AUC)

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Multiclass Boosted Decision Tree	0,999985	0,999985	0,999084	0,999985	0,999079	0,999955001	0,99724703
Two Class Boosted Decision Tree	0,999985	0,999985	0,998929	0,999985	0,999106	0,999955001	0,997055379
Multiclass Decision Forest	0,999913	0,999913	0,998967	0,998967	0,998967	0,998320493	0,9969042
Two Class Decision Forest	0,999965	0,999965	0,999281	0,999965	0,995868	0,999895004	0,992735325
Multiclass Logistic Regression	0,959663	0,959663	0,434057	0,959663	0,978345	0,883804589	0,255357878
Two Class Support Vector Machin	0,955191	0,955191	0,429992	0,955191	0,975664	0,87150657	0,250421872

Após a realização de todos os testes desta tipologia, chegou-se à conclusão de que a métrica de hiperparametrização que se traduzia numa melhor performance foi o AUC. Assim, apenas foram tidos em conta os resultados desse mesmo teste e foi selecionado o algoritmo *Multiclass Boosted Decision Tree,* com um Macro *Recall* de 0,999079 e Macro *F1-Score* de 0,99724703. Como resultado da hiperparametrização, o algoritmo selecionado obteve como melhores parâmetros os seguintes valores:

- Número Máximo de Folhas por Árvore – 128
- Número Mínimo de Amostras por Nó Folha – 50
- Taxa de Aprendizagem – 0,05
- Número de Árvores Construídas – 500

6.3.3 Validação Cruzada

Os testes realizados anteriormente tinham o pressuposto que os dados estavam divididos em *dataset* de treino e validação, assim como, o *dataset* de treino estava *oversampled* através do SMOTE. No entanto, tal pressuposto pode levar a que o modelo fique demasiado influenciado pelo *dataset* de treino e não consiga generalizar o suficiente, originando *overfitting*.

Para evitar o fenómeno de *overfitting*, foi tomada a decisão de se efetuar uma validação cruzada, sendo que esta utiliza todo o *dataset* e efetua *k* partições do mesmo. Iterativamente o modelo é treinado através de *k-1* conjuntos de dados, utilizando depois o *k* conjunto para

validar o modelo. Este processo é repetido *k* vezes, sendo depois calculadas as médias e desvios padrão de cada iteração para se estimar o desempenho do modelo. Contudo, uma vez que o *dataset* se encontra desbalanceado e dado que na validação cruzada os dados são divididos aleatoriamente, tomou-se a decisão de comparar esta com a validação cruzada estratificada *k-fold*. Este tipo de validação cruzada estratifica os dados de modo que cada partição seja um bom representante dos dados, garantindo as proporções da classe [140].

A validação cruzada, em particular a validação cruzada *k-fold*, possui como único parâmetro o valor de *k*, ou seja, o número de partições a aplicar ao *dataset*. Na literatura é comummente utilizado valor de *k=10*, sendo que quanto maior este for maior será a variância e menor será o enviesamento. Portanto, para evitar o fenómeno de *overfitting* torna-se necessário efetuar um ajuste do valor de *k*, para se obter um bom compromisso entre a variância e o enviesamento. Assim, foram realizados testes para três diferentes valores de *k*, efetuando validação cruzada *k-fold* e validação cruzada estratificada *k-fold*, sendo os resultados posteriormente comparados [141].

Como primeiro teste foi tido em conta o valor de *k*=5, sendo que, os resultados obtidos encontram-se na Tabela 28 e Tabela 29. Sobre o teste de validação cruzada *k-fold*, presente na Tabela 28, é possível aferir que os algoritmos *Two Class Decision Forest e Multiclass Decision Forest* se destacam face aos restantes, no que diz respeito às métricas Macro *F1-Score*, obtendo 0,986766956 e 0,979542682 respetivamente. Ou seja, para a validação cruzada *k-fold*, onde *k*=5, o algoritmo com melhor performance foi o *Two Class Decision Forest.* Contudo, foi observado para este algoritmo um desvio padrão do Macro *Recall* de 0,000823.

Tabela 28 - Resultados obtidos Validação Cruzada *K*=5

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999946	0,999946	0,999746	0,999946	0,991416	0,999838009	0,986766956
Multiclass Decision Forest	0,999917	0,999917	0,999865	0,999917	0,986463	0,999751021	0,979542682
Multiclass Logistic Regression	0,99663	0,99663	0,707757	0,99663	0,596415	0,989924032	0,273250261
Two Class Support Vector Machine	0,997155	0,997155	0,664757	0,997155	0,594905	0,991489259	0,24831162
Two Class Boosted Decision Tree	0,989379	0,989379	0,451102	0,989379	0,416075	0,968474219	0,081248433
Multiclass Boosted Decision Tree	0,982482	0,982482	0,388418	0,982482	0,388336	0,948361265	0,058581501

Na Tabela 29 encontram-se os resultados obtidos na validação cruzada estratificada, onde *k*=5. Através desta é possível aferir que os algoritmos com melhor performance foram os mesmos do teste anterior, sendo que existe uma deterioração da métrica Macro *F1-Score*. No entanto, o desvio padrão da métrica Macro *Recall* diminui para 0,000688.

Tabela 29 - Resultados obtidos Validação Cruzada Estratificada *K*=5

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999946	0,999946	0,999772	0,999946	0,990832	0,999838009	0,985932336
Multiclass Decision Forest	0,999911	0,999911	0,999533	0,999911	0,985443	0,999733024	0,977534351
Multiclass Logistic Regression	0,996674	0,996674	0,750654	0,996674	0,596196	0,99005515	0,297418848
Two Class Support Vector Machine	0,997141	0,997141	0,664708	0,997141	0,594273	0,991447498	0,247881793
Two Class Boosted Decision Tree	0,959521	0,959521	0,532963	0,959521	0,518611	0,883412321	0,145300706
Multiclass Boosted Decision Tree	0,967748	0,967748	0,393022	0,967748	0,495554	0,906331026	0,085379009

Um maior desvio padrão pode indicar um maior enviesamento. Assim e dado que a diferença da métrica Macro *F1-Score* é reduzida nestes dois testes, foi tomada a decisão de se optar pela validação cruzada estratificada e pelo algoritmo *Two Class Decision Forest*, quando *k*=5.

Para os testes de *k*=10, foi seguida a mesma estratégia. Assim, os resultados obtidos na validação cruzada para *k*=10 encontram-se na Tabela 30. Através desta verifica-se que o melhor algoritmo foi novamente o *Two Class Decision Forest*, com um Macro *F1-Score* de 0,989305691 e um desvio padrão do Macro *Recall* de 0,002279.

