INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Mecânica

ISEL

Machine Learning aplicado à Gestão de Activos Físicos Industriais

GONÇALO RIBEIRO DE MATOS
(Licenciado em Engenharia Mecânica)

Trabalho Final de Mestrado para obtenção do grau de Mestre
em Engenharia Mecânica

Orientador(es):
José Augusto da Silva Sobral

Júri:
Presidente:
Prof. Doutor Silvério João Crespo Marques
Vogais:
Prof. Doutor Daniel Augusto Estácio Marques Mendes Gaspar
Prof. Doutor José Augusto da Silva Sobral

Março de 2021

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Mecânica

ISEL

Machine Learning aplicado à Gestão de Activos Físicos Industriais

GONÇALO RIBEIRO DE MATOS
(Licenciado em Engenharia Mecânica)

Trabalho Final de Mestrado para obtenção do grau de Mestre em Engenharia Mecânica

Orientador(es):
José Augusto da Silva Sobral

Júri:
Presidente:
Prof. Doutor Silvério João Crespo Marques
Vogais:
Prof. Doutor Daniel Augusto Estácio Marques Mendes Gaspar
Prof. Doutor José Augusto da Silva Sobral

Março de 2021

Agradecimentos

Em primeiro lugar, quero agradecer ao meu irmão, aos meus pais e restante família todo o apoio e motivação que me foram dando ao longo desta longa jornada académica. Sem eles, não seria possível começar e concluir este trajecto.

Um agradecimento especial à minha namorada, Rita Costa, que sempre me tentou chamar à razão nos momentos em que poderia duvidar da conclusão deste curso.

Um agradecimento aos meus amigos que já tinha e aos que fiz no ISEL, todos importantes na conclusão deste percurso, com especial dedicação aos ex-alunos Nuno Beites e Luis Morais, que me acompanham desde o ensino secundário.

Um agradecimento especial ao meu amigo Engº António Cerca, formado em Engª Eletrónica no ISEL, que através dos seus conhecimentos em programação e *Machine Learning* foi sem dúvida uma grande ajuda na conclusão deste Trabalho Final de Mestrado.

Quero agradecer a todos os Professores que, por uma ou outra maneira, me ajudaram a chegar ao fim deste trajecto.

Por fim, quero agradecer ao professor José Sobral pelo conhecimento transmitido, mas também pela orientação, apoio e conselhos que foi dando ao longo deste trabalho, e principalmente a paciência e disponibilidade para concluir este longo projecto.

Glossário / Lista de Acrónimos / Lista de Siglas

ANN – Artificial Neural Networks (Redes Neuronais Artificiais)

CMMS - Computerized Maintenance Management Software

HPC – High Pressure Compressor

HPT – High Pressure Turbine

IoT – Internet of Things (Internet das Coisas)

LinReg – Linear Regression (Regressão Linear)

LPC – Low Pressure Compressor

LPT – Low Pressure Turbine

MAE – Mean Absolute Error

MCC – Manutenção por Controlo de Condição

ML – Machine Learning (Aprendizagem automática)

MSE – Mean Squared Error

OS – Operational Setting

PCA – Principal Component Analysis

PCoE – Prognostics Center of Excellence

RMSE – Root Mean Squared Error

RUL – Remaining Useful Life (Tempo de Vida Útil Restante)

SVM – Support Vector Machine (Máquina de vectores de suporte)

Resumo

Com o rápido avanço da tecnologia moderna em todas as áreas, incluindo a industrial, torna-se necessário e útil a utilização de novas metodologias para a optimização e aprendizagem automática de processos básicos e comuns à maioria das empresas, como produção, manutenção e gestão de activos. O objectivo deste Trabalho Final de Mestrado é entender o que é o *Machine Learning*, quais os seus conceitos e diferentes tipos e a sua aplicação num contexto industrial, na gestão de activos físicos.

Neste trabalho são apresentados os conceitos básicos de *Machine Learning*, os seus tipos (Supervisionados – regressão ou classificação e Não supervisionados – *Clustering*) e descritos alguns algoritmos de cada tipo, como Regressão Linear, Redes Neuronais Artificiais ou *Support Vector Machines*.

Após aquisição de conhecimentos, é realizado um estudo de caso a um *dataset* existente no repositório de dados para prognóstico (PCoE – *Prognostics Center of Excellence*) da NASA. O objectivo deste estudo de caso estipula a previsão da vida útil restante (RUL – *Remaining Useful Life*) de uma frota de motores do mesmo tipo e perceber o quão perto estão do fim da sua vida, baseado nas condições de trabalho e leitura de sensores.

O conjunto de dados é abordado de duas formas distintas, como um problema de regressão e um problema de classificação. É aplicado um algoritmo de regressão (Regressão Linear Múltipla) de forma a tentar prever de forma exacta o RUL e um algoritmo de classificação (SVM - *Support Vector Machines*) de forma a classificar por classes o tempo de vida útil restante de cada motor.

Com os resultados obtidos através da aplicação do algoritmo SVM, é possível prever com uma precisão acima de 90% os motores que se encontram em fim de vida (com um RUL inferior a 50 ciclos) e, desta forma, agendar uma acção de manutenção para os mesmos.

Palavras chave:

Machine Learning, Gestão de Activos, Manutenção, Aprendizagem Automática, Regressão, Classificação, Vida Útil Restante, Regressão Linear, *Support Vector Machines*

Abstract

With the fast progress of modern technology in all areas, including industrial area, it becomes necessary and useful to use new methodologies for the optimization and automatic learning of basic processes common to most companies, such as production, maintenance and asset management. The aim of this Master's Final Work is to understand what Machine Learning is, what its concepts and different types are and its application in an industrial context, in the management of physical assets.

In this work the basic concepts of Machine Learning are presented, its types (Supervised - regression or classification and Unsupervised - Clustering) and some algorithms of each type are described, such as Linear Regression, Artificial Neural Networks or Support Vector Machines.

After acquiring knowledge, a case study is performed on an existing dataset in NASA's Prognostics Center of Excellence (PCoE) data repository. The objective of this case study is to predict the RUL (Remaining Useful Life) of a fleet of engines of the same type and to understand how close they are to the end of their life, based on operating conditions and sensor reading.

The data set is approached in two distinct ways, as a regression problem and a classification problem. A regression algorithm (Multiple Linear Regression) is applied in order to try to predict exactly the RUL and a classification algorithm (SVM - Support Vector Machines) in order to classify by classes the remaining lifetime of each engine.

With the results obtained through the application of the SVM algorithm, it is possible to predict with an accuracy above 90% the motors that are at the end of their life (with a RUL of less than 50 cycles) and thus schedule an maintenance action for them.

Keywords:

Machine Learning, Asset Management, Maintenance, Regression, Classification, Remaining Useful Life, Linear Regression, Support Vector Machine

Índice

Índice de Figuras

Índice de Tabelas

Capítulo 1 – Introdução

O tema "*Machine Learning*" tem obtido muita atenção ultimamente, não só por parte da imprensa, mas também a nível académico e industrial. Com o rápido desenvolvimento da tecnologia moderna, as novas aplicações de internet geram uma larga quantidade de dados a uma velocidade sem precedentes, como vídeo, foto, texto ou voz, utilizando grandes armazenamentos de dados e computação em nuvem. Estes dados têm características de grandes dimensões, o que representa um grande desafio para a análise de dados e tomada de decisão.

Graças às novas tecnologias, o *Machine Learning* de hoje não é como o *Machine Learning* do passado. A partir do reconhecimento de padrões e da teoria de que os computadores poderiam aprender sem terem de ser explicitamente programados para realizar tarefas específicas, vários investigadores interessados em inteligência artificial começaram a pesquisar se as máquinas poderiam aprender através da análise de dados. Enquanto a IA (inteligência artificial) pode ser definida, de modo geral, como a ciência capaz de tornar máquinas capazes de desempenhar tarefas humanas, o *Machine Learning* é uma vertente específica que treina as máquinas para aprender com dados.

Apesar de já existirem alguns algoritmos de *Machine Learning* há bastante tempo, a capacidade de aplicar cálculos matemáticos complexos a grandes quantidades de dados automaticamente, de forma cada vez mais rápida, é um desenvolvimento recente.

1.1. Objectivos

Com o rápido desenvolvimento tecnológico, é possível tornar todos os processos de gestão e manutenção mais fiáveis, eficazes e menos dispendiosos. Com a aplicação de algoritmos de *Machine Learning*, prevê-se que seja possível diminuir o número de falhas de equipamentos ao prever os seus modos de falha, por exemplo.

No presente trabalho, pretende-se descrever de forma clara os grupos de algoritmos existentes de *Machine Learning* e explorá-los individualmente, compreender como e em que áreas são maioritariamente utilizados. Serão utilizados *softwares* para o uso destes algoritmos, de forma a pôr em prática os conhecimentos adquiridos.

Posteriormente, após analisar os algoritmos existentes, serão propostos modelos para aplicar a activos físicos na área industrial, principalmente para fins de gestão e manutenção.

Mais detalhadamente, os objectivos do presente Trabalho Final de Mestrado são:

- Realizar uma descrição das áreas onde se pretende aplicar algoritmos de *Machine Learning*, em contexto industrial;
- Contextualizar e descrever o tema "*Machine Learning*";
    - Exemplos de onde e como pode ser aplicado, noutras áreas;
    - Descrição de forma clara dos grupos de algoritmos existentes de *Machine Learning*;
    - Explorar diferentes modelos de *Machine Learning* e definir quais podem e devem ser utilizados na área em estudo.
- Desenvolver um estudo de caso de aplicação de um algoritmo;
    - Aplicação de um algoritmo num conjunto de activos físicos, em contexto industrial.

1.2. Estrutura do Trabalho

O presente Trabalho Final de Mestrado encontra-se dividido em seis capítulos. Neste primeiro capítulo, pretende-se enquadrar o trabalho desenvolvido, assim como os objectivos propostos.

No capítulo dois, designado por Manutenção e Gestão de Activos, é feita uma breve introdução às áreas onde se pretende implementar algoritmos de aprendizagem automática.

O capítulo três é referente ao enquadramento do tema em relação à indústria e mais recentemente à 4ª revolução industrial.

No capítulo quatro, é feita uma descrição do *Machine Learning*. É realizada uma introdução ao tema, a forma como é aplicado em algumas indústrias, e abordados os diferentes tipos e modelos de *Machine Learning*, a forma como são aplicados e em que contexto podem ser aplicados da melhor forma.

O capítulo cinco é referente ao caso de estudo. É analisado um conjunto de dados existente no repositório de dados para prognóstico da *NASA*, com o objectivo de

determinar o tempo de vida útil restante de motores semelhantes duma mesma frota e perceber quantos ciclos de vida ainda restam até à sua falha.

Por fim, no sexto e último capítulo, apresentam-se algumas conclusões sobre o trabalho realizado, assim como possíveis trabalhos futuros e implementações utilizando os conhecimentos adquiridos neste Trabalho Final de Mestrado.

Nota: este documento foi elaborado segundo o acordo ortográfico anterior ao que está actualmente em vigor.

Capítulo 2 – Manutenção e Gestão de Activos

2.1. Manutenção

A manutenção tem-se tornado uma actividade cada vez mais importante na gestão de uma empresa, uma vez que o bom funcionamento das máquinas e equipamentos é uma prioridade. É actualmente vista como uma actividade geradora de lucro, dado que menos paragens e avarias de equipamentos significam uma maior produção. Segundo a norma EN 13306:2017 – "*Maintenance Terminology"*, manutenção é a combinação de todas as acções técnicas, administrativas e de gestão durante o ciclo de vida de um bem destinadas a mantê-lo, ou restaurá-lo num estado em que possa desempenhar a função requerida [1]. Basicamente, é todo o processo de gestão da condição, situação ou estado de um activo, podendo este estado ser preservado ou alterado.

2.1.1. Evolução histórica da manutenção

A evolução da manutenção é dividida pela maioria dos autores em 4 gerações, desde a 2ª Guerra Mundial. Esta evolução demonstra as necessidades industriais em cada geração.

Na **1ª geração**, entre 1930 e 1950, a manutenção não era vista como geradora de lucro e a produtividade não era prioritária, devido à conjuntura económica. Além disso, os equipamentos eram simples e sobredimensionados para a produção. As técnicas de manutenção baseavam-se apenas em serviços de limpeza, lubrificação, inspecção visual e na reparação após quebra ("*Repair after damage*"), sendo a manutenção correctiva o conceito principal da 1ª geração.

A **2ª geração** veio trazer elementos conceptuais para o desenvolvimento da manutenção preventiva. Após a 2ª Guerra Mundial (1950-1980), com a diminuição da mão-de-obra, houve um aumento da mecanização da indústria e na complexidade das instalações industriais. Por esta altura passou-se a acreditar que o bom funcionamento de um equipamento era uma prioridade para que as falhas fossem evitadas, aumentando assim o seu ciclo de vida. Para isto, a aplicação de alguns elementos como inspecções programadas, planeamento e controlo de sistemas de manutenção e a evolução da ciência computacional aplicada levaram ao desenvolvimento da manutenção preventiva.

Na **3ª geração** (1980-2000) a evolução de técnicas de monitorização da condição, análise de falhas e estudo de riscos veio trazer bases para conceptualizar a manutenção preditiva. A evolução e aplicação destes conceitos foi alimentada pelo aumento das expectativas de manutenção, que forçaram o desenvolvimento de novas tecnologias que puderam ajudar o sector de produção a manter a segurança, qualidade, disponibilidade e fiabilidade dos seus equipamentos e, consequentemente, dos seus processos de produção.

Actualmente, na **4ª geração** (2000 – actualidade), a manutenção centrada na fiabilidade (RCM – *Reliability Centered Maintenance*) veio trazer técnicas avançadas relacionadas com a fiabilidade e disponibilidade que são frequentemente adaptadas a sistemas integrados de gestão, como por exemplo investimentos optimizados, produtos inteligentes, aplicação de algoritmos de *Machine Learning* e redes neuronais, entre outros [2].

2.2. Tipos de manutenção

São várias as formas de realizar manutenção, podendo ser planeada ou não planeada. Na figura 1, estão esquematizados os tipos de manutenção, seguidos de uma breve explicação de cada tipo.

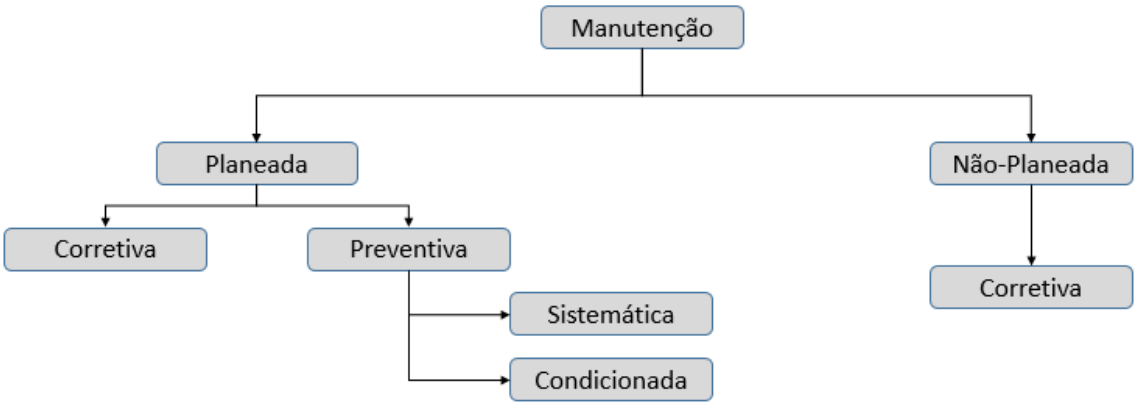

Figura 1 - Tipos de manutenção

Manutenção Correctiva: Manutenção realizada após o reconhecimento de falhas e destinada a colocar um activo num estado no qual ele possa executar a função requerida. Este tipo de manutenção pode ser planeada ou não-planeada, caso seja vantajoso para uma organização realizar acções de manutenção apenas após ocorrência de falhas [1].

Vantagens [3]:

- Não envolve custos relativos à monitorização da condição e manutenção preventiva;
- Os equipamentos não são sujeitos a manutenção excessiva.

Desvantagens [3]:

- Tempo de produção desconhecido;
- Avarias secundárias e falhas catastróficas;
- Perdas de produção;
- Elevados custos de reparação.

Manutenção Preventiva: Manutenção realizada em intervalos de tempo ou critérios pré-determinados e destinada a reduzir a probabilidade de falha ou a degradação das funcionalidades de um activo [1].

Vantagens [3]:

- A manutenção é realizada em data programada e de um modo controlado;
- As avarias não previstas dos equipamentos deverão ser menores;
- Existem assim menos falhas catastróficas e menos perdas de produção.

Desvantagens [3]:

- Os equipamentos são frequentemente "reparados" em data programada e de um modo controlado;
- As intervenções junto dos equipamentos causam frequentemente mais efeitos negativos que positivos, uma vez que o processo de desmontagem, por exemplo, pode causar problemas;
- Ocorrem, apesar de tudo, paragens imprevistas;
- A programação da manutenção é semelhante para todos os equipamentos, não respeitando requisitos individuais de cada equipamento, nem tão pouco a expectativa de tempo de vida útil observado.

A manutenção preventiva divide-se em dois tipos: Sistemática e Condicionada.

Manutenção Sistemática: Manutenção preventiva realizada de acordo com intervalos de tempo estabelecidos ou número de unidades em uso, mas sem investigações anteriores sobre o estado de condição do activo [1].

Manutenção baseada na Condição: Manutenção preventiva que inclui uma combinação de monitorização da condição e/ou inspecção e/ou análises e as acções de manutenção posteriores [1]. Esta é uma importante técnica para assegurar a disponibilidade de um sistema/máquina planeando acções de manutenção e reduzindo a necessidade de manutenção correctiva [4].

2.2.1. Tecnologias modernas de manutenção

Para além dos métodos mais clássicos de manutenção, tem-se realizado alguns progressos no que toca à manutenção baseada na condição, com o objectivo de, por exemplo, detectar falhas antes das mesmas acontecerem. Na figura 2 é possível ver dois tipos de manutenção baseada na condição.