Tabela 30 - Resultados obtidos Validação Cruzada *K*=10

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999959	0,999959	0,99979	0,999959	0,99307	0,999877005	0,989305691
Multiclass Decision Forest	0,999936	0,999936	0,999689	0,999936	0,989158	0,999808012	0,983308478
Multiclass Logistic Regression	0,996656	0,996656	0,734251	0,996656	0,596627	0,99000151	0,288394285
Two Class Support Vector Machine	0,997154	0,997154	0,664751	0,997154	0,594897	0,991486276	0,248303219
Two Class Boosted Decision Tree	0.988511	0,988511	0,459984	0,988511	0,44105	0,965927475	0,091358702
Multiclass Boosted Decision Tree	0,966569	0,966569	0,386435	0,966569	0,410078	0,903022532	0,0630555

Na validação cruzada estratificada, onde *k*=10, o algoritmo com melhor performance foi o *Two Class Decision Forest*. Este obteve um Macro *F1-Score* de 0,987705016, com um desvio padrão de Macro *Recall* de 0,001554.

Tabela 31 - Resultados obtidos Validação Cruzada Estratificada ***κ***=10

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999955	0,999955	0,999901	0,999955	0,99189	0,999865006	0,987705016
Multiclass Decision Forest	0,99993	0,99993	0,999923	0,99993	0,98734	0,999790015	0,980937255
Multiclass Logistic Regression	0,996659	0,996659	0,726286	0,996659	0,59613	0,99001045	0,283504007
Two Class Support Vector Machine	0,997149	0,997149	0,66459	0,997149	0,594749	0,991471361	0,248120352
Two Class Boosted Decision Tree	0,962412	0,962412	0,433315	0,962412	0,46964	0,891421467	0,091727909
Multiclass Boosted Decision Tree	0,934007	0,934007	0,464932	0,934007	0,405454	0,814798824	0,081654506

Uma vez mais, a validação cruzada estratificada obteve uma diminuição do Macro *F1-Score* e do desvio padrão do Macro *Recall.* Contudo, seguindo a mesma linha de raciocínio dos testes anteriores, para *K*=10 foi selecionado a validação cruzada estratificada e o algoritmo *Two Class Decision Forest.*

Os resultados dos testes de validação cruzada para *K*=15, encontram-se na Tabela 32 e Tabela 33. Na primeira encontram-se ilustrados os resultados da validação cruzada *k-fold*, onde é possível aferir que o algoritmo *Two Class Decision Forest* obteve melhores resultados. Este conseguiu um Macro *F1-Score* de 0,990160312 e um desvio padrão de 0,002583.

Tabela 32 - Resultados obtidos Validação Cruzada *K*=15

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999964	0,999964	0,999884	0,999964	0,993548	0,999892004	0,990160312
Multiclass Decision Forest	0,999942	0,999942	0,999887	0,999942	0,99065	0,99982601	0,985830097
Multiclass Logistic Regression	0,996654	0,996654	0,707396	0,996654	0,596436	0,98999555	0,273061994
Two Class Support Vector Machine	0,997154	0,997154	0,664734	0,997154	0,594904	0,991486276	0,248298334
Two Class Boosted Decision Tree	0,987958	0,987958	0,477494	0,987958	0,461882	0,964307283	0,103559143
Multiclass Boosted Decision Tree	0,945441	0,945441	0,427139	0,945441	0,394077	0,845090649	0,069003834

Para a validação cruzada estratificada, onde *K*=15, os resultados encontram-se na Tabela 33. Nesta é possível verificar que o melhor algoritmo foi o *Two Class Decision Forest*, com um Macro *F1-Score* de 0,988655615 e um desvio padrão do Macro *Recall* de 0,002575.

Tabela 33 - Resultados obtidos Validação Cruzada Estratificada *K*=15

Algoritmo	***Accuracy***	**Micro *Precision***	**Macro *Precision***	**Micro *Recall***	**Macro *Recall***	**Micro *F1 Score***	**Macro *F1 Score***
Two Class Decision Forest	0,999958	0,999958	0,99985	0,999958	0,992576	0,999874005	0,988655615
Multiclass Decision Forest	0,99994	0,99994	0,999969	0,99994	0,990041	0,999820011	0,98504069
Multiclass Logistic Regression	0,996634	0,996634	0,746957	0,996634	0,595877	0,989935952	0,295061207

Two Class Support Vector Machine	0,997138	0,997138	0,664858	0,997138	0,594036	0,99143855	0,247813043
Two Class Boosted Decision Tree	0,984009	0,984009	0,487033	0,984009	0,447768	0,952790047	0,101749834
Multiclass Boosted Decision Tree	0,968493	0,968493	0,416476	0,968493	0,422482	0,908425796	0,073805063

Nos testes onde *k*=15, verifica-se o mesmo comportamento dos restantes. Ou seja, a validação cruzada estratificada permite diminuir o desvio padrão do Macro *Recall*, diminuindo também o Macro *F1-Score*. Assim, para os presentes testes foi selecionada a validação cruzada estratificada e o algoritmo *Two Class Decision Forest.*

Em jeito de resumo foi elaborada a Tabela 34, a qual indica os algoritmos e tipo de validação cruzada selecionada em cada valor de *k*. Através desta é possível perceber que quanto maior o valor de *k*, maior o valor de Macro *Recall* e Macro *F1-Score.* Contudo, foi também tido em conta o desvio padrão do Macro *Recall*, por forma a evitar enviesamento. Assim, de forma a ir de encontro com a literatura e para conseguir um bom compromisso entre o Macro *F1-Score*, o desvio padrão do Macro *Recall* e o valor de *k*, foi tomada a decisão de selecionar o valor de *k*=10.

Tabela 34 - Resumo Validação Cruzada

Valor de K	**Algoritmo Selecionado**	**Validação Cruzada Selecionada**	**Macro *F1-Score***	**Macro *Recall***	**Desvio Padrão Macro *Recall***
K=5	*Two Class Decision Forest*	Estratificada	0,985932336	0,990832	0,000688
K=10	*Two Class Decision Forest*	Estratificada	0,987705016	0,99189	0,001554
K=15	*Two Class Decision Forest*	Estratificada	0,988655615	0,992576	0,002575

6.3.4 Conclusão dos Testes

Após os vários testes descritos na seção anterior, torna-se pertinente efetuar uma comparação das diversas pontuações do Macro *F1-Score*. Sendo assim, na Tabela 35 encontra-se representado um resumo geral dos resultados. Através desta é possível perceber qual o algoritmo que apresenta melhor Macro *F1-Score* e consecutivamente melhor performance. O *Multiclass Boosted Decision Tree* obteve uma melhor pontuação de Macro *F1-Score*. Este obteve uma *Accuracy* geral 0,999985, bem como uma Macro *Precision* de 0,999084 e um Macro *Recall* de 0,999079, o que perfaz uma Micro *F1-Score* de 0,99724703.

Tabela 35 - Conclusão dos Testes

Teste	Melhor Algoritmo	Macro *F1-Score*
Condição de separação *dataset*	*Two Class Decision Forest*	0,9999835
Hiperparametrização	*Multiclass Boosted Decision Tree*	0,99724703
Validação Cruzada	*Two Class Decision Forest*	0,987705016

Não obstante a elevada performance do modelo, este poderá indicar que o modelo se encontra demasiado influenciado pelo *dataset* de treino. Assim, para evitar o fenómeno de *overfitting* do modelo foi tomada a decisão de penalizar a performance do modelo e selecionar o algoritmo *Two Class Decision Forest* com validação cruzada estratificada. Na Figura 81 encontra-se ilustrada a matriz confusão deste modelo, com uma *Accuracy* geral de 0,999955, bem como uma Macro *Precision* de 0,999901 e um Macro *Recall* de 0,99189, o que perfaz um Macro *F1-Score* de 0,987705016.

Apesar da performance obtida, os valores das métricas do modelo podem ser explicados através da exploração de dados realizada inicialmente, na secção 5.1.2.2. Nesta foi realizada uma análise dos principais influenciadores das falhas e paragens, tendo sido obtidos valores acima dos 50% para as falhas e acima dos 100% para as paragens.

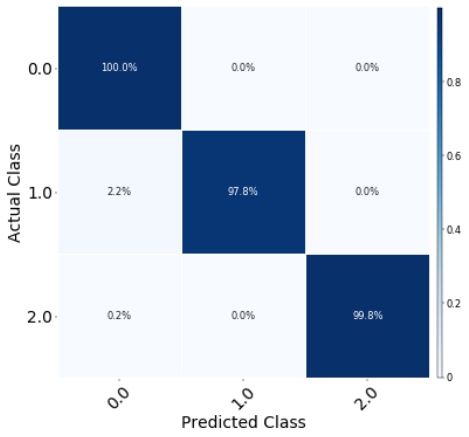


Figura 81 - Matriz Confusão Modelo Final

7 Conclusões

O presente capítulo tem como objetivo resumir e concluir todo o trabalho efetuado. É dado um especial destaque aos objetivos atingidos, às limitações do projeto e ao trabalho futuro.

7.1 Objetivos Realizados

O objetivo principal do presente projeto foi elaborar uma análise e um estudo de técnicas de ML e a partir delas desenvolver um serviço para melhorar e reduzir o esforço na manutenção reativa num processo de limpeza e tratamento de rolhas de cortiça. Esse objetivo levava a um outro, que passava por aproximar o cliente à temática do ML, bem como dotar o mesmo de conhecimento proveniente dos dados produzidos por si.

Assim sendo e tendo por base a Tabela 36, é possível aferir que todos os objetivos foram atingidos.

Tabela 36 - Objetivos Gerais

Objetivo Geral	**Estado**
Análise do estado atual da área tecnológica	Concluído
Estudo e análise de técnicas de ML	Concluído
Análise e desenho de um serviço preditivo capaz de reduzir custos e aumentar o desempenho do cliente	Concluído
Implementação de um serviço ***cloud***, atendendo aos requisitos impostos	Concluído

Relativamente aos objetivos específicos, estes encontram-se explicitados na Tabela 37 e foram todos cumpridos. Apesar disso, alguns deles poderão ser alvo de melhorias e serão devidamente abordados na secção 7.2.

Tabela 37 - Objetivos Específicos

Objetivo Específico	**Estado**
Análise e exploração dados, tendo em vista a obtenção de conhecimento para o cliente sobre os dados produzidos por si	Concluído
Experimentação e ensaios de diversas técnicas de ML	Concluído
Utilização de uma arquitetura totalmente *cloud* e integrada	Concluído
Criação de um modelo preditivo	Concluído
Extração e armazenamento dos dados ***On-Prem*** do cliente para ***Data Lake***	Concluído
Notificação a sistemas terceiros	Concluído

7.2 Limitações e Trabalho Futuro

No presente projeto existem casos de uso que poderiam ser alvo de melhoria e trabalho futuro, quer por melhoria da experiência de configuração do serviço, quer pela inclusão de mais fontes de dados. Existe ainda a possibilidade das tecnologias utilizadas, uma vez que são *cloud*,

sofrerem alterações e/ou lançamento de novas versões. Por esse motivo, com o objetivo de se retirar o máximo partido das mesmas existe a possibilidade de manutenção do projeto.

Assim, tendo em isto em conta, são abaixo apresentadas algumas sugestões de trabalho futuro e limitações:

- Análise e exploração de outras fontes de dados que proliferam em torno do processo de limpeza e tratamento, tais como dados dos alarmes das máquinas, dados do sistema *Manufacturing Execution System* (*MES*) e dados da qualidade do produto final;
- Enriquecimento do modelo preditivo com as variáveis mais pertinentes previamente analisadas de outras fontes de dados;
- Ensaio e implementação de outras abordagens distintas à seguida no presente projeto, com o objetivo de comparar com o atual modelo preditivo. Por exemplo ensaiar uma abordagem de análise de séries temporais;
- Implementação de um módulo de explicações do modelo preditivo, com vista a dar reposta à tendência tecnológica da IA Responsável [143]. A versão atual do serviço não possui um *front-end*, logo seria mais apelativo este possuir um conjunto de gráficos que explicariam o modelo e consequentemente o tornavam mais transparente;
- Implementação de um módulo de configuração de notificações mais amigável. O corrente método de configuração de notificações é direcionado para um perfil técnico-funcional. Contudo, seria interessante tornar este método mais amigável com objetivo de ser um processo mais versátil.
- Implementação de *ML Ops* [107]. Desenvolver *pipelines* integradas com *Azure DevOps* para constante treino, validação e publicação do modelo mais atualizado.

7.3 Apreciação Final

A presente dissertação tinha como principal objetivo efetuar um estudo sobre ML, tendo em vista a implementação de um serviço preditivo na área da Indústria 4.0, com valor para o cliente. Como analisado ao longo do presente documento, a Indústria 4.0 e em particular a área de ML é uma tendência tecnológica em crescendo e com imenso potencial.

Do ponto de vista teórico, toda a pesquisa e análise efetuada tornou-se bastante enriquecedora para o aluno e para o cliente. Através desta, grande parte dos fundamentos teóricos de ML ficaram embebidos no presente documento tornando-se uma fonte de conhecimento teórico-prático para ambos.

Já do ponto de vista prático, esta dissertação foi extremamente desafiante para o aluno, uma vez que o fez sair da zona de conforto e mergulhar numa área de desenvolvimento sobre a qual não possuía um vasto conhecimento. Um outro desafio que o aluno tem a convicção de ter cumprido com sucesso foi a implementação totalmente *cloud*, com a extração de dados para a

mesma. Para o cliente, o projeto desenvolvido foi de extrema pertinência e importância e de grosso modo este faz um balanço positivo.

O aluno considera que a metodologia utilizada no projeto com reuniões periódicas com o cliente e orientadora foram de extrema importância. Isto porque, todas estas sessões permitiram obter um feedback constante, auxiliando em eventuais ajustes, duvidas, discussões sobre ML e orientação no desenvolvimento.