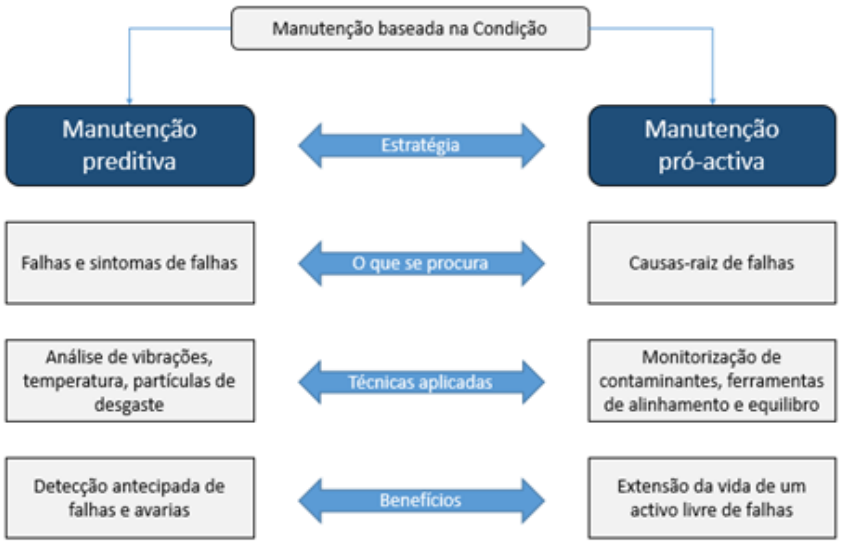

Figura 2 - Manutenção sob Condição [3].

Manutenção Preditiva: Manutenção baseada na condição seguindo previsões provenientes de análises ou de propriedades conhecidas e da avaliação de parâmetros significativos da degradação de um activo [1]. Algumas técnicas de manutenção preditiva

são a análise de vibrações, temperatura, de partículas de desgaste, entre outras. No fundo, este tipo de técnicas permite melhorar a manutenção preventiva, uma vez que altera os intervalos de tempos estabelecidos para efectuar acções de manutenção apenas quando necessário e não sistematicamente, através das previsões calculadas.

Vantagens [3]:

- As perdas de produção não programadas são reduzidas;
- Os componentes dos equipamentos são encomendados e utilizados apenas quando necessário;
- A manutenção é apenas realizada quando necessário;
- Prolonga o tempo de vida útil das máquinas;
- Elimina revisões desnecessárias;
- Possibilita uma operação das máquinas mais eficiente;
- Aumenta a segurança das máquinas;
- Aumenta a qualidade.

Desvantagens [3]:

- Custo dos equipamentos, sistemas, serviços e pessoal.

Manutenção Pró-Activa: Manutenção baseada na condição que cria acções de modo a identificar não apenas os sintomas, mas também as causas-raiz das falhas, geralmente ligadas a factores externos. O objectivo deste tipo de manutenção passa por aumentar a vida de um activo. Uma técnica usada na manutenção pró-activa é, por exemplo, a monitorização de contaminantes [1].

Vantagens [3]:

- O tempo de vida útil dos equipamentos é prolongado;
- A fiabilidade dos equipamentos é optimizada;
- Menos avarias e consequentemente, menos avarias secundárias;
- Redução nas perdas de produção;

- Redução nos custos globais de manutenção.

Desvantagens [3]:

- Custo dos equipamentos, sistemas, serviços e pessoal;
- Necessidade de formação adicional;
- Necessidade de tempo para o investimento inicial;
- Requer uma mudança de filosofia a todos os níveis.

2.3. Gestão de Activos

A gestão de activos permite que uma organização obtenha valor através dos activos na realização dos seus objectivos organizacionais. O que constitui valor dependerá sempre desses objectivos, da natureza e propósito da organização e ainda das necessidades e expectativas dos accionistas [5].

2.3.1. Activo

No fundo, um activo é um item, equipamento ou entidade que tem valor actual ou potencial para uma organização. O valor de cada activo varia de organização para organização e pode ser tangível ou não-tangível, que gere lucro ou não.

A vida de um activo não coincide necessariamente com o período pelo qual uma organização é responsável por ele. Um activo pode providenciar valor para uma ou mais organização durante a sua vida e o seu valor pode mudar durante o seu ciclo de vida.

Uma organização pode optar por gerir os seus activos em grupo de acordo com as suas necessidades e para alcançar benefícios adicionais [5].

2.3.2. Benefícios da gestão de activos

Os benefícios da gestão de activos incluem, entre outros [5]:

- Desempenho financeiro aperfeiçoado;
- Tomada de decisão sobre o investimento em activos;

- Gestão de riscos;
- Serviços melhorados;
- Responsabilidade social demonstrada;
- Conformidade demonstrada;
- Reputação reforçada;
- Sustentabilidade organizacional aperfeiçoada;
- Eficiência melhorada.

2.4. *Machine Learning* em Manutenção e Gestão de Activos

No mundo actual, à medida que o número de activos conectados aumenta, aumenta também a quantidade de dados que estes produzem deixando as empresas sobrecarregadas e sem saber como usar essa quantidade gigante de dados para resolver problemas relacionados com os seus negócios. É necessária uma ferramenta que analise estes dados automaticamente e que torne estes recursos mais eficientes. Através da aplicação de complexos algoritmos é possível pegar em *terabytes* de dados e, por exemplo, através do reconhecimento de padrões detectar anomalias para previsão de falhas [6].

Estas técnicas de aprendizagem automática podem e já começam a ser utilizadas na área da manutenção, principalmente em MCC (Manutenção por Controlo de Condição). São vários os estudos realizados que apresentam resultados positivos quanto à utilização destes algoritmos para detecção de falhas e anomalias, entre outros. Em 2018, foi apresentado um estudo onde foram propostas técnicas supervisionadas e não-supervisionadas para complementar a manutenção por controlo de condição de uns sistemas de propulsão naval de uma embarcação, de forma a prever o seu estado de decadência [7]. Neste estudo, os autores chegaram à conclusão que seria possível efectuar e auxiliar a manutenção por controlo de condição deste problema através de métodos não supervisionados para detecção de anomalias e que podiam ser adoptados para aplicações em tempo real, para fácil e rápida identificação de actividades de manutenção necessárias. Através de técnicas supervisionadas também seria possível identificar falhas, mas seria

necessária uma enorme quantidade de dados com custos elevados, o que seria impraticável.

Outro exemplo de aplicação de técnicas de *Machine Learning* foi apresentado no artigo publicado em Abril de 2018 pela *North West University* [8]. Neste artigo, é feito um estudo de forma a avaliar se seria possível detectar falhas numa instalação de uma central nuclear, com auxílio de ANN (*Artificial Neural Networks*), treinando os parâmetros da instalação e de um simulador que só conseguia identificar falhas em regime estacionário. Os *outputs* da rede neuronal da instalação e do simulador são posteriormente comparados chegando à conclusão que através deste método alcança-se uma melhor identificação de falhas em regimes transientes.

Em 2018, foi realizado um estudo onde foram utilizadas técnicas de *Machine Learning* para a detecção de fugas em tubagens através de Emissão Acústica [9]. Neste caso, foram utilizados algoritmos como PCA (*Principal Component Analysis)* e GA (*Genetic Algorithms)* para seleccionar e/ou reduzir os parâmetros utilizados para a detecção de fugas. Com estes algoritmos, é possível obter uma maior exactidão nos resultados obtidos, mas também tornar a programação mais leve, sem prejudicar os resultados.

Estes são apenas alguns exemplos de aplicações de algoritmos de aprendizagem automática que mostram a sua importância na gestão de activos e como podem ajudar a prever situações de falha de forma a serem corrigidas antes de ocorrerem.

Capítulo 3 – Indústria 4.0

A partir da 1ª revolução industrial, a produção e a manutenção têm evoluído muito. Desde a introdução de máquinas a vapor de forma a mecanizar a produção – 1ª revolução – à produção em massa utilizando energia eléctrica – 2ª revolução – e a aplicação de sistemas IT (*Information Technology*) e PLCs (*Programmable Logic Controllers*) para automação – 3ª revolução – os processos de produção têm-se tornado mais complexos, automatizados e sustentáveis para que as pessoas consigam operar os equipamentos mais fácil e eficientemente [10].

A indústria 4.0, como é denominada a 4ª revolução industrial, é uma tendência actual no domínio da produção, baseada no conceito de "fábrica inteligente" [11]. O objectivo central da indústria 4.0 é preencher as necessidades individuais dos clientes, que afecta áreas como o desenvolvimento, pesquisa e pedidos de produtos, projectos de fabrico, a entrega dos produtos finais à utilização e a reciclagem dos mesmos [10]. A necessidade da indústria 4.0 para as empresas é a conversão dos equipamentos vulgares para equipamentos auto-conscientes e de auto-aprendizagem para melhorar o seu desempenho global e gestão da manutenção dos mesmos.

Figura 3 - Revoluções Industriais

3.1. Pilares da indústria 4.0

A indústria 4.0 assenta em 9 pilares. Estes pilares transformam células de produção isoladas e optimizadas num fluxo de produção completamente integrado, automatizado e optimizado. Isto leva a uma maior eficiência e altera as relações tradicionais de produção entre fornecedores, produtores e clientes assim como entre humanos e máquinas [10].

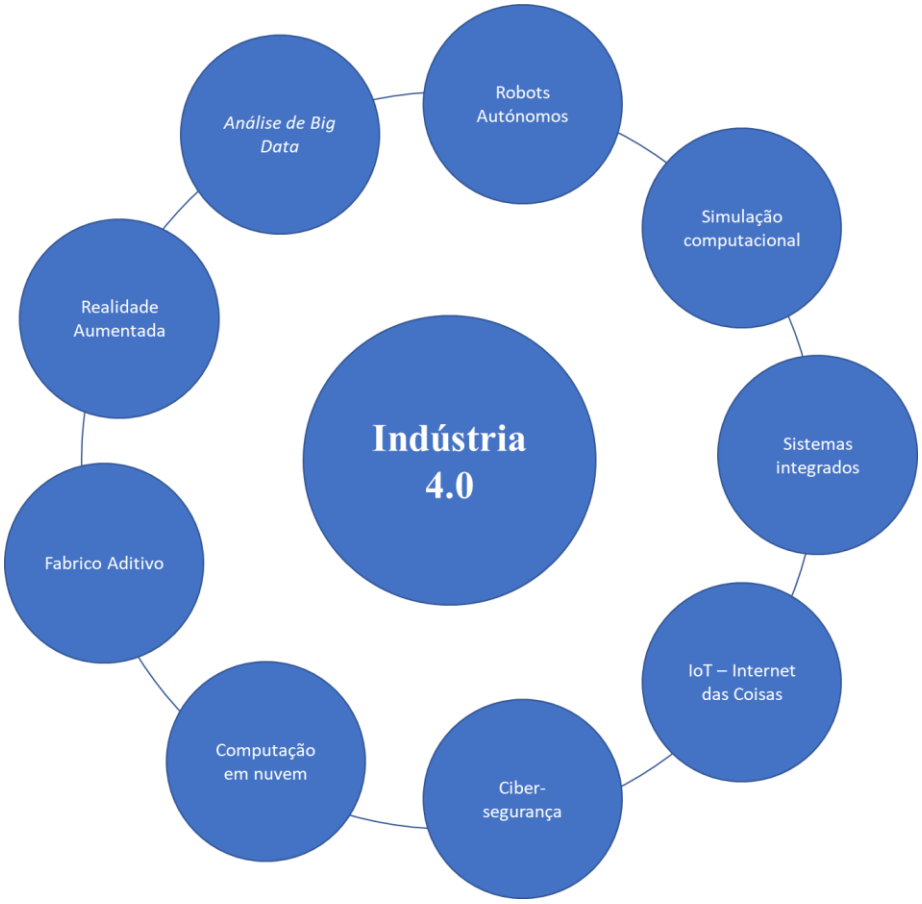

Figura 4 - Indústria 4.0 e os seus 9 pilares

Nos próximos subcapítulos, segue-se uma descrição sucinta de cada pilar, com exemplos do que representam.

3.1.1. *Big Data* e *Data Analytics* – Análise de grandes volumes de dados

A capacidade de armazenar, recolher e analisar quantidades gigantes de dados de várias fontes permite uma optimização dos processos industriais, melhorando o consumo de energia e qualidade de produção nas fábricas. A compreensão e análise rápida desta quantidade de dados diferente tornar-se-á fulcral como apoio à tomada de decisão em tempo real [12]. O processo de *Big Data Analytics* passa por analisar grandes conjuntos

de dados que possam dar informações relevantes. A análise de grandes quantidades de dados pode ser usada em várias áreas e oferece uma vantagem competitiva em relação a outras empresas, podendo beneficiar as suas actividades, marketing, experiência do cliente, entre outros [13].

Um exemplo prático da utilização de *Big Data*, no ramo de entretenimento, é o *Spotify*. Esta aplicação colecciona dados de todos os seus utilizadores em todo o mundo e analisa-os para providenciar recomendações musicais e sugestões individuais para cada utilizador, usando *Big Data Analytics*. Nos transportes, um grande exemplo é a *Uber*. A *Uber* gera e utiliza uma grande quantidade de dados através dos seus condutores, veículos, localizações, viagens, etc. Todos estes dados são analisados e utilizados para prever a procura, oferta e localização de condutores, por exemplo [14].

3.1.2. *Autonomous Robots* – Robots Autónomos

Os *robots* estão a tornar-se cada vez mais autónomos, flexíveis e cooperativos e dentro de pouco tempo será possível interagirem uns com os outros e trabalhar em segurança lado a lado com os humanos, aprendendo com eles. Um *robot* automatizado é utilizado para realizar tarefas de produção autónomas de um modo mais preciso e também para trabalhar em lugares que os operadores não conseguem [10].

O uso de *robots* industriais nas fábricas acelerou com a indústria 4.0. Estes *robots* podem ser utilizados em diversas áreas como produção, logística, actividades de distribuição e podem ser controlados remotamente por humanos. O Kuka LBR IIWA, por exemplo, possui a habilidade de aprender com os seus colegas humanos e verificar, optimizar e documentar as tarefas com a ajuda de *clouds*, não havendo perda de dados [13]. LBR significa "*Leichtbauroboter"* (alemão para *robot* de construção leve) e IIWA "*Intelligent Industrial Work Assistant*". Um dos exemplos da sua utilização, é, por exemplo, a colaboração deste *robot* numa das fábricas da *Ford*, na Alemanha. O LBR IIWA ajuda a instalar amortecedores pesados no popular *Ford Fiesta*, tarefa difícil de implementar com a automação tradicional, que deixava os trabalhadores humanos em risco para efectuar um trabalho repetitivo, ergonomicamente difícil e tecnicamente desafiador num ambiente de ritmo acelerado, por conta própria. Agora, com o LBR IIWA, os trabalhadores podem focar-se em assegurar que a produção funciona sem problemas, enquanto as tarefas mais pesadas são efectuadas pelo *robot* [15].

3.1.3. *Simulation* – Simulação Computacional

Por simulação entenda-se simulação computacional, essencial para garantir a qualidade e eficiência no desenvolvimento de produtos. Na indústria 4.0, pretende-se que as simulações computacionais utilizem também informações da planta de forma a explorar os dados em tempo real, aproximando o mundo físico do mundo virtual, incluindo máquinas, produtos e seres humanos. O uso de simulação de processos de produção pode não só diminuir os tempos de paragem e mudá-los, mas também reduzir as falhas de produção durante a fase inicial [16].

Existem vários tipos de *softwares* de simulação. Um desses exemplos é o *SIMUL8*. Este *software* de simulação permite aos seus utilizadores a criação de um protótipo da implementação dos princípios da indústria 4.0 e encontrar os *outputs* necessários para tomar decisões confiantes, antes de qualquer investimento. Este *software* baseia-se em 4 princípios [17]:

- **Interoperabilidade** – habilidade dos dispositivos e pessoas para comunicar e processar informação entre eles através da Internet das Coisas (*Internet of Things – IoT);*
- **Informação transparente** – a criação de uma cópia virtual do mundo físico utilizando dados de sensores para representar um sistema completo, como por exemplo a planta de uma fábrica;
- **Assistência técnica** – ferramentas de informação que auxiliem as tomadas de decisões através de uma rápida agregação e visualização dos dados de processo, assim como ferramentas físicas que ajudem ou substituam o esforço humano em tarefas perigosas e exigentes;
- **Decisões descentralizadas** – habilidade dos sistemas computacionais para tomarem as suas próprias decisões para eventos regulares num sistema, precisando apenas de supervisão humana quando ocorre um evento excepcional ou confuso.

3.1.4. *System Integration* – Sistemas integrados (horizontais e verticais)

A integração horizontal e vertical de sistemas está totalmente relacionada com a indústria 4.0. O objectivo é garantir que ao acompanhar o processo de revolução industrial, as empresas operem com uma integração universal, de forma integrada e sistémica,

optimizando processos. A integração horizontal consiste em conectar etapas como o desenvolvimento de produtos, produção, logística e distribuição, incluindo parceiros externos, com o objectivo de, no final, entregar valor acrescentado ao cliente. Na integração vertical, o fluxo ocorre verticalmente, ou seja, na composição hierárquica da empresa. Normalmente, o processo vertical é composto por 5 níveis hierárquicos [18]:

- Nível de campo (chão de fábrica) – interface com o processo de produção através de sensores e actuadores;
- Nível de controlo – Regulação tanto das máquinas como dos sistemas;
- Nível de produção – Engloba monitorização, controlo e supervisão;
- Nível operacional – Planeamento de produção, gestão de qualidade e fiabilidade e eficiência dos equipamentos;
- Nível de planeamento empresarial – Gestão e processamento de pedidos, planeamento geral da produção, monitorização de desempenho da empresa/fábrica, gestão dos processos administrativos do negócio, entre outros.

3.1.5. IoT (*Internet of Things*) – A internet das coisas

A "internet das coisas" é um conceito que se refere à conexão digital entre objectos do quotidiano com a internet. Consiste numa rede global de objectos físicos, ambientes, veículos e máquinas interconectados e uniformes que comunicam através de protocolos normalizados, permitindo a troca de informações. Com a aplicação do IoT, as operações de negócio passam a ser mais ágeis e integradas e as capacidades de IoT de uma empesa passarão a ser cruciais no futuro, associadas à agilidade de operações e na tomada de decisão [13].

A IoT é bastante aplicada na concepção de fábricas inteligentes, e um exemplo disso é o iVTP – *Smart Factory Visibility and Traceability Platform*, utilizando *scanners* a *laser* [19]. Basicamente são usados aparelhos/*chips* de identificação por radiofrequência (RFID) para converter todos os recursos em "objectos" de fábrica inteligente (SMO – *Smart Manufacturing Objects*) e as interacções entre eles são capazes de reflectir em tempo real as operações e comportamentos de produção. Através da utilização de *laser-scanners* no chão da fábrica, o iVTP consegue exibir em tempo real os movimentos de vários SMO's e juntar os dados RFID para mostrar o seu estado actual.

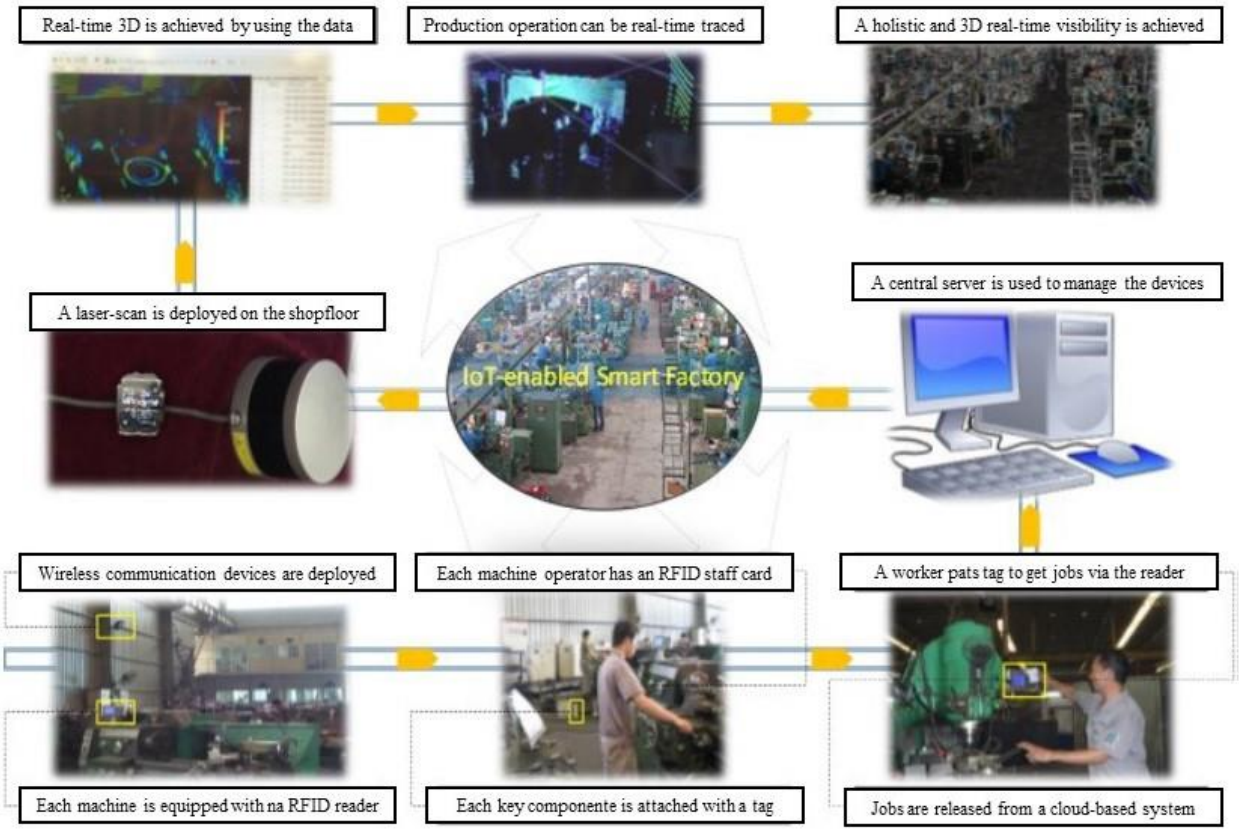

Figura 5 - Aplicação de IoT para uma fábrica inteligente [19].

3.1.6. *CyberSecurity* – Cibersegurança

Com o aumento da conectividade não só entre organizações, mas também internamente, a necessidade de proteger os sistemas industriais críticos e as linhas de produção de ameaças à segurança cibernética aumenta dramaticamente. Como resultado desta protecção, esperam-se comunicações fidedignas e seguras assim como identificações e acessos sofisticados à gestão de máquinas e utilizadores, que acabam por ser essenciais à produção [10]. De forma a defender uma empresa de ataques terroristas cibernéticos, é necessário construir sistemas de defesa e formar os colaboradores para que estejam preparados. Apesar destas soluções terem custos elevados para as empresas, o custo total expectável da implementação de sistemas de segurança é bastante baixo considerando os potenciais efeitos negativos de ataques cibernéticos [13].

3.1.7. *Cloud Computing* – Computação em nuvem

As plataformas sustentadas em "*clouds*" fornecem uma grande redução de custos e tempo e um aumento de eficiência. Com a indústria 4.0, as organizações necessitam de uma

partilha de dados muito maior e de uma resposta rápida à análise e troca de dados entre organizações. O conceito de "*cloud*" permite que haja múltiplas conexões entre diferentes aparelhos e que estes partilhem informações em tempo real [10]. Exemplos deste tipo de plataformas são, por exemplo, a *Google Drive* da *Google* ou a *Microsoft Azure* da *Microsoft*.

Existem muitas empresas que utilizam *clouds* para auxiliar a gestão de activos e manutenção dos mesmos. A *PRIMAVERA*, através do *Valuekeep*, é uma delas [20]. Esta empresa utiliza o *Cloud Computing* para auxiliar o seu CMMS (*Computerized Maintenance Management Software)*. As vantagens de utilizar um sistema em *Cloud* são várias, como por exemplo:

- O *software* está hospedado numa *cloud*, não sendo necessário a instalação do mesmo num computador ou servidor da empresa que o utiliza;
- A informação está disponível em qualquer lado, bastando ter uma conexão via Internet;
- O armazenamento de dados é feito num centro de dados com sistemas de elevada segurança;
- Fácil de utilizar e implementar;
- Actualizações e manutenção da infra-estrutura automáticas;
- Custos de implementação e manutenção mais baixos.

3.1.8. *Additive Manufacturing* – Fabrico Aditivo

O fabrico aditivo é uma tecnologia revolucionária e alternativa para a produção. A flexibilidade e capacidade de impressão de geometrias complexas são duas das principais características do fabrico aditivo. Esta técnica consiste na impressão de objectos a partir da deposição de variados materiais em camadas, sendo cada vez mais utilizada no mercado. A produção deve ser rápida com o uso de tecnologias de fabrico aditivo como *Fused Deposition Method* (FDM), *Selective Laser Melting* (SLM) e *Selective Laser Sintering* (SLS). À medida que as necessidades do consumidor se vão alterando, o desafio de aumentar a individualização de produtos e reduzir o seu tempo de produção para ser distribuído no mercado é cada vez maior [10].

Em 2018, a *Spirit Aerosystems* anunciou a construção de uma peça em titânio para um Boeing 787 a partir de fabrico aditivo [21]. Este é um exemplo de inovação e de variedade

que esta tecnologia pode ter, sendo neste caso utilizada para a construção de uma peça, um acessório de *backup* para o trinco da porta de acesso.

3.1.9. *Augmented Reality* – Realidade Aumentada

Sistemas baseados em realidade aumentada suportam uma grande variedade de serviços, como a medicina e a educação. A indústria pode usar a realidade aumentada de forma a oferecer aos trabalhadores informação em tempo real para melhorar a tomada de decisão e procedimentos de trabalho. É possível, por exemplo, ter instruções de montagem enviadas via *smartphone* para desenvolvimento de protótipos até à utilização de óculos de VR (*Virtual Reality*) para a gestão e operação de determinadas máquinas [10].

A *AugView*, por exemplo, é composta por um *software* GIS (*Geographic Information System)* móvel, que permite aos seus utilizadores ver e editar os dados dos seus activos no campo e também por uma aplicação de realidade aumentada, permitindo aos seus utilizadores visualizar objectos subterrâneos. Neste caso, a tecnologia de realidade aumentada é usada para exibir um modelo 3D do activo na aplicação, através da câmara, para que o utilizador consiga ver objectos no seu dispositivo que possam estar ocultos, como por exemplo canalizações, tubos subterrâneos, etc [22].

Figura 6 - Aplicação de Realidade Aumentada da AugView.

Capítulo 4 – *Machine Learning*

Nos dias que correm, *Machine Learning* é um termo bastante usado e falado mundialmente e é usado em muitas aplicações diárias, como por exemplo no motor de pesquisa da *Google*, recomendações de produtos em lojas online, filtros de *spam* para *e-mails*, entre outros.

Neste capítulo, pretende-se apresentar alguns dos conceitos base de *Machine Learning* e também explicar de forma sucinta como se criam algoritmos de aprendizagem e como é que funcionam.

4.1. "*Machine Learning*"

O *Machine Learning* é um campo de estudo que ensina aos computadores uma habilidade que é natural aos humanos e animais – aprender através da experiência, sem terem de ser explicitamente programados. Os algoritmos de *Machine Learning* usam métodos computacionais para recolher informação directamente dos dados sem depender numa equação pré-determinada como modelo. Estes algoritmos vão-se adaptando e melhorando o seu desempenho à medida que o número de amostras disponíveis para recolher aumenta.

Os algoritmos de *Machine Learning* encontram padrões naturais nos dados que geram conhecimento e ajudam a calcular previsões e tomar melhores decisões. Estes algoritmos são utilizados todos os dias para tomar decisões cruciais em diagnósticos médicos, trocas de acções, previsões de gastos energéticos, entre outros. A maioria dos *sites* de comunicação dependem destes algoritmos para filtrar milhões de opções e oferecer ao utilizador recomendações de músicas ou filmes. Já os *sites* de compras, por exemplo, usam estes algoritmos de forma a ganhar informação sobre o comportamento de compra dos seus clientes [23].

O *Machine Learning* é basicamente um ramo dentro do campo da inteligência artificial. Este termo foi criado por Arthur Samuel, por volta de 1950, que criou o primeiro programa capaz de aprender e jogar damas. Neste caso, o processo de "aprendizagem" correspondia a actualizar incrementalmente a base de dados com movimentos (posições

das damas) e o seu resultado, de acordo com a probabilidade de poder a vir ganhar ou perder o jogo [24].

Os termos *Machine Learning*, *Deep Learning* e *Artificial Inteligence* (AI) são frequentemente descritos de forma semelhante, apesar de não terem o mesmo significado. Na figura 7 é ilustrada a distinção entre os três termos. É possível observar que que a inteligência artificial cobre todos as técnicas de aprendizagem, incluindo a regressão, classificação, e tarefas de *clustering* e cognitivas. A inteligência artificial pode ser definida de modo amplo como a ciência capaz de mimetizar tarefas humanas.

Dentro do campo da AI está o *Machine Learning* com a sua larga variedade de algoritmos, desde *Support Vector Machines*, *K-Means Clustering*, *Random Forests* e muitos mais que foram desenvolvidos nas últimas décadas. De facto, o *Machine Learning* é um ramo da estatística onde os algoritmos aprendem através dos dados à medida que são inseridos no sistema.

Finalmente, temos o *Deep Learning*, também conhecido como *Artificial Neural Networks* (ANN) uma vez que estes algoritmos são modelados na forma como o cérebro processa dados, ainda que de uma maneira simplificada [25].

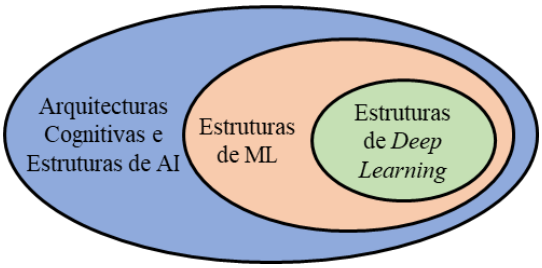

Figura 7 - Relação entre AI, ML e *Deep Learning* (adaptado de [2])

4.2. Exemplos de *Machine Learning*

Todas as indústrias serão afectadas pelo desenvolvimento e aplicação de algoritmos de inteligência artificial. A aplicação destes algoritmos em processos, produtos e serviços tornarão o negócio de qualquer empresa mais eficiente, inovador e rentável.

Nos serviços, não só haverá uma melhor previsão do que o que o cliente procura como será fornecido o produto certo baseado numa hiperindividualização. No retalho serão apresentadas cadeias de produtos mais sofisticadas e uma compreensão aprofundada

sobre as preferências do consumidor. No fabrico haverá uma evolução para a monitorização completa dos equipamentos em tempo real. Os algoritmos de *Machine Learning* podem ser usados para revelar problemas antes de acontecerem, optimizar a vida útil dos componentes e reduzir a necessidade de intervenção humana [26].

Estes algoritmos já estão a ser aplicados em quase todas as áreas, tomando como exemplo as seguintes [25]:

- Saúde;
- Transportes;
- Tecnologia;
- Energia.

4.2.1. Saúde

A inteligência artificial tem sido utilizada em bastantes subáreas de saúde, incluindo genética, descoberta de medicamentos, cura para o cancro, entre outras.

Olhando exclusivamente para a genética clínica, estes algoritmos são essenciais uma vez que têm a capacidade de analisar bases de dados extremamente grandes e encontrar padrões dentro da mesma. O genoma humano tem aproximadamente 4 biliões de pares de bases. Se se pensar em analisar centenas ou milhares de genomas, chega-se facilmente a bases de dados na ordem dos *petabytes*.

Felizmente, com a utilização de algoritmos de *Deep Learning* é possível analisar estas bases de dados em intervalos de tempo realistas – dias em vez de meses como seria de esperar há poucos anos. Isto faz com que o tempo e custos sejam efectivos na aplicação desta metodologia no campo da genética. Os resultados destes estudos são usados para encontrar a cura contra o cancro e outras doenças como Alzheimer e Parkinson, assim como para acelerar a descoberta de medicamentos para doenças genéticas (como esclerose lateral amiotrófica) e mentais (como esquizofrenia) que ainda afectam a humanidade.

4.2.2. Transportes

São várias as áreas e sistemas que já utilizam a AI, como carros autónomos, optimização de rotas, cidades inteligentes, voos, embarques, entre outros.

Todos os anos, cerca de 1.25 milhões de pessoas morre na estrada. É expectável que os veículos autónomos diminuam este número significativamente, na ordem dos 99% [25]. Estes veículos não só tornam as estradas mais seguras como também abrem a porta para novos serviços. Novos tipos de carros serão inventados assemelhando-se a escritórios, salas de estar ou mesmo quartos de hotéis sobre rodas. Os utilizadores apenas terão de solicitar o tipo de veículo que desejam baseado no destino e actividades que planearam para o caminho.

Os algoritmos de *Machine Learning* são usados para permitir que o veículo navegue segura e inteligentemente pelo ambiente de condução, prevendo os movimentos e evitando colisões com objectos, pessoas, animais e outros carros. Estes veículos vêm equipados com sensores de radar, infravermelhos, ultra-sons, microfones e câmaras que são usados para calcular distâncias, velocidades e tipo de objectos que circundam à sua volta, assim como prever o movimento através do espaço e tempo.

4.2.3. Tecnologia

Nesta área, a inteligência artificial tem sido bastante utilizada principalmente no desenvolvimento de *software*. Muitos investigadores sugerem que o foco principal da tecnologia é tornar a vida humana mais fácil e agradável automatizando tarefas que os humanos acham aborrecidas e repetitivas ou que simplesmente os impedem de fazer aquilo que realmente gostam. Alguns tipos de programação podem encaixar-se nesta categoria e automatizar as tarefas banais no desenvolvimento de *software* deixaria os programadores mais contentes e produtivos.

Para além disto, as empresas querem sempre reduzir custos e aumentar a velocidade e o rigor de qualquer processo de fluxo de trabalho, incluindo programação. É então inevitável a automação da maioria do ciclo de vida no desenvolvimento de *software*.

4.2.4. Energia

Também nas várias áreas da gestão de energia tem sido usado inteligência artificial, como por exemplo em medições inteligentes, na distribuição, eficiência energética e em centro de dados.

Em 2016, a *Google* anunciou que, usando algoritmos de *Machine Learning* da *Deep Mind*, a companhia reduziu a quantidade de energia usada para arrefecer os seus centros de dados em 40%. Usando um sistema de redes neuronais testado em diferentes cenários e parâmetros operacionais dentro de centro de dados, a *Google* conseguiu criar um fluxo de trabalho mais eficiente e adaptado às dinâmicas de um centro de dados.

Em Março de 2017, a *Deep Mind* anunciou que entrou em negociações com o governo do Reino Unido de forma a tornar a rede nacional de energia mais eficiente, aplicando os mesmos algoritmos à fonte nacional de alimentação. Demis Hassabis, CEO da *Deep Mind*, acredita que o uso de energia pode ser reduzido em 10% com a aplicação destes algoritmos. Com isto, o custo de energia para os consumidores iria reduzir bastante, assim como o aquecimento global, especialmente se esta medida fosse aplicada em todos os países do mundo.

4.3. Aprendizagem através de dados

A aprendizagem automática pode ser representada e modelada através de algoritmos para uma grande variedade de propósitos, como predição, classificação, detecção de anomalias, entre outros. Esta tarefa torna-se desafiante actualmente, uma vez que envolve processar e analisar enormes quantidades de dados e informação. O processo simplificado de criação de um algoritmo passa pela representação gráfica presente na figura 8.

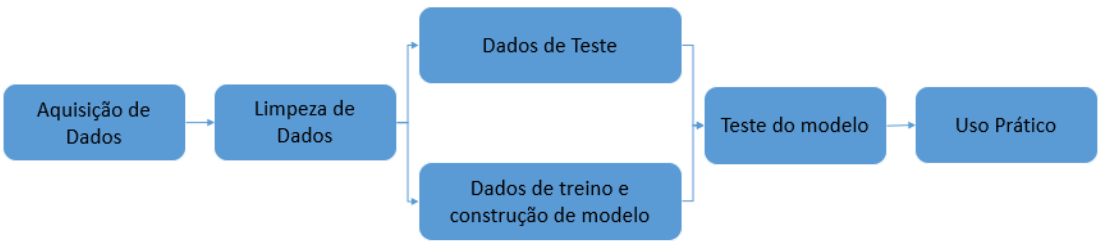

Figura 8 - Esquema de processo de *Machine Learning*

Basicamente, o que se chama de dados são observações de fenómenos do mundo real, dependendo do que se analisa. Por exemplo, os dados do mercado de acções podem envolver observações sobre cotações diárias, anúncios de ganhos de empresas individuais ou até artigos de opinião de especialistas. Já os dados biométricos pessoais podem incluir medições da frequência cardíaca, níveis de açúcar no sangue, pressão do sangue, entre outros. Existem vários exemplos de dados nos diferentes domínios [27].