Como nota final, o estudante faz um balanço positivo da presente dissertação, tendo obtido um largo conhecimento e experiência em técnicas de ML. Este tem também a convicção que auxiliou o cliente na jornada da Indústria 4.0.

8 Referências

[1] D. Newman, “Top 5 Digital Transformation Trends In Manufacturing,” 8 Agosto 2017. [Online]. Available: https://www.forbes.com/sites/danielnewman/2017/08/08/top-5-digital-transformation-trends-in-manufacturing/?sh=78f0d875249f. [Acedido em 30 Janeiro 2021].

[2] B. Marr, “What is Industry 4.0? Here's A Super Easy Explanation For Anyone,” 2 Setembro 2018. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2018/09/02/what-is-industry-4-0-heres-a-super-easy-explanation-for-anyone/?sh=7638f3c89788. [Acedido em 30 Janeiro 2021].

[3] República Portuguesa, “Portugal tem os recursos essenciais para estar na vanguarda da revolução digital,” 30 Janeiro 2017. [Online]. Available: https://www.portugal.gov.pt/pt/gc21/comunicacao/noticia?i=20170130-pm-economia-digital. [Acedido em 25 Janeiro 2021].

[4] I-Scoop, “Industry 4.0: the fourth industrial revolution – guide to Industrie 4.0,” 2020. [Online]. Available: https://www.i-scoop.eu/industry-4-0/. [Acedido em 30 Janeiro 2021].

[5] institute of Entrepreneurship Development (iED), “The 4 Industrial Revolutions,” 30 Junho 2019. [Online]. Available: https://ied.eu/project-updates/the-4-industrial-revolutions/. [Acedido em 30 Janeiro 2021].

[6] J. Prinsloo, . J. C. Vosloo e E. H. Mathews, “TOWARDS INDUSTRY 4.0: A ROADMAP FOR THE SOUTH AFRICAN HEAVY INDUSTRY SECTOR,” p. 13, Outubro 2019.

[7] Y. Demchenko, P. Grosso, C. de Laat e P. Membrey, “Addressing Big Data Issues in Scientific Data Infrastructure,” em *The 2013 International Conference on Collaboration Technologies and Systems (CTS 2013)*, San Diego, CA, USA, 2013.

[8] A. Oussous, F.-Z. Benjelloun, A. A. Lahcen e S. Belfkih, “Big Data technologies: A survey,” p. 18, Outubro 2018.

[9] H. Ashmore, “Industry 4.0 and the Impacts of Machine Learning on the Manufacturing Industry,” 17 Novembro 2020. [Online]. Available: https://aithority.com/machine-learning/industry-4-0-and-the-impacts-of-machine-learning-on-the-manufacturing-industry/. [Acedido em 30 Janeiro 2021].

[10] L. Lopes e A. Trindade, “António Amorim: “Vamos eliminar o TCA até 2020”,” *VINHO Grandes Escolhas,* Dezembro 2018.

[11] Scrum Portugal, “AGILE MANIFESTO,” 18 Abril 2017. [Online]. Available: http://www.scrumportugal.pt/agile-manifesto/. [Acedido em 20 Fevereiro 2021].

[12] G. Windsor, “5 Stages of the Agile System Development Life Cycle,” 28 Fevereiro 2020. [Online]. Available: https://medium.com/brightwork-collaborative-project-management-blog/5-stages-of-the-agile-system-development-life-cycle-brightwork-com-a207bdf61696. [Acedido em 20 Fevereiro 2021].

[13] R. Wirth e J. Hipp, “CRISP-DM: Towards a Standard Process Model for Data Mining,” em *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, Manchester, UK, 2000.

[14] L. Wehrstein e D. B. Bachmann, “CRISP-DM ready for Machine Learning Projects,” 19 Dezembro 2020. [Online]. Available: https://towardsdatascience.com/crisp-dm-ready-for-machine-learning-projects-2aad9172056a. [Acedido em 30 Janeiro 2021].

[15] S. J. Russell e P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.

[16] S. v. Duin e N. Bakhshi, “Part 1: Artificial Intelligence Defined - The most used terminology around it,” Março 2017. [Online]. Available: https://www2.deloitte.com/fi/fi/pages/technology/articles/part1-artificial-intelligence-defined.html. [Acedido em 8 Janeiro 2021].

[17] M. M. Mijwel , “History of Artificial Intelligence,” p. 6, Abril 2015.

[18] P. McCorduck, Machines Who Think, USA: A K Peters, Ltd, 1979.

[19] J. Retto, “SOPHIA, FIRST CITIZEN ROBOT OF THE WORLD,” p. 9, Novembro 2017.

[20] M. Si, “China Daily || Robots help medics combat epidemic,” 6 Março 2020. [Online]. Available: http://global.chinadaily.com.cn/a/202003/06/WS5e61dd33a31012821727cedc.html. [Acedido em 30 Janeiro 2021].

[21] A. Montaqim, “Cheetah Mobile’s epidemic prevention and control robots keeping busy in China,” 26 Agosto 2020. [Online]. Available: https://roboticsandautomationnews.com/2020/08/26/cheetah-mobiles-epidemic-prevention-and-control-robots-keeping-busy-in-china/35595/. [Acedido em 30 Janeiro 2021].

[22] M. Mohri, A. Rostamizadeh e A. Talwalkar, Foundations of Machine Learning, Second Edition, Cambridge, Massachusetts: MIT Press Ltd, 2018.

[23] W. T. Chen, H. C. Merrett e N. Fauzia, “A Review of Using Technology to Support Value Engineering Study,” Janeiro 2021.

[24] V. Woods, “Gartner Identifies the Top 10 Strategic Technology Trends for 2016,” Gartner, 6 Outubro 2015. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2015-10-06-gartner-identifies-the-top-10-strategic-technology-trends-for-2016. [Acedido em 3 Janeiro 2021].

[25] L. Goasduff, “2 Megatrends Dominate the Gartner Hype Cycle for Artificial Intelligence, 2020,” 28 Setembro 2020. [Online]. Available: https://www.gartner.com/smarterwithgartner/2-megatrends-dominate-the-gartner-hype-cycle-for-artificial-intelligence-2020/. [Acedido em 30 Janeiro 2021].

[26] K. Panetta, “Gartner Top Strategic Technology Trends for 2021,” Gartner, 19 Outubro 2020. [Online]. Available: https://www.gartner.com/smarterwithgartner/gartner-top-strategic-technology-trends-for-2021/. [Acedido em 3 Janeiro 2021].

[27] B. Marr, “The 5 Biggest Technology Trends In 2021 Everyone Must Get Ready For Now,” Forbes, 14 Setembro 2020. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2020/09/14/the-5-biggest-technology-trends-in-2021-everyone-must-get-ready-for-now. [Acedido em 3 Janeiro 2021].