Pela quantidade e diversidade de dados que existem, torna-se útil caracterizar os algoritmos de aprendizagem de acordo com o tipo de dados que sejam utilizados. Estes dados podem aparecer ou ser modelados de várias formas, como por exemplo vectores, matrizes, listas, imagens, vídeos, etc [28].

4.3.1 "*Feature Engineering*" – Engenharia das características dos dados

Uma característica (*feature)* é uma representação numérica de um conjunto de dados não tratados. Normalmente, os algoritmos de aprendizagem automática não interpretam dados não tratados e usam estas características para os representar. Há muitas maneiras de transformar dados não tratados em representações numéricas, o que os torna a todos muito semelhantes. As características são o "*input"* nos algoritmos de aprendizagem automática e derivam/dependem do tipo de data que queremos classificar. Estas características tão também relacionadas com o modelo. Alguns modelos são mais apropriados para certos tipos de características e vice-versa. ***Feature Engineering*** é o processo de formular as características mais apropriadas tendo em conta os dados, o modelo e a tarefa.

O número de características também é importante. Se não houver características com informação suficiente, o modelo será incapaz de executar a tarefa pretendida. Caso haja demasiadas características ou a maior parte irrelevante, o modelo será muito dispendioso e mais difícil de treinar. É necessário que haja um balanço certo de características [27].

Normalmente, o processo de escolha de características para um algoritmo pode dividir-se entre 2 a 3 fases. A 1ª fase destina-se a definir quais as características que possivelmente podem descrever os dados, tendo uma influência considerável nos dados de saída. A 2ª fase destina-se a transformar os valores das características de forma a pôr todas numa escala similar. Um método muito utilizado é o ***Feature Scaling***. Deste modo, todas as características terão a mesma importância na construção de um algoritmo. Por último, pode por vezes ser necessário diminuir, descartar e/ou transformar algumas destas

características caso se entenda que não têm grande impacto nos dados de saída, tornando desta forma o algoritmo mais "leve".

4.3.2. "*Dimensionality Reduction*" – Redução de Dimensão de dados

Para algoritmos mais pesados e com uma grande dimensão de dados, poderá haver a necessidade de reduzir esta dimensão de maneira a diminuir o seu custo computacional e uso de memória. Basicamente, há duas maneiras possíveis de o fazer – seleccionar um subconjunto de características existentes, sem as transformar, ou transformar as características existentes num subconjunto com menor dimensão, sem perder muita informação. Estes dois métodos são ***Feature Selection*** **e** ***Feature Extraction***, respectivamente. A escolha entre estes dois métodos está dependente do domínio de aplicação dos dados [29].

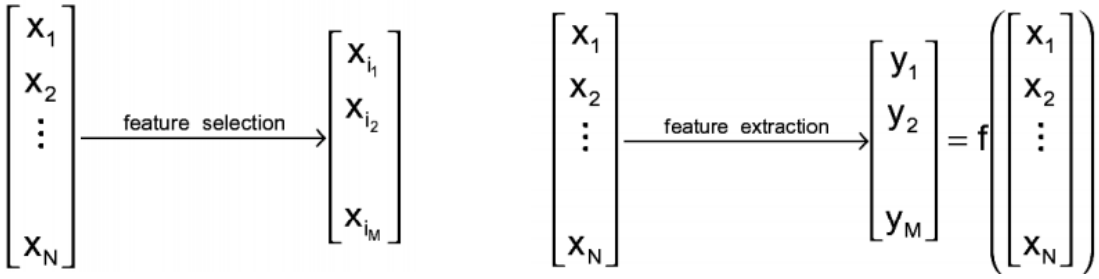

Figura 9 - Métodos de selecção de características

Grandes conjuntos de dados consistem em características que podem ser muitas vezes irrelevantes, traiçoeiros e/ou redundantes o que por vezes aumenta desnecessariamente o tamanho do conjunto e torna o processamento de dados mais complicado, dificultando o processo de aprendizagem. ***Feature Selection*** é o processo de selecção das melhores características entre todas que são úteis para qualificar classes.

Por outro lado, ***Feature Extraction*** consiste na transformação das características originais originando novas características mais significativas. Este método pode ser usado em conjuntos de dados muito pesados de maneira a reduzir a sua complexidade diminuindo a sua dimensão. O algoritmo mais popular e utilizado de ***Feature Extraction*** é conhecido como *Principle Component Analysis* (PCA). Este método trata-se de uma transformação linear que minimiza a redundância e maximiza a informação de um conjunto de dados "ruidoso".

4.3.3. *"Bias"* e *"Variance"* – Ajustamento e Variância

Os conceitos de ajustamento e variância podem ser pensados como pertencentes a uma escala que define a proximidade de um procedimento de aprendizagem ao seu conjunto de dados de treino.

O ajustamento é a diferença entre a predição média do nosso modelo e o valor correcto que se está a tentar prever. Modelos com um ajustamento elevado prestam pouca atenção ao conjunto de dados de treino e simplifica em demasia o modelo, levando sempre a um erro maior nos dados de treino e teste.

A variância é a variabilidade do modelo de predição para um determinado valor do nosso conjunto de dados. Modelos com alta variância prestam muita atenção aos dados de treino e não generalizam aos dados que não viram anteriormente. Como resultado, estes modelos têm um desempenho bastante positivo nos dados de treino, mas têm um erro muito maior nos dados de teste.

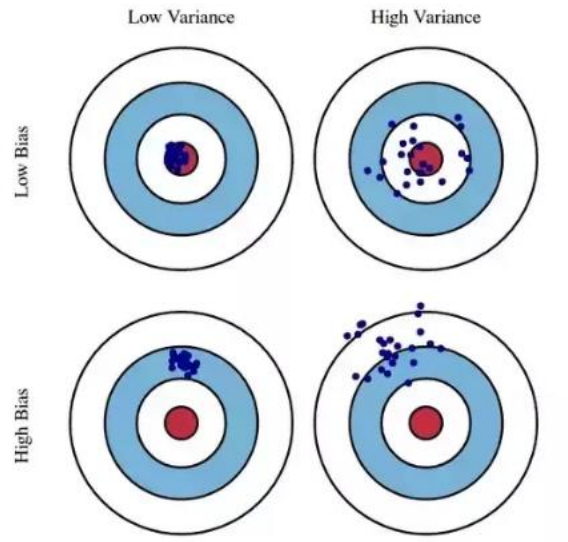

Figura 10 -Ilustração gráfica da *Bias* e *Variance [26].*

Relacionado com o ajustamento e variância, estão dois problemas associados aos dados: *Overfitting* e *Underfitting.* Estes problemas são originados quando as características escolhidas ou a selecção das mesmas não é feita da melhor maneira. O ***underfitting*** acontece quando um modelo é incapaz de se adaptar e entender o padrão subjacente dos dados. Estes modelos costumam ter elevado ajustamento e baixa variância. Este problema acontece maioritariamente quando a quantidade de dados para construir um modelo

preciso é limitada ou quando se tenta construir um modelo linear com um conjunto de dados não linear.

Já o ***overfitting*** acontece quando o modelo se adapta perfeitamente aos dados de treino, mas não consegue generalizar para os dados de teste e previsão de resultados. Este problema acontece maioritariamente quando um modelo é treinado em demasia num conjunto de dados ruidoso. Normalmente, estes modelos apresentam um baixo ajustamento e elevada variância [30].

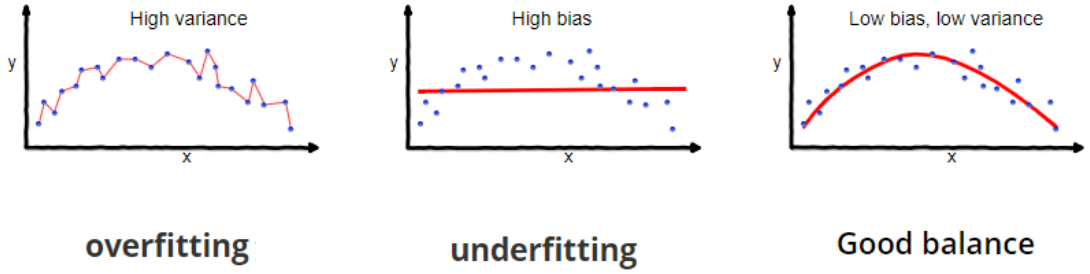

Figura 11 - Overfitting e Underfitting [30]

4.4. Tipos de *Machine Learning*

São utilizados dois tipos de técnicas para a aplicação de *Machine Learning – Supervised Learning*, que formula um modelo sobre dados de entrada e saída conhecidos para que possa prever futuras saídas e *Unsupervised Learning*, que encontra padrões escondidos ou estruturas intrínsecas nos dados de entrada [23]. Na figura 12 estão esquematizados os diferentes tipos de *Machine Learning*.

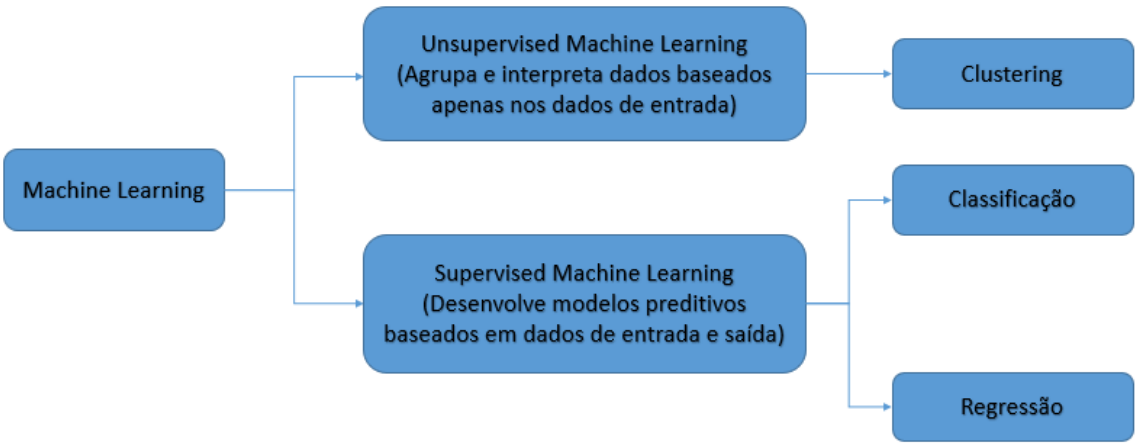

Figura 12 - Técnicas de *Machine Learning*

4.4.1. *Supervised Machine Learning* – Aprendizagem Automática Supervisionada

Um algoritmo de aprendizagem supervisionada baseia-se num conjunto conhecido de dados de entrada e de respostas a esses dados (saída) e ensaia um modelo que seja capaz de gerar previsões sensatas para uma resposta a novos dados [23].

Um exemplo de aprendizagem automática supervisionada é, por exemplo, a previsão de preços de casas. Tendo um conjunto de dados de entrada como tamanho da casa, localização, número de quartos, idade, etc. que correspondem a um conjunto de dados de saída (preço), é possível prever o preço de novas casas introduzindo os seus dados de entrada. Nesta situação, a previsão do preço de uma casa será feita comparando as suas características com as existentes na base de dados.

Basicamente, no *Machine Learning* supervisionado os dados consistem num conjunto de n características, X = {X_1, …, Xn} em que cada característica X_j é composta pelo seu próprio conjunto de possíveis valores x^i, com *m* vectores de características, $x^i = (x^1,\ldots,x^m)$ e uma resposta, $y = (y^1,\ldots,y^m)$, contendo um valor para cada vector [31] [32].

Um conjunto de dados típicos de um problema de *Machine Learning* supervisionado é o seguinte, presente na tabela 1.

Tabela 1 - Conjunto de dados típico de um problema de *Machine Learning* supervisionado

X_1	…	X_j	…	X_n	y
x_1^1	…	x_j^1	…	x_n^1	y^1
⋮		⋮		⋮	⋮
x_1^i	…	x_j^i	…	x_n^i	y^i
⋮		⋮		⋮	⋮
x_1^m	…	x_j^m	…	x_n^m	y^m

Este tipo de algoritmos usa técnicas de classificação e regressão para desenvolver modelos preditivos:

- **Técnicas de regressão** prevêem respostas contínuas. Por exemplo, mudanças na temperatura ou variações nos preços de casas. Ou seja, estes modelos de regressão tentam mapear os dados de entrada numa função contínua. Nestes casos, assume-se que a resposta da variável Y é quantitativa. Alguns tipos utilizados são, por exemplo:
    - Regressão Linear Simples;
    - Regressão Linear Múltipla;
    - Redes Neuronais Artificiais;
    - SVM (*Support Vector Machines*);
- **Técnicas de classificação** prevêem respostas distintas. Por exemplo, a distinção entre se um *email* é genuíno ou *spam*, se um tumor é maligno ou benigno. Ou seja, os modelos de classificação ordenam os dados de entrada em categorias. É recomendado se os dados podem ser categorizados, separados ou identificados em grupos ou classes específicas. Nestes casos, assume-se que a resposta da variável Y é qualitativa. Alguns tipos utilizados são, por exemplo:
    - Regressão Logística;
    - SVM (*Support Vector Machines*);
    - *Random Forests*.

4.4.2. *Unsupervised Machine Learning* – Aprendizagem automática não-supervisionada

A aprendizagem não-supervisionada encontra padrões escondidos ou estruturas intrínsecas nos dados. É usada para obter inferências de conjuntos de dados constituídos por dados de entrada sem respostas rotuladas.

Um exemplo de um modelo de aprendizagem não supervisionada pode ser algo do género: tem-se um conjunto de dados e pretende-se saber como separá-los em grupos significativos, como por exemplo uma pesquisa/questionário sobre a altura e peso das pessoas. Com a ajuda de algoritmos de aprendizagem não supervisionada pode-se

encontrar uma maneira de agrupar os dados em ***"clusters"*** significativos para os quais se poderá definir tamanhos de roupa. Neste caso, o modelo não tem uma resposta (Como por exemplo "Para a altura e peso introduzidos desta pessoa, devem ser classificados como um tamanho de calças pequeno"), tendo de descobri-la por si mesmo [33].

Enquanto que os algoritmos supervisionados tentam prever valores de um ou mais *outputs* ou respostas $Y = (Y^1,\ldots,Y^m)$ para um conjuntos de características, os algoritmos não-supervisionados têm como objectivo inferir directamente as propriedades de um conjunto de dados $(x_1,x_2,x_3,\ldots,x_n)$ sem a ajuda de uma resposta correctas. Enquanto que em *Machine Learning* supervisionado temos uma medida clara de sucesso, que pode ser mais tarde usada para comprar a eficácia de diferentes métodos sobre diferentes situações, no *Machine Learning* não-supervisionado não há qualquer medida de sucesso e torna-se difícil averiguar a validade das inferências realizadas, sendo mais uma matéria de opinião [31] [32].

Um conjunto de dados típicos de um problema de *Machine Learning* não-supervisionado é o seguinte, presente na tabela 2.

Tabela 2 - Conjunto de dados típico de um problema de *Machine Learning* não-supervisionado

X_1	…	X_j	…	X_n
x_1^1	…	x_1^1	…	x_1^1
⋮		⋮		⋮
x_1^i	…	x_j^i	…	x_j^n
⋮		⋮		⋮
x_1^m	…	x_1^1	…	x_n^m

A técnica mais usada de aprendizagem não-supervisionada é o *clustering*. É usado para explorar os dados de forma a encontrar padrões ou agrupamentos. As aplicações mais comuns para este tipo de algoritmos é a análise de sequência genética, pesquisa de mercado e reconhecimento de objectos [23].

4.5. Algoritmos de *Machine Learning*

4.5.1. Regressão Linear Múltipla

Este método é um dos métodos mais simples para implementar um algoritmo de *Supervised Machine Learning.*

A regressão linear simples é um bom método para prever a resposta baseada em apenas uma característica. No entanto, na prática haverá sempre mais do que uma característica que ajude na previsão [34].

A regressão linear é uma aproximação bastante eficaz para prever uma resposta quantitativa baseada num conjunto de características. Matematicamente, esta relação linear pode ser representada da seguinte maneira [32]:

$$h_\theta(x) = \theta_0 + \theta_1 X_1 + \theta_2 X_2 + \cdots + \theta_j X_j \tag{4.1}$$

Este modelo é composto por um conjunto específico de *j* dados de entrada (características x_j^i) e um conjunto de dados de saída conhecidos (y^i) para cada exemplo de treino *i*. Para além destes dados, é atribuído a cada característica um factor de escala, denominado coeficiente (θ_j) e ainda um coeficiente independente que dará alguma liberdade para ajustes (θ_0) [32]. Na prática, estes coeficientes são desconhecidos e é necessário estimá-los para se poder utilizar o modelo, utilizando os dados [34].

O objectivo principal da regressão linear é calcular vários valores dos coeficientes e obter os que melhor se encaixem nos dados e, desta forma, diminuam ao máximo o Erro Quadrático Médio (MSE – *Mean Squared Error*), normalmente chamado de *Cost Function* $J(\theta)$, entre a hipótese calculada e o valor de saída conhecido. Esta técnica é chamada de *Gradient Descent* [32], e é representada da seguinte maneira:

$$\theta_j = \begin{cases} \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i) & , \ j = 0 \\ \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^i) - y^i)\, x_j^i & , \ j > 0 \end{cases} \tag{4.2}$$

O *Gradient Descent* deve ser aplicado um determinado número de vezes, chamadas iterações, e parado quando os valores de θ não variarem significativamente por cada iteração. O α é chamado de *Learning Rate*, e representa a magnitude de cada iteração. Valores mais elevados de α significam que, por cada iteração, os valores de θ mudarão significativamente, levando o algoritmo aos valores óptimos de θ. No entanto, se α for

muito alto, o algoritmo pode nunca atingir o valor óptimo, saltando várias vezes. Usar valores de α mais baixos leva a que haja sempre uma aproximação desse valor, sem saltos. No entanto, torna a convergência mais lenta e o algoritmo mais pesado [27].