[28] B. Marr, “These 25 Technology Trends Will Define The Next Decade,” Forbes, 20 Abril 2020. [Online]. Available: https://www.forbes.com/sites/bernardmarr/2020/04/20/these-25-technology-trends-will-define-the-next-decade. [Acedido em 3 Janeiro 2021].

[29] M. J. Miller, “Gartner's Top Strategic Technology Trends for 2021,” 22 Outubro 2020. [Online]. Available: https://sea.pcmag.com/news/39833/gartners-top-strategic-technology-trends-for-2021. [Acedido em 30 Janeiro 2021].

[30] A. Kantarci, “131 Myth-Busting Statistics on Artificial Intelligence (AI) in 2021,” 1 Janeiro 2021. [Online]. Available: https://research.aimultiple.com/ai-stats/#market-size. [Acedido em 30 Janeiro 2021].

[31] IAPMEI, “Capacitar i4.0,” IAPMEI, 2020. [Online]. Available: https://www.iapmei.pt/Paginas/Capacitar-i4-0.aspx. [Acedido em 3 Janeiro 2021].

[32] A. Osterwalder, Y. Pigneur, G. Berna e A. Smith, Value Proposition Design: How to Create Products and Services Customers Want, WILEY, 2014.

[33] B2B International, “What is the Value Proposition Canvas?,” [Online]. Available: https://www.b2binternational.com/research/methods/faq/what-is-the-value-proposition-canvas/. [Acedido em 6 Fevereiro 2021].

[34] Peter Thomson, “What is the Value Proposition Canvas?,” Novembro 2013. [Online]. Available: https://www.peterjthomson.com/2013/11/value-proposition-canvas/. [Acedido em 6 Fevereiro 2021].

[35] M. Lewandowski, “Designing the Business Models for Circular Economy - Towards the Conceptual Framework,” p. 28, Janeiro 2016.

[36] J. S. Borza, “FAST Diagrams: The Foundation for Creating Effective Function Models,” p. 10, 28 Novembro 2011.

[37] T. D. C. Frazão, D. G. G. Camilo, E. L. S. Cabral e R. P. Souza, “Multicriteria decision analysis (MCDA) in health care: a systematic review of the main characteristics and methodological steps,” p. 16, 2018.

[38] M. M. e. a. (. Widianta, , “Multicriteria decision analysis (MCDA) in health care: a systematic review of the main characteristics and methodological steps,” *Journal of Physics: Conference Series,* 2018.

[39] Z. M. Çınar, A. A. Nuhu, Q. Zeeshan, O. Korhan, M. Asmael e B. Safaei, “Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0,” p. 42, 5 Outubro 2020.

[40] M. Canizo, E. Onieva, A. Conde, S. Charramendieta e S. Trujillo, “Real-time Predictive Maintenance for Wind Turbines Using Big Data Frameworks,” em *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, Dallas, 2017.

[41] C. J. Su e S. F. Huang, “Real-time big data analytics for hard disk drive predictive maintenance,” p. 9, 2018.

[42] M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni e J. Loncarski, “Machine Learning approach for Predictive Maintenance in Industry 4.0,” p. 6, Julho 2018.

[43] T. Praveenkumar, M. Saimurugan, P. Krishnakumar e K. I. Ramachandran, “Fault diagnosis of automobile gearbox based on machine learning techniques,” em *12th GLOBAL CONGRESS ON MANUFACTURING AND MANAGEMENT, GCMM 2014*, 2014.

[44] J. Mathew, M. Luo e C. K. Pang, “Regression Kernel for Prognostics with Support Vector Machines,” p. 10, 2017.

[45] N. Kolokas, T. Vafeiadis, D. Ioannidis e D. Tzovaras, “Forecasting faults of industrial equipment using machine learning classifiers,” p. 6, 2018.

[46] I. Amihai, M. Chioua, R. Gitzel, A. M. Kotriwala, D. Pareschi, G. Sosale e S. Subbiah, “Modeling Machine Health Using Gated Recurrent Units with Entity Embeddings and K-Means Clustering,” p. 10, 2018.

[47] B. Brik, B. Bettayeb, M. Sahnoun e F. Duval, “Towards Predicting System Disruption in Industry 4.0: Machine Learning-Based Approach,” *The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40),* p. 8, Maio 2019.

[48] O. C. Carrasco, “Support Vector Machines for Classification,” 7 Julho 2019. [Online]. Available: https://towardsdatascience.com/support-vector-machines-for-classification-fc7c1565e3. [Acedido em 25 Fevereiro 2021].

[49] M. O. Riedl, “Human-centered artificial intelligence and machine learning,” p. 4, 7 Fevereiro 2019.

[50] T. Munakata, Fundamentals of the New Artificial Intelligence - Neural, Evolutionary, Fuzzy and More, Cleveland, OH, 44115, USA: Springer, 2008.

[51] Honda, “Honda Debuts New ASIMO,” 13 Dezembro 2005. [Online]. Available: https://global.honda/newsroom/news/2005/c051213-asimo-eng.html. [Acedido em 21 Fevereiro 2021].

[52] New Delhi Television Ltd, “Honda's ASIMO robot turns 10,” 2 Novembro 2010. [Online]. Available: https://www.ndtv.com/photos/business/hondas-asimo-robot-turns-10-8511#photo-104099. [Acedido em 22 Fevereiro 2021].

[53] J. Brownlee, “A Tour of Machine Learning Algorithms,” 12 Agosto 2019. [Online]. Available: https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/. [Acedido em 23 Janeiro 2021].

[54] E. G. Learned-Miller, “Introduction to Supervised Learning,” Fevereiro 2014.

[55] T. Dietterich, “Overfitting and Undercomputing in Machine Learning,” *ACM Computmg Survevs,* vol. 27, 1995.

[56] P. Cunningham, M. Cord e S. J. Delany, “Supervised Learning,” em *Machine Learning Techniques for Multimedia: Case Studies on Organization and Retrieval*, Springer, 2008, pp. 21-49.

[57] M. Alloghani, D. Al-Jumeily, J. Mustafina e A. Hussain, “A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science,” em *Supervised and Unsupervised Learning for Data Science*, Springer, 2020, pp. 3-21.

[58] A. C. Lorena, L. F. Jacintho , M. F. Siqueira, R. De Giovanni, L. G. Lohmann, A. C. de Carvalho e M. Yamamoto, "Comparing machine learning classifiers in potential distribution modelling," 2011.

[59] M.-L. Zhang e Z.-H. Zhou, "ML-KNN:A lazy learning approach to multi-label learning," 15 Dezembro 2006.

[60] F. Franssen, P. Alter, N. Bar, B. J. Benedikter, S. Iurato, D. Maier, M. Maxheim, F. K. Rössler, M. A. Spruit, C. F. Vogelmeier, E. F. Wouters e B. Schmeck, "Personalized medicine for patients with COPD: Where are we?," *International Journal of COPD,* vol. 14, 2019.