Este método é um dos mais aplicados quando os dados de saída são valores contínuos, mas é bastante sensível a *outliers*, afectando o seu desempenho.

4.5.2. Regressão Logística

Considera-se um conjunto de dados aleatório, em que a resposta pode pertencer a uma de duas categorias (Sim (1) ou Não (0), por exemplo). Em vez de modelar directamente a resposta Y, a regressão logística modela a probabilidade de Y pertencer a uma categoria particular [34].

Intuitivamente, não faz sentido que a hipótese $h_\theta(x)$ tenha valores maiores que 1 e menores que 0 quando se sabe que a resposta Y pertence entre 0 e 1 ($Y \in \{0; 1\}$). Para resolver este problema, reformula-se a expressão da hipótese $h_\theta(x)$ para satisfazer $0 \leq h_\theta(x) \leq 1$. Na regressão logística, a função utilizada que fornece *outputs* entre 0 e para todos os valores de X chama-se Função Logística (*Logistic Function* ou *Sigmoid Function)* [32]:

$$h_\theta(x) = g(\theta^T x) \tag{4.3}$$

$$z = \theta^T x \tag{4.4}$$

$$g(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+e^{-\theta_0+\theta_1 X_1+\theta_2 X_2+\cdots+\theta_j X_j}} \tag{4.5}$$

A função g(z) mapeia qualquer número real para o intervalo entre 0 e 1, sendo útil para transformar qualquer função numa função utilizável para modelos de classificação.

O resultado de $h_\theta(x)$ será a probabilidade de o *output* ser 1. Por exemplo, $h_\theta(x) = 0.7$ significaria que a probabilidade de um output ser 1 seria 70%. A probabilidade de a predição ser 0 é o complemento da probabilidade de ser 1, neste caso 30%.

Os valores de θ podem ser calculados com a expressão do *Gradient Descent* como utilizado em (4.2) mas com a expressão da hipótese da regressão logística.

4.5.3. Redes Neuronais Artificiais

Inspirado pelo cérebro humano, a rede neuronal consiste em redes de neurónios bastante conectados que relacionam os *inputs* com os *outputs* desejados. A rede é treinada modificando iterativamente os pontos fortes das conexões para que os *inputs* fornecidos sejam mapeados para a resposta correcta [23].

Este algoritmo é usado principalmente [23]:

- Para modelar sistemas não-lineares de grandes dimensões;
- Quando os dados ficam disponíveis incrementalmente e pretende-se actualizar constantemente o modelo;
- Quando pode existir mudanças não expectáveis nos dados de entrada;
- Quando a interpretação do modelo não é o factor chave.

Num nível elementar, os neurónios são basicamente unidades computacionais que recebem *inputs* (dendritos) como *inputs* eléctricos (chamadas "espinhas dendríticas") que são canalisados para os *outputs* (axónios). Neste modelo, os dendritos são as características de entrada $\boldsymbol{x_1} \cdots \boldsymbol{x_n}$, e o output é o resultado da função da hipótese. Nas redes neuronais, é usado a mesma função logística usada na classificação, $\frac{1}{1+e^{-z}}$, no entanto é normalmente chamada de função *sigmoid* de activação, e os parâmetros "theta" são chamados de pesos.

Este tipo de técnica de aprendizagem supervisionada é composto por três tipos de camada, denominadas *layers: input layer*, *hidden layers* e *output layer*. O primeiro *layer* é o ponto inicial e é onde introduzimos os nossos dados de entradas (*inputs*), o *layer* final é também chamado de *output layer* uma vez que o "neurónio" que possui é o dado de saída calculado por uma hipótese $h_\theta(x)$. Por fim, os *layers* intermédios, denominados de *hidden layers*, são criados pelo algoritmo RNA (ANN – *Artificial Neural Networks*) e não são observáveis nos dados de treino. Pode haver vários *hidden layers*, mas na maioria dos casos basta um para resolver a os problemas, normalmente. O número de unidades destes *layers* depende do tipo de dados. Cada unidade escondida receber os sinais de entrada, trabalha-os de diferentes maneiras e converte-os nas saídas correspondentes. Na figura 13, é possível observar um diagrama de uma rede neuronal com 2 *hidden layers*.

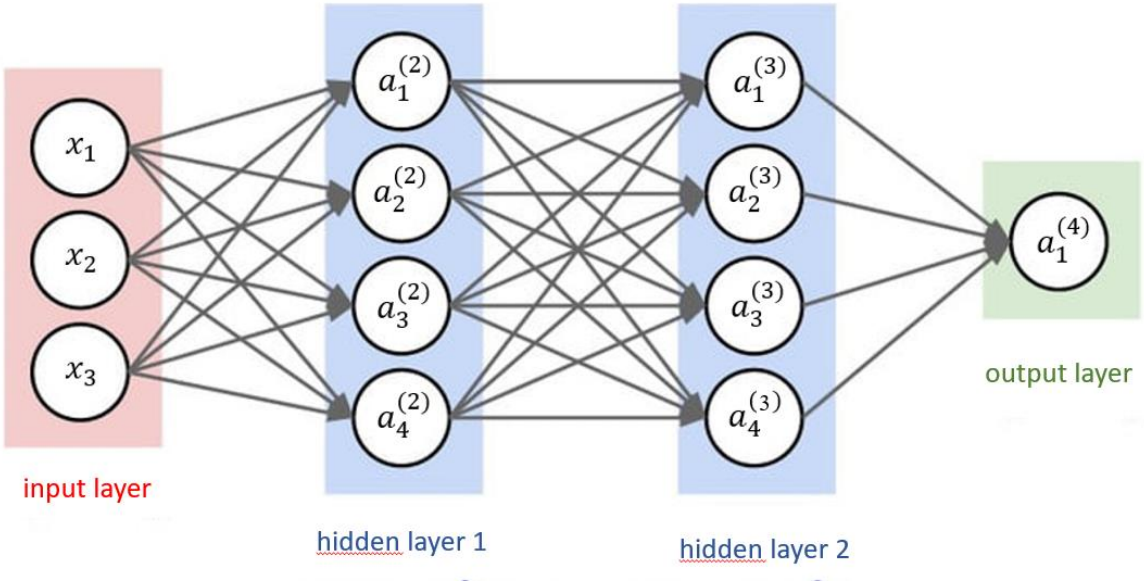

Figura 13 - Representação de uma rede neuronal artificial [35].

Aos nós dos *hidden layers*, pode-se chamar de unidades de activação. Podem ser representados da seguinte maneira:

$$a_i^{(j)} = \text{unidade de activação } i \text{ no } layer\ j \tag{4.6}$$

$$\Theta^{(j)} = \text{matriz de pesos que mapeam a função desde o } layer\ j \text{ até ao } layer\ j+1 \tag{4.7}$$

Uma representação simples dos dados de entrada, nós de activação e dados de saída, com apenas um *hidden layer* e com a representação da figura 13 seria:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \begin{bmatrix} a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \\ a_4^{(2)} \end{bmatrix} \rightarrow h_\theta(x) \tag{4.8}$$

O valor para cada nó de activação é calculado da seguinte maneira:

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)}x_0 + \Theta_{11}^{(1)}x_1 + \Theta_{12}^{(1)}x_2 + \Theta_{13}^{(1)}x_3\right) \tag{4.9}$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)}x_0 + \Theta_{21}^{(1)}x_1 + \Theta_{22}^{(1)}x_2 + \Theta_{23}^{(1)}x_3\right) \tag{4.10}$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)}x_0 + \Theta_{31}^{(1)}x_1 + \Theta_{32}^{(1)}x_2 + \Theta_{33}^{(1)}x_3\right) \tag{4.11}$$

$$a_4^{(2)} = g\left(\Theta_{40}^{(1)}x_0 + \Theta_{41}^{(1)}x_1 + \Theta_{42}^{(1)}x_2 + \Theta_{43}^{(1)}x_3\right) \tag{4.12}$$

$$h_\theta(x) = a_1^{(3)} = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)} + \Theta_{14}^{(2)} a_4^{(2)}\right) \quad (4.13)$$

Em cada *layer*, existe um neurónio "extra" chamado de *bias unit*, com o valor de 1, que não representam nenhuma unidade de activação real, mas que são utilizados para facilitar o algoritmo, como x0 e a0.

Simplisticamente falando, cada nó de activação/neurónio (a) está ligado por um peso (Θ) ao nó de activação do seguinte layer.

Neste exemplo, existem 3 unidades de *inputs*, 3 unidades de activação escondidas e $\Theta^{(1)}$ será uma matriz de dimensão de 4x4. Se uma rede tiver s_j unidades no *layer* j, s_{j+1} no *layer* j+1, então $\Theta^{(j)}$ será de uma dimensão $s_{j+1} \times (s_j + 1)$.

4.5.4. *Support Vector Machines* (SVM)

O algoritmo *Support Vector Machines*, normalmente abreviado para SVM, é utilizado tanto em tarefas de regressão e classificação, sendo maioritariamente utilizados nos problemas de classificação [36].

Este algoritmo é uma generalização de um classificador simples e intuitivo chamado *Maximal Margin Classifier*.

Basicamente, os dados são divididos por um hiperplano, sendo que novos dados são mapeados de acordo com a zona em que caírem. No entanto, para alguns conjuntos de dados, há diversas opções de hiperplanos possíveis, como é possível observar na figura 14.

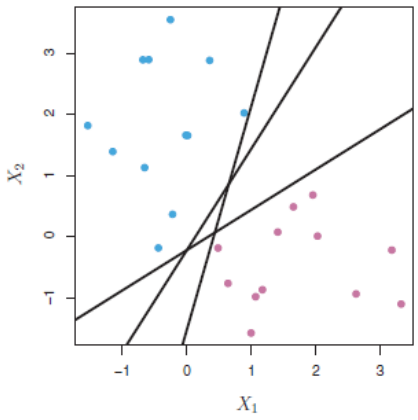

Figura 14 – Exemplo onde existem duas classes de observações que podem ser divididas por infinitos hiperplanos.

A escolha natural é o hiperplano que maximize a margem entre as duas classes, ou seja, utilizar um hiperplano que esteja equidistante de pontos das duas classes. Os pontos que criam estas margens são normalmente chamados de "*Support Vectors*" e têm este nome uma vez que caso fossem movidos, o hiperplano também se moveria, suportando-o. Na figura 15 as duas classes de observação, a azul e roxo, são divididas por uma linha preta (hiperplano) criada por dois pontos azuis e um ponto roxo, sendo que a distância destes pontos ao hiperplano é igual.

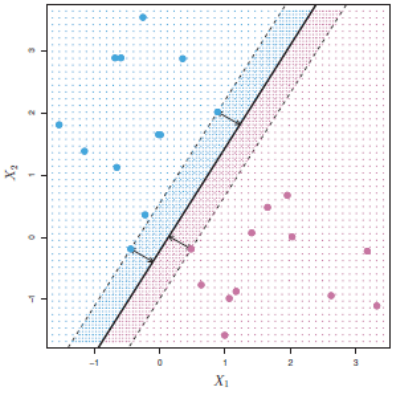

Figura 15 - Support Vectors, as suas margens e o hiperplano [34].

É neste conceito que assenta o *Maximal Margin Classifier*. No entanto, este modelo depende directamente dos *support vectors*, mas não do resto dos dados. Uma alteração em qualquer dos restantes dados não afectaria o hiperplano que divide as duas classes. Para além disto, a adição de uma observação individual poderia levar a uma mudança dramática de hiperplano, o que pode ser problemático uma vez que essas mudanças dramáticas sugerem que este modelo pode sofrer de *overfitting* nos dados de treino.

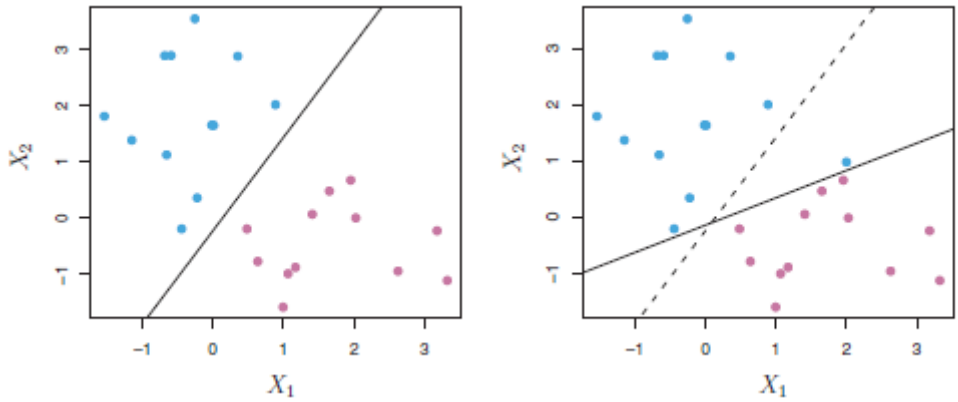

Figura 16 - A adição de uma nova observação, na figura da direita, leva a uma mudança dramática no hiperplano [34].

O *Support Vector Classifier* (SVC) é uma generalização do *Maximal Margin Classifier*. Este modelo permite que algumas observações estejam no lado incorrecto da margem, de forma a não ocorrer *overfitting* de dados.

É nestes dois modelos anteriores que o *Support Vector Machine* se baseia. O SVC é uma escolha natural para problemas de classificação de duas classes, se o limite entre as duas classes é linear. No entanto, na prática é muito comum encontrar limites de decisão não lineares.

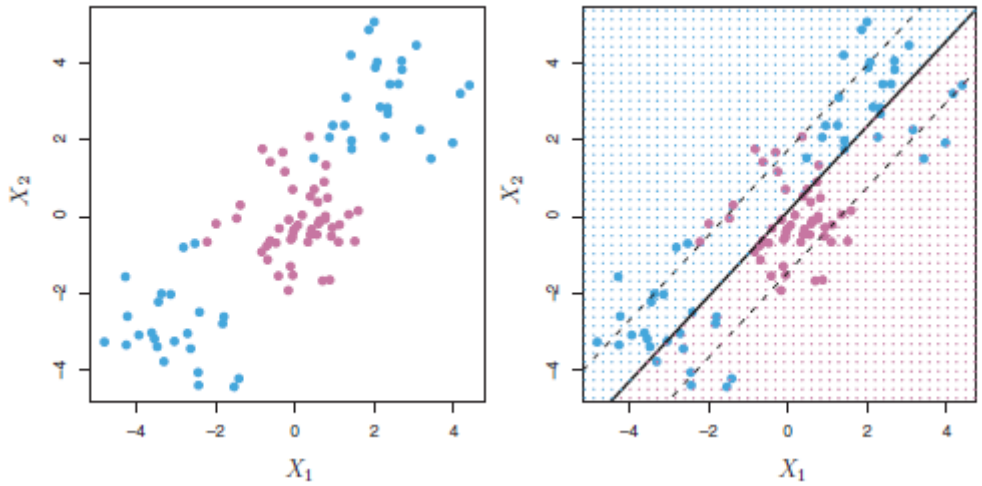

Figura 17 - Exemplo onde um modelo linear tem um desempenho bastante fraco.

Na figura 17, é possível observar que o *Support Vector Classifier* ou outro modelo linear não classificará os dados correctamente, sendo inútil neste caso. Neste tipo de situações, em que há grandes conjuntos de dados, usar mais características seria benéfico para separar os dados, mas por vezes é difícil extrai-las do conjunto de dados. O SVM (*Support Vector Machine)* tem um método específico para aumentar o número de características automaticamente, através de *kernels*. Um *kernel* não é mais que um método computacional bastante eficiente para garantir a ideia principal descrita neste capítulo – ser necessário aumentar o espaço das características de forma a acomodar o limite não-linear entre as classes. Existem vários tipos de *kernels*, sendo possível observar na figura 18 a utilização de um *kernel* polinomial à esquerda e de um *kernel* radial à direita.

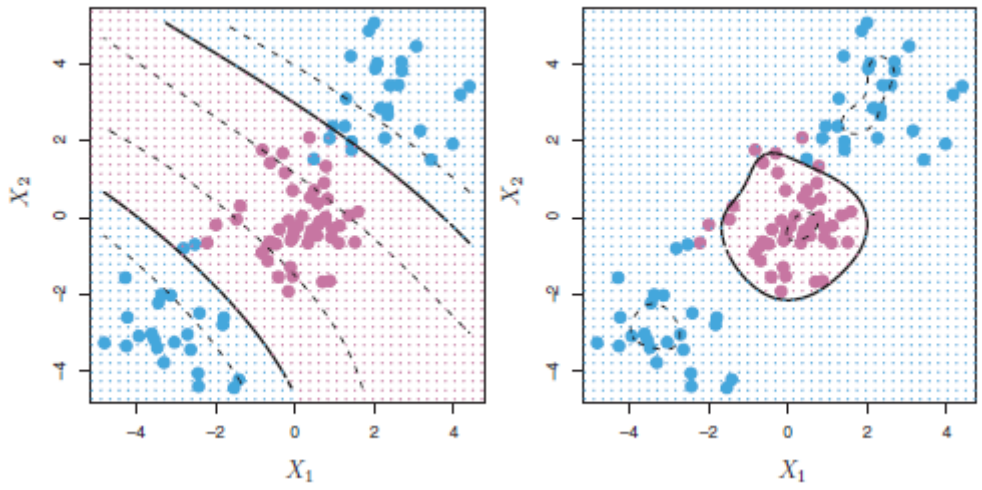

Figura 18 - Utilização de um *kernel* polinomial e um radial à esquerda e direita, respectivamente.

4.5.5. K-Means

O *K-Means* [32] [34] é um método de *clustering* onde se procura dividir as observações em K *clusters* distintos e sem sobreposição. Para executar este modelo, é necessário pré-definir um número de *clusters* K, sendo que depois o algoritmo *K-Means* vai atribuir cada "observação" apenas a um *cluster* K. O algoritmo tentará manter os *clusters* o mais afastado possível, de forma a que os dados de entrada atribuídos a cada *cluster* sejam o mais parecidos possível. Os dados de entrada ("observações") são atribuídos a um *cluster* de forma a que a soma da distância ao quadrado entre cada observação e o centroide do *cluster* (média aritmética de todos os pontos de entrada que pertencem ao *cluster*) seja a mínima possível.

Basicamente, o algoritmo *K-Means* funciona da seguinte maneira [37]:

1. Pré-definir o número de *clusters* K;

2. Designar aleatoriamente um número, de 1 até K, a cada uma das observações. Esta tarefa servirá para definir um *cluster* inicial para cada observação;

3. Continuar a iterar valores para os centroides até que as alterações verificadas sejam mínimas ou praticamente nulas. Por exemplo, deixar de haver mudanças de observações nos *clusters* definidos.

Na figura 19 é possível observar o progresso de um algoritmo *K-Means*.

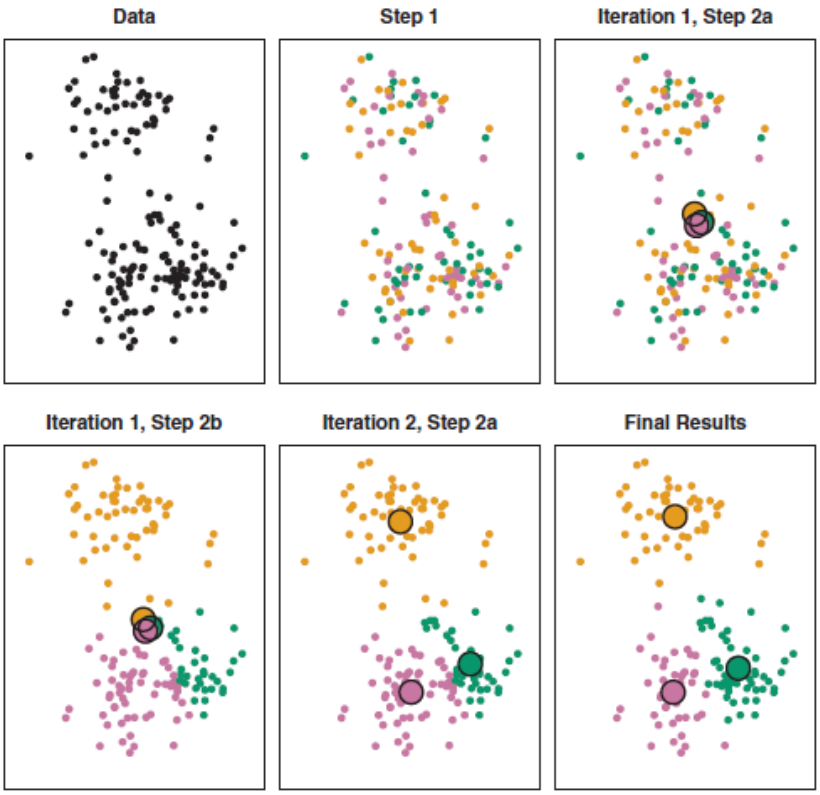

Figura 19 - Exemplo de progresso de um algoritmo *K-Means*, com K = 3 [34].

Nesta figura, é possível observar os passos na construção do algoritmo. O primeiro passo (2a) passa por definir médias diferentes para os centroides de cada *cluster* k, que são inicializadas aleatoriamente. De seguida, o algoritmo irá passar por cada observação individualmente, medido a distância entre esse ponto e cada um dos três centroides. Deste modo, irá agrupar a observação com o centroide mais próximo (em distância). Após ter associado cada observação ao centroide mais próximo, os valores de cada centroide (as médias) serão recalculados. O novo valor do centroide será a soma de todos os pontos que pertencem a esse centroide a dividir pelo número total de observações do grupo. Este passo é repetido até que o centroide não mude de valor cada vez que é recalculado [8].

Uma vez que o algoritmo encontra um local através de médias, os resultados obtidos dependerão da designação inicial de cada observação a um *cluster*, que é aleatória. Por esta razão, é necessário correr o algoritmo várias vezes com diferentes configurações iniciais, e seleccionar a melhor solução, ou seja, que a variação de cada observação com o centroide do seu *cluster* seja a mais pequena possível [34].

Capítulo 5 – Estudo de Caso

Neste capítulo pretende-se aplicar vários modelos de algoritmos de *Machine Learning*, de forma a comprovar que poderão ser úteis na gestão de activos físicos industriais. Neste caso, o estudo proposto passa pela determinação da vida útil restante (RUL – *Remaining Useful Life*) de uma frota de motores do mesmo tipo e perceber o quão perto estão do fim da sua vida, baseado nas suas condições de trabalho e leitura de sensores.

Para alcançar este objectivo, analisou-se um *dataset* existente no repositório de dados para prognóstico (PCoE – *Prognostics Center of Excellence*) da *NASA*. O repositório de dados para prognóstico é uma colecção de *datasets* que foram doados por várias universidades, agências e empresas e foca-se exclusivamente em conjunto de dados de prognósticos, que podem ser utilizados para o desenvolvimento de algoritmos de previsão [39].

O *software* utilizado no presente trabalho para tratar os dados e aplicar os algoritmos de *Machine Learning* foi o MATLAB®.

5.1. Objectivo

O *dataset* analisado consiste em várias séries temporais multivariadas, para vários motores. Isto é, para cada motor, cada linha representa o ciclo de vida em que o motor se encontra e os dados referentes à sua condição naquele número de ciclos realizados. Estes dados podem ser considerados como pertencentes a uma frota de motores do mesmo tipo.

Cada motor começa com diferentes graus de desgaste e produção que são desconhecidos para o utilizador. O desgaste e a diferença na produção de cada motor são considerados normais, pelo que não representam uma condição de falha. Ou seja, são todos semelhantes entre si. Cada conjunto de dados está dividido entre dados de treino e dados de teste. Os dados fornecidos da leitura de sensores são a partir de um determinado ciclo, não sabendo quantos ciclos o motor já percorreu.

Os motores estão a trabalhar normalmente no início de cada série temporal e desenvolvem uma falha em algum ponto durante o seu ciclo de vida. Nos dados de treino, a série

temporal termina quando há uma falha no motor. Nos dados de teste, a série temporal é indefinida.

O objectivo da análise deste *dataset* é prever o RUL antes da falha nos dados de teste, isto é, determinar o número de ciclos de vida que faltam após o último ciclo presente nos dados. Para isto, é fornecido o RUL real para estes dados de teste, para verificar a precisão e desvio do RUL previsto.

5.2. Tratamento de dados

Os dados foram descarregados em ficheiro de texto com 26 colunas, separadas com um espaço. Cada linha representa um conjunto de dados durante um único ciclo operacional e cada coluna é uma variável diferente. As colunas representam o seguinte:

1. Número da unidade;
2. Tempo, em ciclos;
3. Configuração operacional 1 (OP 1 – *Operational setting 1*);
4. OP 2;
5. OP 3;
6. Medição de sensor 1;
7. Medição de sensor 2;
8. Medição de sensor 3;

…

26. Medição de sensor 21.

Apesar de não estar definido no ficheiro de texto a que corresponde cada medição de sensor, é fornecida uma referência com informação referente às medições, apesar de não distinguir que sensor fez a medição respectiva [40]:

- Temperatura total à entrada do ventilador, em ºR;
- Temperatura total à saída do LPC (*Low Pressure Compressor*), em ºR;
- Temperatura total à saída do HPC (*High Pressure Compressor)*, em ºR;
- Temperatura total à saída do LPT (*Low Pressure Turbine*), em ºR;
- Pressão à entrada do ventilador, em psia;
- Pressão total no canal de *bypass*, em psia;

- Pressão total à saída do HPC, em psia;
- Velocidade do ventilador, em rpm;
- Velocidade do núcleo, em rpm;
- Relação de pressão no motor;
- Pressão estática à saída do HPC, em psia;
- Relação entre o fluxo de combustível e pressão à saída do HPC, em pps/psi;
- Velocidade do ventilador corrigida, em rpm;
- Velocidade do núcleo corrigida, em rpm;
- Taxa no *bypass*;
- Relação combustível-ar do queimador;
- Entalpia;
- Velocidade exigida do ventilador, em rpm;
- Velocidade exigida do ventilador corrigida, em rpm;
- Sangramento do líquido de refrigeração do HPT, em lbm/s;
- Sangramento do líquido de refrigeração do LPT, em lbm/s.

Os dados foram então carregados para o MATLAB® e foram atribuídos os respectivos títulos a cada coluna, de forma a ficar perceptível o que significa cada coluna, como representado na figura 20.

	1 id_motor	2 cycles	3 op1	4 op2	5 op3	6 sensor1	7 sensor2	8 sensor3	9 sensor4	10 sensor5	11 sensor6
1	1	1	-7.0000e-04	-4.0000e-04	100	518.6700	641.8200	1.5897e+03	1.4006e+03	14.6200	21.6100
2	1	2	0.0019	-3.0000e-04	100	518.6700	642.1500	1.5918e+03	1.4031e+03	14.6200	21.6100
3	1	3	-0.0043	3.0000e-04	100	518.6700	642.3500	1.5880e+03	1.4042e+03	14.6200	21.6100
4	1	4	7.0000e-04	0	100	518.6700	642.3500	1.5828e+03	1.4019e+03	14.6200	21.6100
5	1	5	-0.0019	-2.0000e-04	100	518.6700	642.3700	1.5829e+03	1.4062e+03	14.6200	21.6100
6	1	6	-0.0043	-1.0000e-04	100	518.6700	642.1000	1.5845e+03	1.3984e+03	14.6200	21.6100
7	1	7	1.0000e-03	1.0000e-04	100	518.6700	642.4800	1.5923e+03	1.3978e+03	14.6200	21.6100
8	1	8	-0.0034	3.0000e-04	100	518.6700	642.5600	1.5830e+03	1.4010e+03	14.6200	21.6100

Figura 20 - Primeiras linhas e colunas dos dados de treino.

Analisou-se e tratou-se inicialmente apenas os dados de treino, que serão os dados utilizados para treinar e avaliar a precisão do nosso algoritmo.

5.2.1. Uniformização dos dados

Sabe-se, à partida, que nos dados de treino cada motor tem representado os seus ciclos até ao momento em que falha. Por exemplo, o motor 1 tem linhas até ao ciclo 192, logo falha passado 192 ciclos e o motor 2 tem linhas até ao ciclo 287, logo teve uma vida

máxima de 287 ciclos, continuando até ao motor 100. Uma vez que o *dataset* é uma matriz de 26631x26, foi necessário utilizar duas funções para encontrar o último ciclo para cada motor. Estas funções têm como objectivo encontrar todos os valores únicos na coluna 1 (coluna dos IDs dos motores) e encontrar o valor máximo na coluna 2 (coluna do número de ciclos) para cada um desses valores únicos. Na figura 21 pode-se observar quais as funções implementadas e na figura 22 o gráfico do tempo de vida de cada motor.

```
[u,~,c] = unique(train_data{:,1});
MAX = accumarray(c,train_data{:,2},[],@max);
maxlifetrain = [u,MAX];
barh(maxlifetrain)
title('Engine Life Time');
xlabel('Cycles');
ylabel('id motor');
```

Figura 21 - Funções unique e accumarray

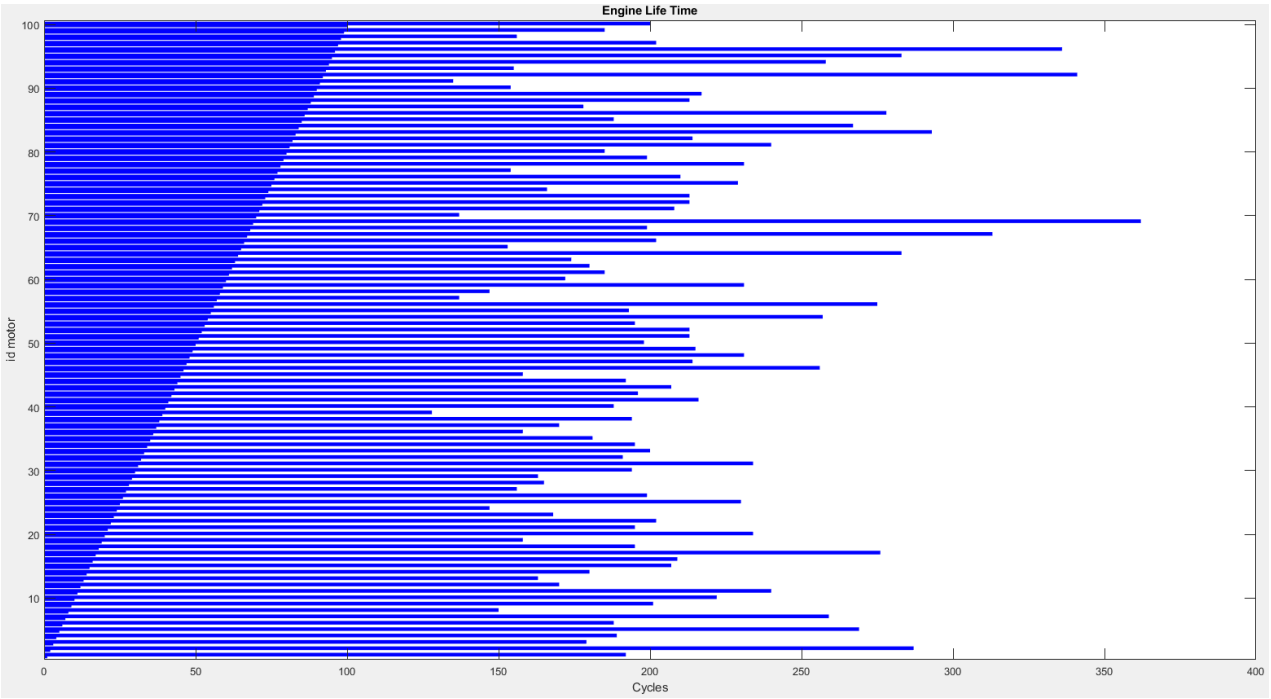

Figura 22 - Tempo de vida dos 100 motores presentes nos dados de treino.

Sabendo que cada linha do *dataset* representa um ciclo e sabendo o tempo de vida útil máximo de cada motor, é possível assumir que cada linha dos dados é um motor, criando uma nova coluna com o RUL para cada linha.

Por exemplo, sabemos que para o motor 1, no ciclo 1, o RUL é de 191 ciclos (uma vez que tem vida máxima de 192), no ciclo 2 é de 190, continuando até ao ciclo 192, em que o RUL é 0. Assim, é possível individualizar cada linha do motor de ID 1 e contar como se fossem 192 motores, todos com medições diferentes e com RULs diferentes. Sabendo

o tempo de vida de todos os IDs, é possível fazer o mesmo para todos os motores, tornando cada linha do *dataset* um motor individual. Para tal, foi utilizado um ciclo *for*, como representado na figura 23.

```
for numerolinha = 1:size(train_data,1)

    motor = traindata(numerolinha,1);

    train_data.RUL(numerolinha) = maxlifetrain(motor,2) - traindata(numerolinha,2);
end
```

Figura 23 - ciclo *for* para determinar o RUL de cada linha.

5.2.2. Eliminação de dados irrelevantes

De forma a tornar o processamento de dados e os algoritmos mais rápidos e leves, é necessário avaliar todas as características (*features*) do conjunto de dados e perceber se são relevantes para o estudo ou não. Por vezes existem características que não variam de caso para caso e que dessa forma não influenciam o resultado. Um método eficaz de verificar a existência de dados irrelevantes é a realização de um gráfico de dispersão de dados, que neste caso são os valores das medições dos sensores e dos *operational settings*. Nas figuras 24 e 25 é possível verificar os gráficos de dispersão obtidos.

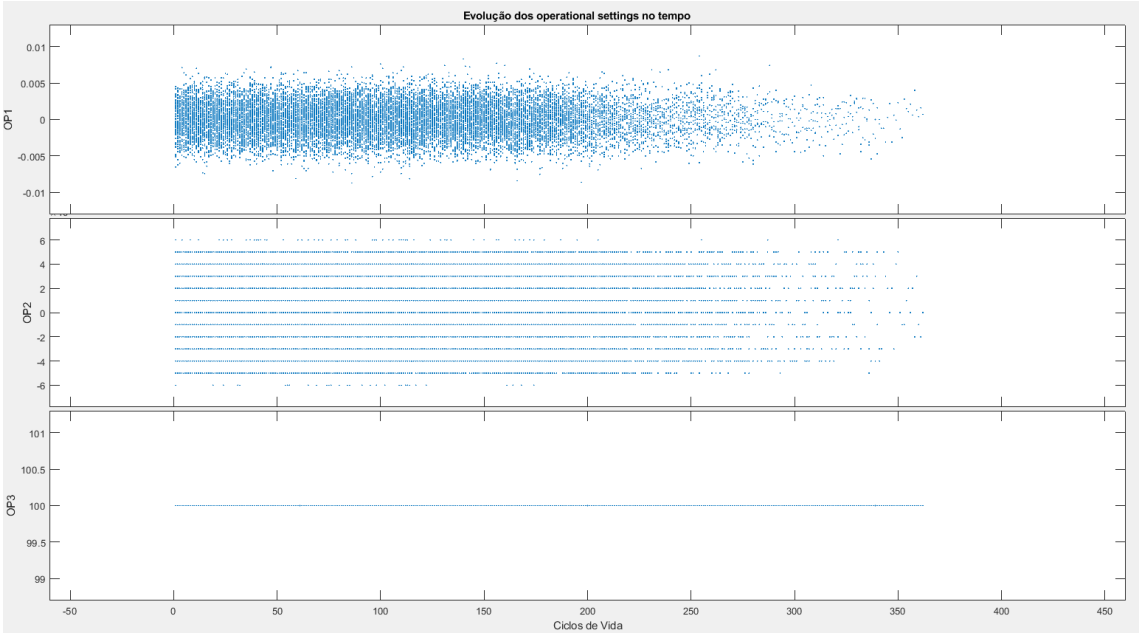

Figura 24 - *Scatter Plot* dos *operational settings*.