[61] S. Ray, "6 Easy Steps to Learn Naive Bayes Algorithm with codes in Python and R," 11 Setembro 2017. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/. [Acedido em 23 Janeiro 2021].

[62] W. N. H. W. Mohamed, M. N. M. Salleh e A. H. Omar, "A Comparative Study of Reduced Error Pruning Method in Decision Tree Algorithms," em *2012 IEEE International Conference on Control System, Computing and Engineering*, Malaysia, 2012.

[63] A. Singh, N. Thakur e A. Sharma, "A Review of Supervised Machine Learning," em *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, 2016.

[64] S. ALBAWI, T. A. MOHAMMED e S. AL-ZAWI, "Understanding of a Convolutional Neural Network," 2017.

[65] "Machine Learning - Artificial Neural Networks," Tutorials Point, [Online]. Available: https://www.tutorialspoint.com/machine_learning/machine_learning_artificial_neural_networks.htm. [Acedido em 23 Janeiro 2021].

[66] S. B. Unadkat, M. M. Ciocoiu e L. R. Medsker, "Introduction," em *RECURRENT NEURAL NETWORKS: Design and Applications*, CRC Press LLC, 2001.

[67] W. Zaremba, I. Sutskever e O. Vinyals, "RECURRENT NEURAL NETWORK REGULARIZATION," em *International Conference on Learning Representations*, 2015.

[68] A. Tch, "Towards Data Science || The mostly complete chart of Neural Networks, explained," 4 Agosto 2017. [Online]. Available: https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464. [Acedido em 23 Janeiro 2021].

[69] D. R. B. Dhumale, N. D. Thombare e P. M. Bangare, “Machine Learning: A Way of Dealing with Artificial Intelligence,” em *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, 2019.

[70] R. Gandhi, “Introduction to Machine Learning Algorithms: Linear Regression,” 27 Maio 2018. [Online]. Available: https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a. [Acedido em 23 Janeiro 2021].

[71] D. Sharma e N. Kumar, “A Review on Machine Learning Algorithms, Tasks and Applications,” *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET),* vol. 6, 2017.

[72] P. Dayan, M. Sahani e G. Deback, “Unsupervised Learning,” em *The MIT Encyclopedia of the Cognitive Sciences*, 1999, pp. 857-859.

[73] P. Rai e S. Singh, “A Survey of Clustering Techniques,” *International Journal of Computer Applications,* vol. 7, nº 12, 2010.

[74] N. Sharma, “Understanding the Mathematics behind K-Means Clustering,” 21 Fevereiro 2020. [Online]. Available: https://heartbeat.fritz.ai/understanding-the-mathematics-behind-k-means-clustering-40e1d55e2f4c. [Acedido em 23 Janeiro 2021].

[75] A. Saxena, M. Prasad, A. Gupta, N. Bharill , O. P. Patel, A. Tiwari, M. J. Er, W. Ding e C.-T. Lin, “A review of clustering techniques and developments,” em *Neurocomputing*, 2017, pp. 664-681.

[76] J. Shin, T. A. Badgwell, K.-H. Liu e J. H. Lee, “Reinforcement Learning – Overview of recent progress and implications for process control,” em *Computers and Chemical Engineering*, 2019, pp. 282-294.

[77] S. Ali e K. A. Smith, “On learning algorithm selection for classification,” *Applied Soft Computing,* pp. 119-138, 2006.

[78] D. H. Wolpert e W. G. Macready, “No Free Lunch Theorems for Search,” Technical Report SFI-TR-05-010, anta Fe Institute, Santa Fe, NM, 1996.

[79] M. Hossin e M. N. Sulaiman, “A REVIEW ON EVALUATION METRICS FOR DATA CLASSIFICATION EVALUATIONS,” *International Journal of Data Mining & Knowledge Management Process (IJDKP),* vol. 5, nº 2, 2015.

[80] M. Grandini, E. Bagli e G. Visani, “METRICS FOR MULTI-CLASS CLASSIFICATION: AN OVERVIEW,” p. 17, 14 Agosto 2020.

[81] A. Botchkarev, “Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology,” *Interdisciplinary Journal of Information, Knowledge, and Management,* pp. 45-79, 2019.

[82] D. Saunders, “The Bias-Variance Tradeoff,” Minds, Brains, and Programs, 17 Julho 2017. [Online]. Available: https://djsaunde.wordpress.com/2017/07/17/the-bias-variance-tradeoff/. [Acedido em 3 Março 2021].

[83] F. Stulp e O. Sigaud, “Many regression algorithms, one unified model: A review,” p. 20, 5 Junho 2015.

[84] J. Brownlee, “Naive Bayes for Machine Learning,” 11 Abril 2016. [Online]. Available: https://machinelearningmastery.com/naive-bayes-for-machine-learning/. [Acedido em 23 Janeiro 2021].

[85] J. Brownlee, “Classification And Regression Trees for Machine Learning,” 8 Abril 2016. [Online]. Available: https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/. [Acedido em 23 Janeiro 2021].

[86] J. Brownlee, “Machine Learning Mastery || 10 Clustering Algorithms With Python,” 6 Abril 2020. [Online]. Available: https://machinelearningmastery.com/clustering-algorithms-with-python/. [Acedido em 23 Janeiro 2021].

[87] I. Dabbura, “K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks,” 17 Setembro 2018. [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a. [Acedido em 23 Janeiro 2021].

[88] S. Sharma, “Data Science Central || Artificial Neural Network (ANN) in Machine Learning,” 8 Agosto 2017. [Online]. Available: https://www.datasciencecentral.com/profiles/blogs/artificial-neural-network-ann-in-machine-learning. [Acedido em 23 Janeiro 2021].

[89] E. Lutins, “Towards Data Science || Ensemble Methods in Machine Learning: What are They and Why Use Them?,” 2 Agosto 2017. [Online]. Available: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f. [Acedido em 23 Janeiro 2021].

[90] Microsoft, “O que é o Azure Machine Learning?,” 4 Novembro 2019. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/overview-what-is-azure-ml. [Acedido em 20 Janeiro 2021].

[91] J. Barnes, Azure Machine Learning - Microsoft Azure Essentials, Redmond, Washington 98052-6399: Microsoft Press , 2015.

[92] Microsoft, “Tutorial: Prever o preço de um automóvel com o estruturador,” 28 Setembro 2020. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/tutorial-designer-automobile-price-train-score. [Acedido em 20 Janeiro 2021].

[93] N. Thacker, “Automated machine learning and MLOps with Azure Machine Learning,” 28 Outubro 2019. [Online]. Available: https://azure.microsoft.com/pt-pt/blog/automated-machine-learning-and-mlops-with-azure-machine-learning/. [Acedido em 20 Janeiro 2021].

[94] Microsoft, “Como funciona a Azure Machine Learning: Arquitetura e conceitos,” 20 Agosto 2020. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/concept-azure-machine-learning-architecture. [Acedido em 20 Janeiro 2021].

[95] Google Cloud, “AI Plataform,” 2020. [Online]. Available: https://cloud.google.com/ai-platform. [Acedido em Janeiro 20 2021].

[96] J. Green, “Google Cloud AI Platform: Hyper-Accessible AI & Machine Learning,” 4 Novembro 2020. [Online]. Available: https://towardsdatascience.com/google-cloud-ai-platform-hyper-accessible-ai-machine-learning-cddd8c3348b3. [Acedido em 20 Janeiro 2021].

[97] AWS Amazon Web Services, “Apresentação do Amazon SageMaker,” 29 Novembro 2017. [Online]. Available: https://aws.amazon.com/pt/about-aws/whats-new/2017/11/introducing-amazon-sagemaker/. [Acedido em 22 Janeiro 2021].

[98] AWS Amazon Web Services, “AWS: Amazon SageMaker,” AWS Amazon Web Services, [Online]. Available: https://aws.amazon.com/pt/sagemaker/. [Acedido em 22 Janeiro 2021].

[99] Microsoft Azure, “Preços de Azure Machine Learning,” [Online]. Available: https://azure.microsoft.com/pt-pt/pricing/details/machine-learning/. [Acedido em 9 Fevereiro 2021].

[100] Amazon AWS, “Definição de preço do Amazon SageMaker,” [Online]. Available: https://aws.amazon.com/pt/sagemaker/pricing/. [Acedido em 9 Fevereiro 2021].

[101] Google Cloud, “AI Platform Pricing,” [Online]. Available: https://cloud.google.com/ai-platform/pricing. [Acedido em 9 Fevereiro 2021].

[102] C. Breuel, “ML Ops: Machine Learning as an Engineering Discipline,” 3 Janeiro 2020. [Online]. Available: https://towardsdatascience.com/ml-ops-machine-learning-as-an-engineering-discipline-b86ca4874a3f. [Acedido em 9 Fevereiro 2021].

[103] Google Cloud, “Como configurar um ambiente MLOps no Google Cloud,” [Online]. Available: https://cloud.google.com/solutions/machine-learning/setting-up-an-mlops-environment. [Acedido em 9 Fevereiro 2021].

[104] J. Simon, “Amazon SageMaker Pipelines Brings DevOps Capabilities to your Machine Learning Projects,” 8 Dezembro 2020. [Online]. Available: https://aws.amazon.com/pt/blogs/aws/amazon-sagemaker-pipelines-brings-devops-to-machine-learning-projects/. [Acedido em 9 Fevereiro 2021].

[105] AWS Amazon, “SageMaker MLOps Project Walkthrough,” [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-projects-walkthrough.html#sagemaker-proejcts-walkthrough-create. [Acedido em 9 Fevereiro 2021].

[106] M. Jalloul, “Continuous integration and delivery of web apps from Atlassian Bitbucket,” 24 Maio 2016. [Online]. Available: https://azure.microsoft.com/pt-pt/blog/continuous-integration-delivery-of-web-apps-from-atlassian-bitbucket/. [Acedido em 9 Fevereiro 2021].

[107] Microsoft, “MLOps: Model management, deployment, and monitoring with Azure Machine Learning,” 17 Março 2020. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/concept-model-management-and-deployment. [Acedido em 9 Fevereiro 2021].

[108] J. Lee, H. Davari, J. Singh e V. Pandhare, “Industrial Artificial Intelligence for industry 4.0-based manufacturing systems,” em *Manufacturing Letters*, 2018, pp. 20-23.

[109] P. ZHENG, H. WANG, Z. SANG, R. Y. ZHONG, Y. LIU, C. LIU, K. MUBAROK, S. YU e X. XU, “Smart manufacturing systems for Industry 4.0: Conceptual framework, scenarios, and future perspectives,” em *Front. Mech. Eng*, 2018, pp. 137-150.

[110] B. Bajic, I. Cosic, M. Lazarevic, N. Sremcev e A. Rikalovic, “Machine Learning Techniques for Smart Manufacturing: Applications and Challenges in Industry 4.0,” *Conference: 9th International Scientific and Expert Conference TEAM 2018,* Outubro 2018.

[111] K. Liulys, “Machine Learning Application in Predictive Maintenance,” em *2019 Open Conference of Electrical, Electronic and Information Sciences (eStream)*, Vilnius, Lithuania, 2019.

[112] X. Yang, D. Lo, X. Xia, Y. Zhang e J. Sun, “Deep Learning for Just-In-Time Defect Prediction,” em *2015 IEEE International Conference on Software Quality, Reliability and Security*, 2015.

[113] К. Рольф e R. Clauberg, “GLOBALIZATION AND TECHNOLOGY EVOLUTION: IMPACT ON ECONOMY AND SOCIETY OF THE USA,” Julho 2020.

[114] P. A. Koen, G. M. Ajamian, S. Boyce, A. Clamen, E. Fisher, S. Fountoulakis, A. Johnson, P. Puri e R. Seibert, “Fuzzy Front End: Effective Methods, Tools and Techniques,” *, Greg M.Ajamian, Scott Boyce,,* p. 28, 2002.

[115] P. Koen, G. Ajamian, R. Burkart, A. Clamen, J. Davidson, R. D’Amore, C. Elkins, K. Herald, M. Incorvia, A. Johnson, R. Karol, R. Seibert, A. Slavejkov e K. Wagner, “PROVIDING CLARITY AND A COMMON LANGUAGE TO THE “FUZZY FRONT END”,” 2001.

[116] pwc, “Industry 4.0: what are the key success factors to become a digital factory?,” 2020. [Online]. Available: https://www.pwc.be/en/news-publications/2020/industry-4-0-key-success-factors-industry-of-future.html. [Acedido em 19 Fevereiro 2021].

[117] D. Mishra, “The Stages Of Industry 4.0: Where Are You Now?,” 19 Outubro 2020. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2020/10/19/the-stages-of-industry-40-where-are-you-now/?sh=2310cc0d6005. [Acedido em 19 Fevereiro 2021].

[118] S. Nicola, E. P. Ferreira e J. J. P. Ferreira, “International Journal of Information Technology & Decision Making ,” pp. 661-703, 2012.

[119] A. Shanker, “Q&A: What is customer value and how do you deliver it?,” *Technology Innovation Management Review,* Fevereiro 2012.

[120] W. T. Chen, H. C. Merrett e N. Fauzia, “A Review of Using Technology to Support Value Engineering Study,” Janeiro 2021.

[121] V. A. Zeithaml, “Consumer Perceptions of Price, Quality and Value: A Means-End Model and Synthesis of Evidence,” *Journal of Marketing,* Julho 1988.