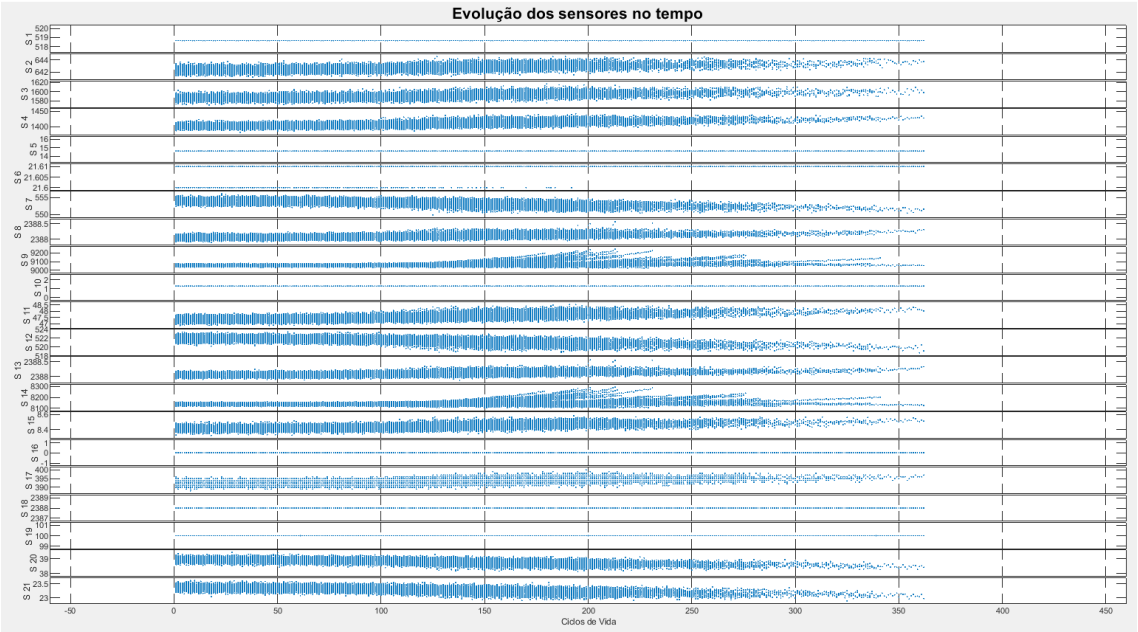

Figura 25 - *Scatter Plot* dos sensores do conjunto de dados.

Analisando o gráfico de dispersão, é possível verificar que dos 3 *operational settings*, o *operational setting* 3 é o único que apresenta um valor constante em todos os casos, pelo que não é relevante para o RUL. Em relação aos sensores, é possível verificar que os sensores 1, 5, 6, 10, 16, 18 e 19 também apresentam valores constantes para todos os casos, pelo que também não são relevantes para o estudo em causa.

Assim, podemos desprezar os valores destes sensores e retirá-los do nosso estudo, tornando o *dataset* mais limpo e apenas com variáveis relevantes para o estudo.

5.3. Previsão do tempo de vida útil restante (RUL)

De forma a estudar os dados de diferentes modos, aplicaram-se dois algoritmos de aprendizagem supervisionada, um de regressão e outro de classificação.

Aplicando um algoritmo de regressão, o objectivo passa por determinar o tempo de vida útil exacto para cada entrada de dados. Por exemplo, para uma linha de dados em que o RUL é 140 ciclos, pretende-se que o algoritmo consiga estimar, baseando-se nos dados fornecidos pelos sensores, que 140 é o número de ciclos que o motor ainda possui até falhar.

Aplicando um algoritmo de classificação, o objectivo passa por classificar cada linha a uma classe pré-definida. Por exemplo, definindo 2 classes em que classe 1 são todas as

linhas de dados com RUL > 100 e classe 2 são todas as linhas de dados com RUL < 100, pretende-se que o algoritmo consiga estimar que para uma linha de dados em que o RUL é 140 este pertença à classe 1.

O MATLAB® foi o *software* utilizado para construir e treinar estes algoritmos. A base de dados é bastante importante para treinar um algoritmo e o desempenho deste estará sempre dependente no tipo de conjunto de dados utilizado e da sua qualidade.

5.3.1. Aplicação de Algoritmo de Regressão – Regressão Linear

A regressão linear é um dos algoritmos de ***Machine Learning*** mais simples e mais fáceis de aplicar.

A primeira fase deste algoritmo, após tratamento de dados, passa por dividir os dados carregados em dados de entrada (X) e dados de saída (y). Os dados de entrada são os 14 sensores relevantes para o estudo e o *output* é o RUL. Quando se trata de conjunto de dados em que os valores de cada *feature* podem variar bastante entre si, torna-se necessário normalizá-los. De forma a normalizar todos os *features* (que neste caso são os sensores), utilizou-se o ***Feature Normalization*** (figura 26). Este modelo devolve uma versão normalizada da matriz X, onde o valor médio é 0 e o desvio-padrão é 1. É bastante utilizado como um passo de pré-processamento para algoritmos de aprendizagem, quando se tem valores na ordem de grandezas das centenas e milhares e na ordem das casas decimais, o que é o caso.

```
function [X_norm, mu, sigma] = featureNormalize(X)

X_norm = X;
mu = zeros(1, size(X, 2));
sigma = zeros(1, size(X, 2));

mu = mu + mean(X);
sigma = sigma + std(X);

m = length(X);
n = size(X,2);
for i = 1:m
    for j = 1:n
X_norm(i,j) = (X(i,j) - mu(1,j))/sigma(1,j);
    end
end

end
```

Figura 26 - Função para o cálculo do Feature Normalization

De seguida, foi aplicado o *Gradient Descent*. Esta técnica, descrita no capítulo 4, irá calcular os valores de cada Theta para qual a *Cost Function* é a menor possível. Neste caso, escolheu-se um *Learning Rate* de $\alpha = 0{,}2$ e 100 iterações.

Após o cálculo dos thetas, calculou-se o ytheta, que representa os dados de saída estimados, utilizando os thetas calculados e de seguida o RMSE (*Root Mean Squared Error* – Raíz do Erro Quadrático Médio) e o MAE (*Mean Absolute Error* – Erro Absoluto Médio). Estes erros não têm uma gama de valores aceitáveis ou não, dependem dos dados que são alvos de estudo. Neste caso em particular, o Erro Absoluto Médio deu um valor de 34.1154, o que significa que existe um erro médio na previsão do RUL de 34 ciclos, arredondadamente. Caso se trate de um motor em início de vida, prever um RUL de 180 ciclos para 150 ciclos reais pode não ser problemático, mas se se tratar de um motor com um RUL de 10 ciclos e for previsto um RUL de 40, já se torna problemático.

Tendo em conta as conclusões descritas no parágrafo anterior, consideraram-se fracos os resultados deste algoritmo. Estes resultados podem dever-se ao facto dos dados se encontrarem bastante dispersos nos primeiros ciclos ou também ao facto deste algoritmo ser bastante simples e linear.

5.3.2. Aplicação de Algoritmo de Classificação – *Support Vector Machine*

O SVM é um algoritmo de ***Machine Learning*** complexo e é normalmente utilizado para problemas de classificação. Como referido no capítulo 4, este algoritmo utiliza hiperplanos para delinear as várias classes, podendo estes serem lineares ou não-lineares.

A primeira fase na aplicação deste algoritmo, após tratamentos dos dados, foi definir quantas classes queremos e quais as condições que as definem. Neste caso, as classes foram divididas tendo em conta o tempo de vida útil restante do motor, uma vez que se pretende prever o RUL a fim de se poder intervir, executar uma tarefa de manutenção ou substituir um motor antes da falha. Em contexto industrial, estas classes não poderiam ser definidas apenas pelos dados obtidos dos sensores e seriam definidas por uma equipa de manutenção que, sabendo que a partir de um certo número de ciclos os motores tendem a piorar o seu estado de forma exponencial e a partir de que RUL é oportuno realizar manutenção, definiriam várias classes.

Neste estudo de caso, as classes foram definidas arbitrariamente para estudo, como se pode ver na figura 27.

```
for numerolinha = 1:size(train_data,1)

    if train_data.RUL(numerolinha) <= 50
        train_data.CLASS(numerolinha) = 2;
    elseif train_data.RUL(numerolinha) > 50 && train_data.RUL(numerolinha) < 125
        train_data.CLASS(numerolinha) = 1;
    else
        train_data.CLASS(numerolinha) = 0;
    end
end
```

Figura 27 - Definição das classes para aplicação de algoritmo.

Descrevendo cada classe:

- Classe 2 – Necessidade de manutenção urgente – motor em fim de vida (<50 ciclos até falha);
- Classe 1 – Necessidade de manutenção média – motor entre 50 a 125 ciclos até falha;
- Classe 0 – Longo período temporal até necessitar de manutenção – motor com mais de 125 ciclos até falha.

Tendo as classes definidas, recorreu-se ao MATLAB® para aplicar um SVM, uma vez que se trata de uma ferramenta bastante poderosa e utilizada na área, tendo os seus principais algoritmos de inteligência artificial optimizados. Através de uma aplicação denominada "*Classification Learner"*, é possível aplicar um algoritmo de classificação à escolha, definindo os seus parâmetros e após aplicação, é possível gerar um código função para aplicação futura, utilizando o mesmo algoritmo para trabalhar novas entradas de dados e prever resultados.

Ao abrir a aplicação *Classification Learner*, é necessário criar uma sessão e o primeiro passo passa por escolher os dados que queremos analisar. Ao escolher a matriz com os dados tratados, define-se quais colunas são características (*features)* e qual coluna é a resposta, como demonstrado na figura 28. Como resposta, definiu-se a coluna que apresenta as classes e como *features* definiram-se os 14 sensores que não apresentavam dados constantes durante os ciclos de vida.

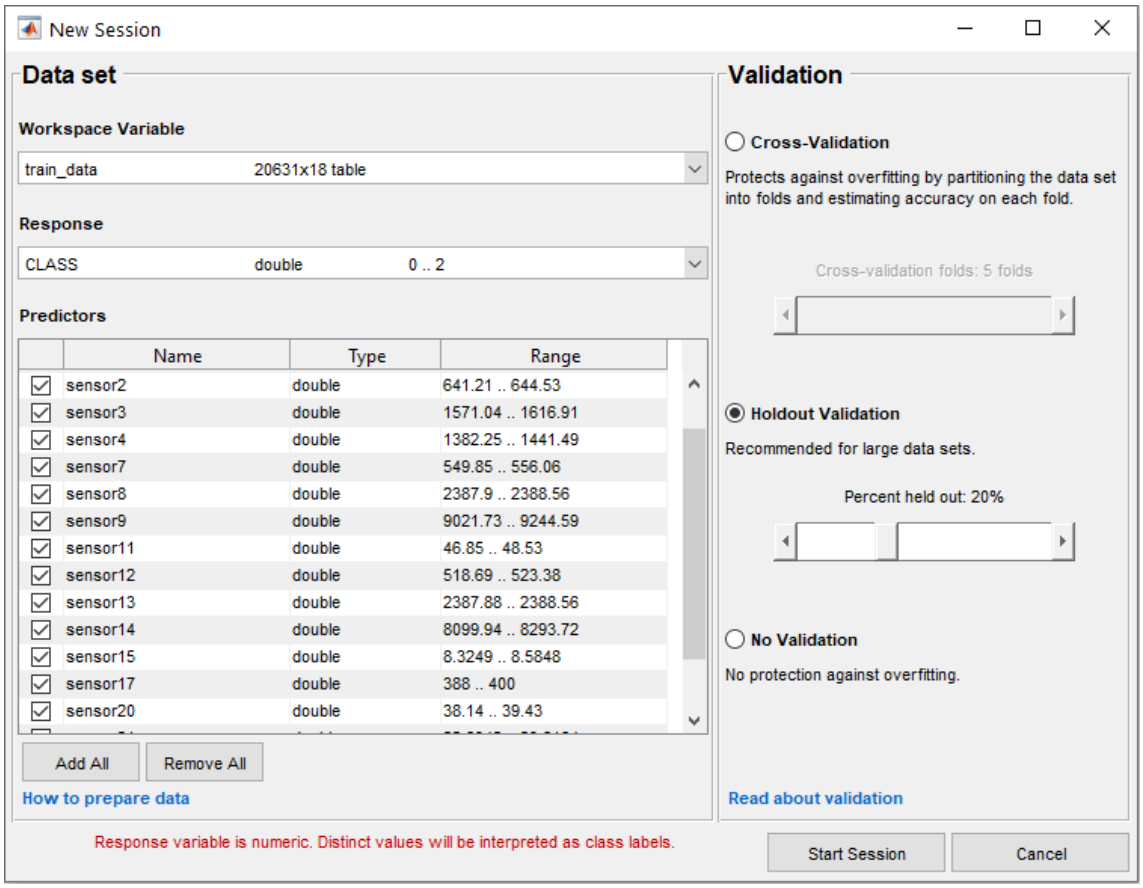

Figura 28 - Definição de nova sessão no *Classification Learner*.

Antes de se iniciar a sessão, é necessário definir o tipo de validação que queremos. De forma a proteger-se o modelo de *overfitting*, escolheu-se o método *Holdout Validation*, que nada mais é do que dividir os dados de treino em dois grupos. Neste caso e como tipicamente se procede, dividiu-se os dados em 80% para dados de treino e 20% para dados de teste.

Após iniciar sessão, é apresentado um novo ecrã onde se pode escolher o tipo de algoritmo que queremos utilizar e, após escolha do mesmo, treiná-lo, vendo qual é a sua precisão. Este processo é repetido em vários tipos de algoritmos, treinando-os e vendo qual apresenta os melhores resultados para o caso em estudo. No entanto, e uma vez que utilizaremos um algoritmo "*multi-class*", em que o resultado não é binário (0 ou 1), a precisão pode não ser o factor decisivo na escolha do algoritmo. Por exemplo, um algoritmo pode ter uma precisão global de 85% e prever correctamente as classes 0 e 1 em 90% dos casos e apenas prever correctamente a classe 2 em 75% dos casos, enquanto que outro algoritmo pode ter uma precisão global de 75% em que prevê correctamente em 67,5% dos casos as classes 0 e 1 e 90% a classe 2. Neste caso de estudo, a previsão correcta da classe 2 torna-se significativamente mais importantes que as restantes, uma

vez que significa que um motor está a precisar urgentemente de manutenção e a possibilidade de o algoritmo prever classe 0 ou 1 num motor que na realidade pertence à classe 2 pode significar a sua utilização numa fase de vida final, onde está próximo de falhar, o que seria catastrófico. Prever classe 1 quando um motor pertence à classe 0 não é problemático, e vice-versa, uma vez que não estão em fase final de vida. No entanto, é expectável que à medida que o algoritmo é alimentado com mais dados, a sua precisão global aumente para qualquer classe. Após serem analisados vários algoritmos, escolheu-se o *Fine Gaussian SVM*, um algoritmo SVM baseado em *kernels* gaussianos, para *multi-class*. Na figura 29, é possível ver os algoritmos treinados e as suas precisões gerais, assim como a *Confusion Matrix* do modelo escolhido.

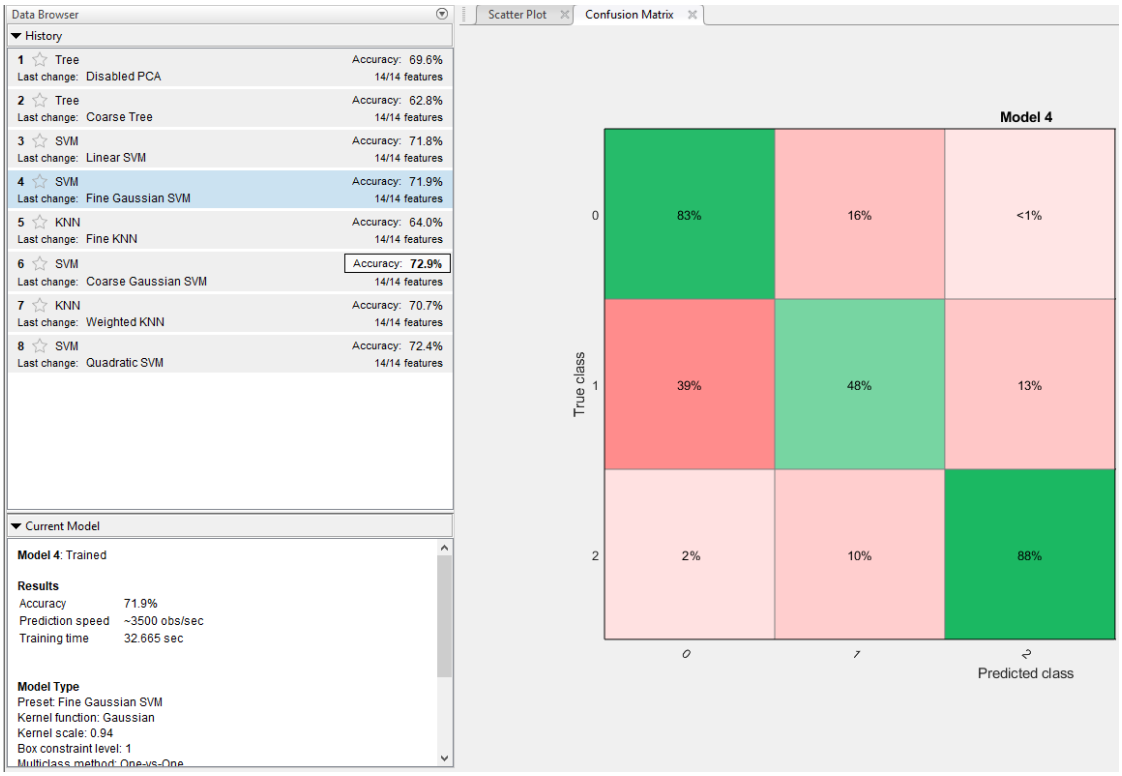

Figura 29 - *Accuracy* dos modelos treinados e *Confusion Matrix* do modelo escolhido.

É precisamente na *Confusion Matrix* que assenta a escolha do modelo. Esta matriz é bastante eficaz e utilizada para perceber qual foi o desempenho do algoritmo em cada classe e permite assim identificar em que classes o algoritmo teve um fraco desempenho. Este algoritmo foi escolhido precisamente por ser o que apresenta melhor desempenho na previsão da classe 2, que neste estudo é a classe crítica. No entanto, 88% pode ainda ser um valor baixo quando se trata da previsão do RUL de um motor. Para aumentar este valor, é possível elaborar uma matriz de custo de forma a dar prioridade a alguns erros.

Basta de seguida acrescentar ao código um parâmetro chamado "*Cost*", que irá ler a matriz e permitir que ele preveja melhor os dados de determinada categoria.

```
>> CostMatrix = ones(3)-eye(3);
CostMatrix(3,1:2) = 10;
CostMatrix(2,3) = 3;
CostMatrix(1,3) = 3;
disp(CostMatrix)
     0     1     3
     1     0     3
    10    10     0
```

Figura 30 - Matriz de Custo.