[122] C. A. Gutiérrez, M. J. Montero-Simó, R. A. Araque-Padilla e L. Gutierrez, “Evaluation of perceived value in the consumption of coffee with ethical attributes,” p. 13, Fevereiro 2013.

[123] T. Morphy, “Stakeholder | Definition - What is a stakeholder?,” 2008. [Online]. Available: https://www.stakeholdermap.com/stakeholder-definition.html. [Acedido em 5 Fevereiro 2021].

[124] L. Chung, B. A. Nixon, E. Yu e J. Mylopoulos, Non-Functional Requirements in Software Engineering, Springer Science+Business Media, LLC, 2000.

[125] Microsoft, “What is Software Architecture?,” 14 Janeiro 2010. [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658098(v=pandp.10)?redirectedfrom=MSDN. [Acedido em 21 Fevereiro 2021].

[126] Microsoft, “Introduction to Azure Data Lake Storage Gen2,” 25 Fevereiro 2020. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/storage/blobs/data-lake-storage-introduction. [Acedido em 21 Fevereiro 2021].

[127] Microsoft, “On-premises data gateway architecture,” 15 Julho 2019. [Online]. Available: https://docs.microsoft.com/en-us/data-integration/gateway/service-gateway-onprem-indepth. [Acedido em 21 Fevereiro 2021].

[128] N. Carlini, Ú. Erlingsson e N. Papernot, “Distribution Density, Tails, and Outliers in Machine Learning: Metrics and Applications,” Outubro 2019.

[129] Microsoft, “AzureML: Clip Values,” 5 Junho 2019. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/studio-module-reference/clip-values. [Acedido em 14 Agosto 2021].

[130] D. Singh e B. Singh, “Investigating the impact of data normalization on classification performance,” *Applied Soft Computing Journal,* vol. 27, 2020.

[131] A. Fernández, S. García, F. Herrera e N. V. Chawla, “SMOTE for Learning from Imbalanced Data: Progress andChallenges, Marking the 15-year Anniversary,” *Journal of Artificial Intelligence Research,* vol. 61, pp. 863-905, 2018.

[132] M. Fernandes, A. Canito, V. B. Canedo, L. Conceição, I. Praça e G. Marreiros, “Data analysis and feature selection for predictive maintenance: A case-study in the metallurgic industry,” *International Journal of Information Management,* vol. 46, pp. 252-262, 2019.

[133] Y. Zhang, Y. Li, J. Song, X. Chen, Y. Lu e W. Wang, “Pearson correlation coefficient of current derivatives based pilot protection scheme for long-distance LCC-HVDC transmission lines,” *Electrical Power and Energy Systems,* vol. 116, 2020.

[134] Microsoft, “Azure ML: SMOTE,” 16 Outubro 2019. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/algorithm-module-reference/smote. [Acedido em 20 Setembro 2021].

[135] P. Kruchten, The Rational Unified Process: An Introduction, United States of America: Addison-Wesley, 2004.

[136] S. A. Alvarez, “An exact analytical relation among recall, precision, and classification accuracy in information retrieval,” p. 22, Janeiro 2002.

[137] Q. Yang, L. Tan, B.-Q. Wu, G.-L. Tian, L. Xu, J.-T. Yang, J.-H. Jiang e R.-Q. Yu, "Beyond one-against-all (OAA) and one-against-one (OAO): An exhaustive and parallel half-against-half (HAH) strategy for multi-class classification and applications to metabolomics," *Chemometrics and Intelligent Laboratory Systems,* vol. 204, 2020.

[138] M. Zahid, Y. Chen, A. Jamal e M. Q. Memon, "Short Term Traffic State Prediction via Hyperparameter Optimization Based Classifiers," *Sensors,* nº 3, p. 22, Janeiro 2020.

[139] Microsoft, "Tune Model Hyperparameters," 10 Outubro 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/tune-model-hyperparameters. [Acedido em 03 Outubro 2021].

[140] H. He e Y. Ma, "Imbalanced Learning: Foundations, Algorithms, and Applications," Wiley-IEEE Press, 2013, p. 205.

[141] M. Kuhn e K. Johnson, "Applied Predictive Modeling," Springer, 2013, p. 70.

[142] M. L. Wolverton, "Research Design, Hypothesis Testing, and Sampling," *The appraisal Journal,* pp. 370-382, 2009.

[143] A. B. Arrieta, N. D. Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. G. Lopez, D. Molina, R. Benjamins, R. Chatila e F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion,* vol. 58, pp. 82-115, Junho 2020.

[144] Microsoft, "Machine Learning Algorithm Cheat Sheet for Azure Machine Learning designer," 5 Março 2020. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/algorithm-cheat-sheet. [Acedido em 23 Janeiro 2021].

[145] J. Brownlee, "4 Types of Classification Tasks in Machine Learning," 8 Abril 2020. [Online]. Available: https://machinelearningmastery.com/types-of-classification-in-machine-learning/. [Acedido em 23 Janeiro 2021].

[146] T. Baker, "The machine learning trends transforming finance," 17 Abril 2019. [Online]. Available: https://www.refinitiv.com/perspectives/ai-digitalization/the-machine-learning-trends-transforming-finance/. [Acedido em 30 Janeiro 2021].

[147] Y. Kinha, "An easy guide to choose the right Machine Learning algorithm," Maio 2020. [Online]. Available: https://www.kdnuggets.com/2020/05/guide-choose-right-machine-learning-algorithm.html. [Acedido em 7 Fevereiro 2021].

[148] Cogito Tech LLC, "What are Features in Machine Learning and Why it is Important?," 29 Julho 2019. [Online]. Available: https://cogitotech.medium.com/what-are-features-in-

machine-learning-and-why-it-is-important-e72f9905b54d. [Acedido em 7 Fevereiro 2021].

[149] SAS Software, “Hui Li,” [Online]. Available: https://blogs.sas.com/content/author/huili/. [Acedido em 7 Fevereiro 2021].

[150] H. Li, “Which machine learning algorithm should I use?,” 9 Dezembro 2020. [Online]. Available: https://blogs.sas.com/content/subconsciousmusings/2020/12/09/machine-learning-algorithm-use/. [Acedido em 7 Fevereiro 2021].

[151] Microsoft, “Git integration for Azure Machine Learning,” 16 Novembro 2020. [Online]. Available: https://docs.microsoft.com/pt-pt/azure/machine-learning/concept-train-model-git-integration. [Acedido em 9 Fevereiro 2021].

[152] Microsoft, “Azure ML: Filter Based Feature Selection,” 10 Outubro 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/filter-based-feature-selection. [Acedido em 18 Setembro 2021].

[153] Microsoft, “MLOps: Model management, deployment, lineage, and monitoring with Azure Machine Learning,” 10 Abril 2021. [Online]. Available: https://docs.microsoft.com/en-us/azure/machine-learning/concept-model-management-and-deployment. [Acedido em 1 Outubro 2021].