Como se pode observar na figura 30, deu-se uma importância maior na linha 3, colunas 1 e 2. Estas duas entradas representam as previsões de classe 0 e 1, quando a classe real é 2. Estes são os resultados que se pretende optimizar, uma vez que se está a prever um RUL acima de 50 ciclos quando o motor já se encontra abaixo de 50.

Após definir a matriz de custo, correu-se novamente o código já com este parâmetro introduzido e obteve-se a seguinte *Confusion Matrix*, apresentada na figura 31.

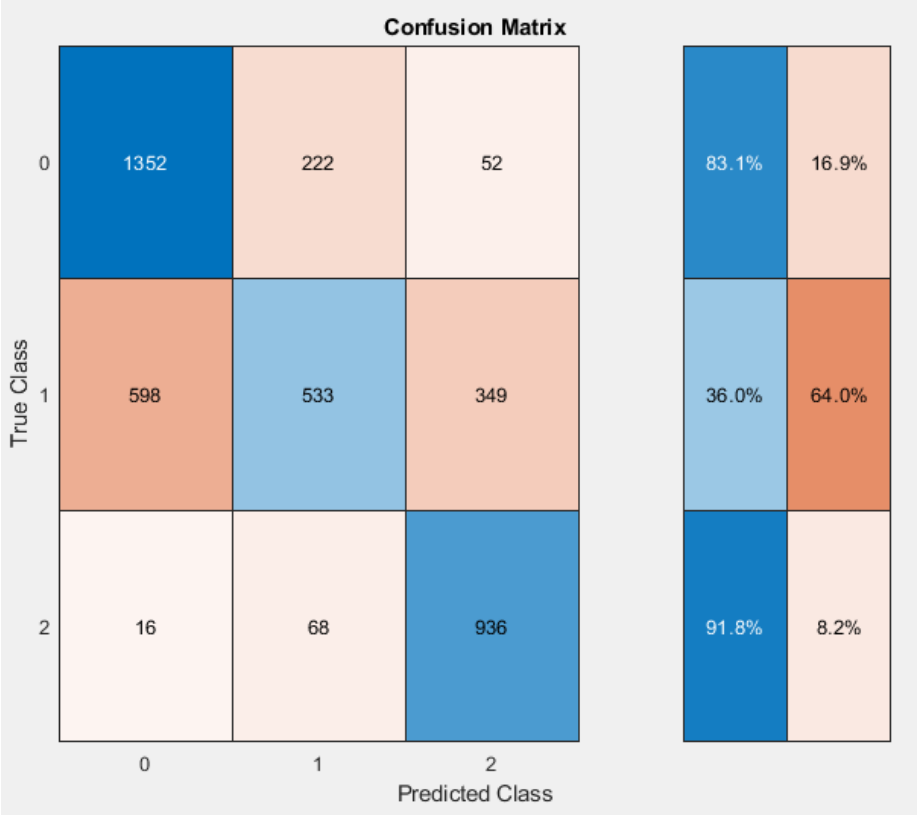

Figura 31 - *Confusion Matrix* após introdução de matriz de custo.

Apesar da precisão geral ter diminuído (de 72% para 68%), o algoritmo apresenta agora uma precisão de 92% na previsão de motores com RUL menor que 50 (classe 2). A classe

0 manteve a precisão e a classe 1 piorou, o que era expectável uma vez que se trata de uma classe intermédia com 2 limites. No entanto, e uma vez que a prioridade é estimar correctamente os motores que se encontrem na classe 2, considerou-se estes resultados bastante positivos. Ainda assim, e à medida que forem introduzidos mais dados para alimentar o algoritmo, é expectável que este produza melhores resultados.

Capítulo 6 – Conclusões

Este Trabalho Final de Mestrado propôs uma abordagem ao tema *Machine Learning* e à sua importância e aplicação nos últimos anos, em contexto industrial. Propôs também a aplicação de um algoritmo para gestão de activos físicos industriais.

Foi possível verificar a grande versatilidade de aplicações que o *Machine Learning* tem em todas as áreas e a forma como se pode revelar bastante útil no contexto industrial. A capacidade de os algoritmos aprenderem sozinhos e serem alimentados através de novos dados traz uma grande utilidade a este tipo de processo, uma vez que permite analisar e entender tendências que outrora demorariam bastante mais tempo a serem descobertas.

Após aquisição de conhecimentos nos vários tipos e aplicações de *Machine Learning*, aplicaram-se dois algoritmos numa base de dados de activos físicos de forma a prever o tempo de vida útil restante dos mesmos. Apesar de não haver muita informação sobre o conjunto de dados a ser utilizada para estudo, conseguiu-se estimar com uma precisão acima de 90% nos casos críticos, nos quais o motor apresenta um número de ciclos restantes abaixo de 50. Para estudo, é um valor bastante positivo e que pode ser aumentado, à medida que o algoritmo recebe mais dados.

Tendo em conta os objectivos inicialmente estabelecidos para este Trabalho Final de Mestrado, considera-se que o primeiro foi concluído, uma vez que foi feita uma introdução nas áreas onde se pretende aplicar conhecimentos de *Machine Learning*, mais propriamente na área de Manutenção, área de Gestão de Activos e em contexto industrial.

O segundo objectivo passava por contextualizar o tema propriamente dito. Este objectivo também foi cumprido, uma vez que foram dados exemplos de aplicação de algoritmos de *Machine Learning* em várias áreas, como Saúde, Energia ou Transportes e foram mencionados os vários tipos de *Machine Learning* existentes, sendo ainda descritos alguns algoritmos específicos de cada tipo.

O terceiro e último objectivo também foi cumprido. Apesar das limitações a nível de programação, foram aplicados dois algoritmos, um de regressão e outro de classificação, ambos de *Supervised Machine Learning*, a um conjunto de dados proveniente de um repositório de dados para estudo da NASA.

O algoritmo de regressão apresentou resultados fracos, uma vez que os dados possuem bastante dispersão inicialmente, mas conseguiu-se resultados que se consideraram bastante positivos para o algoritmo de classificação, uma vez que se conseguiu precisões acima de 90% na classe crítica, provando que este campo de estudo pode ser bastante importante e relevante na gestão de activos físicos.

O trabalho alcançado neste TFM representa uma mais-valia para o desenvolvimento da área de Manutenção e Gestão de Activos, uma vez que tanto este como estudos semelhantes promovem uma maior preparação para o actual e futuro contexto industrial que se vive. O desenvolvimento e aplicação de técnicas de *Machine Learning* no estudo de *datasets* deste género pode ser bastante benéfico e criar metodologias mais eficazes do que as que existem no que diz respeito, por exemplo, à detecção de falhas e avarias em equipamentos.

6.1. Trabalhos Futuros

Ainda que apresente alguns resultados interessantes, o estudo efectuado no âmbito deste Trabalho Final de Mestrado demonstra bastantes limitações e melhorias a realizar em trabalhos futuros.

Apesar dos objectivos referentes à temática do *Machine Learning* e a sua aplicação a um estudo de caso na área de Manutenção e/ou Gestão de Activos, na área industrial terem sido concluídos, com a existência de um maior conhecimento na área de programação poderia ter sido possível a aplicação de outros algoritmos mais eficientes, o que poderia ter resultado numa maior precisão geral.

Tratando-se também de um conjunto de dados utilizado para estudos, não há aquisição de novos dados para ir alimentando este algoritmo, o que também seria benéfico, uma vez que, adquirindo novos dados, o algoritmo tornar-se-ia também mais eficiente.

Referências

[1] European Committee for Standardization. (2017). EN 13306: Maintenance – "Maintenance terminology".

[2] Trojan, F., & Marçal, R. F. (2016). "Sorting maintenance types by multi-criteria analysis to clarify maintenance concepts in POM". In 27 th POMS conference, At Orlando-Florida-USA, pp. 1-10.

[3] Mobius Institute. (2005). Analista de Vibrações – Categoria II, ISO 18436-2 e SNT-TC-1A da ASNT.

[4] Accorsi, R., Manzini, R., Pascarella, P., Patella, M., & Sassi, S. (2017). "Data mining and machine learning for condition-based maintenance". Procedia Manufacturing, vol. 11, pp. 1153-1161.

[5] International Organization for Strandardization. (2014). NP-EN ISO 55000 – Asset Management – International Standard. International Organization for Standardization.

[6] (2018). "*Machine Learning em Gestão de Activos"* [Online]. Available: https://www.youtube.com/watch?v=_R4nJieIIGI [Accessed: 05-08-2019]

[7] Cipollini, F., Oneto, L., Coraddu, A., Murphy, A. J., & Anguita, D. (2018). "Condition-based maintenance of naval propulsion systems: Data analysis with minimal feedback". Reliability Engineering & System Safety, vol. 177, pp. 12-23.

[8] Ayo-Imoru, R. M., & Cilliers, A. C. (2018). "Continuous machine learning for abnormality identification to aid condition-based maintenance in nuclear power plant". Annals of Nuclear Energy, vol. 118, pp. 61-70.

[9] Ahn, B., Kim, J., & Choi, B. (2019). "Artificial intelligence-based machine learning considering flow and temperature of the pipeline for leak early detection using acoustic emission". Engineering Fracture Mechanics, vol. 210, pp. 381-392.

[10] Vaidya, S., Ambad, P., & Bhosle, S. (2018). "Industry 4.0–a glimpse". Procedia Manufacturing, vol. 20, pp. 233-238.

[11] Masoni, R., Ferrise, F., Bordegoni, M., Gattullo, M., Uva, A. E., Fiorentino, M., ... & Di Donato, M. (2017). “Supporting remote maintenance in industry 4.0 through augmented reality”. Procedia manufacturing, vol. 11, pp. 1296-1302.

[12] BigData Republic (2017). “Machine learning for predictive maintenance: where to start?”. [Online]. Available: https://medium.com/bigdatarepublic/machine-learning-for-predictive-maintenance-where-to-start-5f3b7586acfb [Accessed: 14-01-2020]

[13] Erboz, G. (2017). “How to Define Industry 4.0: The Main Pillars of Industry 4.0”. [Online]. Available: https://www.researchgate.net/publication/326557388_How_To_Define_Industry_40_Main_Pillars_Of_Industry_40 [Accessed: 23.12. 2019].

[14] Intelipaat (2016). “7 Big Data Examples: Applications of Big Data in Real Life”. [Online]. Available: https://intellipaat.com/blog/7-big-data-examples-application-of-big-data-in-real-life/ [Accessed: 03-03-2020].

[15] KUKA (2016). “LBR iiwa”. [Online]. Available: https://www.kuka.com/pt-pt/produtos-servi%C3%A7os/sistemas-de-rob%C3%B4/rob%C3%B4s-industriais/lbr-iiwa [Accessed: 03-03-2020].

[16] ESSS (2017). “Conheça os pilares da indústria 4.0”. [Online]. Available: https://www.esss.co/blog/os-pilares-da-industria-4-0/ [Accessed: 03-03-2020].

[17] Simul8 Corporation (2017). “Planning for Industry 4.0 with Simulation”. [Online]. Available: https://www.simul8.com/applications/manufacturing/implementing-industry-4-0-with-simulation [Accessed: 03-03-2020].

[18] Transformação Digital (2018). “Integração de sistemas: horizontal x vertical”. [Online]. Available: https://transformacaodigital.com/gestao/integracao-de-sistemas-horizontal-x-vertical/ [Accessed: 04-03-2020].

[19] Zhong, R. Y., Xu, X., & Wang, L. (2017). “IoT-enabled smart factory visibility and traceability using laser-scanners”. Procedia Manufacturing, vol. 10, pp. 1-14.

[20] Valuekeep (2019). “Differences between a cloud and an on-premise CMMS”. [Online]. Available: https://valuekeep.com/resources/e-books-articles/differences-between-a-cloud-and-an-on-premise-cmms/ [Accessed: 04-03-2020].

[21] Spirit AeroSystems (2018). “Spirit AeroSystems to Deliver First 3D-Printed Commercial Aircraft Part”. [Online]. Available:

https://www.spiritaero.com/pages/release/spirit-aerosystems-to-deliver-first-3d-printed-commercial-aircraft-part [Accessed: 04-03-2020].

[22] AugView (2018). “About the Augview product”. [Online]. Available: https://www.augview.net/English/product.html [Accessed: 04-03-2020].

[23] The MathWorks, Inc. (2016). “Introducing Machine Learning”. TheMathWorks, Inc.

[24] Pereira, F. C., & Borysov, S. S. (2019). “Machine learning fundamentals. In Mobility Patterns, Big Data and Transport Analytics”, pp. 9-29. Elsevier.

[25] Morgan, P. (2018). “Machine Learning is Changing the Rules: Ways Business Can Utilize AI to Innovate”. O'Reilly Media, Inc.

[26] Hodson, J. (2016). “How to Make Your Company Machine Learning Ready”. Harvard Business Review, pp. 1-4.

[27] Zheng, A., & Casari, A. (2018). “Feature engineering for machine learning: principles and techniques for data scientists”. O'Reilly Media, Inc.

[28] Smola, A., & Vishwanathan, S. V. N. (2008). “Introduction to machine Learning”. Cambridge University, UK.

[29] Khalid, S., Khalil, T., & Nasreen, S. (2014). “A survey of feature selection and feature extraction techniques in machine Learning”. Science and Information Conference, pp. 372-378. IEEE.

[30] Singh, S. (2018). Towards Data Science, “Understanding the Bias-Variance Tradeoff” [Online]. Available: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229 [Accessed: 08-06-2020].

[31] Friedman, J., Hastie, T., & Tibshirani, R. (2001). “The elements of statistical Learning”. New York: Springer series in statistics, vol. 1, no. 10.

[32] Andrew NG, Coursera. “Lecture Slides from Machine Learning Course” [Online]. Available: https://www.coursera.org/learn/machine-learning [Accessed: 20-07-2019].

[33] Burger, S. V. (2018). “Introduction to machine learning with R: Rigorous mathematical analysis”. O'Reilly Media, Inc.

[34] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). “An introduction to statistical Learning”. New York: Springer, vol. 112, p. 18.

[35] Dormehl, L. (2019). Digitaltrends, “What is an artificial neural network?” [Online]. Available: https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/ [Accessed: 20-06-2020].

[36] Gandhi, R. (2018). Towards Data Science, “Support Vector Machine — Introduction to Machine Learning Algorithms” [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 [Accessed: 20-06-2020].

[37] Dabbura, I. (2018). Towards Data Science, “K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks” [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a [Accessed: 02-07-2020].

[38] Al-Masri, A. (2019). Towards Data Science, “How Does k-Means Clustering in Machine Learning Work?” [Online]. Available: https://towardsdatascience.com/how-does-k-means-clustering-in-machine-learning-work-fdaaaf5acfa0 [Accessed: 02-07-2020].

[39] A. Saxena and K. Goebel (2008). "Turbofan Engine Degradation Simulation Data Set", NASA Ames Prognostics Data Repository (http://ti.arc.nasa.gov/project/prognostic-data-repository), NASA Ames Research Center, Moffett Field, CA

[40] Saxena, A., Goebel, K., Simon, D., & Eklund, N. (2008, October). “Damage propagation modeling for aircraft engine run-to-failure simulation”. In 2008 international conference on prognostics and health management, pp. 1-9. IEEE.

Anexos

Anexo A

Código para tratamento de dados

```
%% Tratamento de dados

clear; clc; close all;

% Carregamento de Dados

traindata = load('train_FD001.txt');

train_data =
array2table(traindata,'VariableNames',{'id_motor','cycles',...

'op1','op2','op3','sensor1','sensor2','sensor3','sensor4','sensor5','s
ensor6',...

'sensor7','sensor8','sensor9','sensor10','sensor11','sensor12','sensor
13',...

'sensor14','sensor15','sensor16','sensor17','sensor18','sensor19',...
    'sensor20','sensor21'});

% Máximo de vida de cada motor

[u,~,c] = unique(train_data{:,1});
MAX = accumarray(c,train_data{:,2},[],@max);
maxlifetrain = [u,MAX];
figure
barh(maxlifetrain,2,'blue')
title('Engine Life Time');
xlabel('Cycles');
ylabel('id motor');

clear('c');
clear('u');

%Criação de coluna RUL para cada linha

for numerolinha = 1:size(train_data,1)

    motor = traindata(numerolinha,1);

    train_data.RUL(numerolinha) = maxlifetrain(motor,2) -
traindata(numerolinha,2);

end

clear('motor');
clear('numerolinha');

%Análise de useless features
```

```
%Gráfico com evolução dos operational settings ao longo do tempo

figure
title('Evolução dos operational settings no
tempo','fontweight','bold','fontsize',16);
[~,ax]=plotmatrix(traindata(:,2),traindata(:,3:5));
title('Evolução dos operational settings no tempo');
ax(1,1).YLabel.String='OP1';
ax(2,1).YLabel.String='OP2';
ax(3,1).YLabel.String='OP3';
ax(3,1).XLabel.String='Ciclos de Vida';

%Gráfico com evolução dos sensores ao longo do tempo

figure
[~,ax]=plotmatrix(traindata(:,2),traindata(:,6:26));
title('Evolução dos sensores no
tempo','fontweight','bold','fontsize',16);
ax(1,1).YLabel.String='S 1';
ax(2,1).YLabel.String='S 2';
ax(3,1).YLabel.String='S 3';
ax(4,1).YLabel.String='S 4';
ax(5,1).YLabel.String='S 5';
ax(6,1).YLabel.String='S 6';
ax(7,1).YLabel.String='S 7';
ax(8,1).YLabel.String='S 8';
ax(9,1).YLabel.String='S 9';
ax(10,1).YLabel.String='S 10';
ax(11,1).YLabel.String='S 11';
ax(12,1).YLabel.String='S 12';
ax(13,1).YLabel.String='S 13';
ax(14,1).YLabel.String='S 14';
ax(15,1).YLabel.String='S 15';
ax(16,1).YLabel.String='S 16';
ax(17,1).YLabel.String='S 17';
ax(18,1).YLabel.String='S 18';
ax(19,1).YLabel.String='S 19';
ax(20,1).YLabel.String='S 20';
ax(21,1).YLabel.String='S 21';
ax(21,1).XLabel.String='Ciclos de Vida';

%Pelo gráfico é possível observar que o op3 e os sensores
1,5,6,10,16,18 e
%19 não são relevantes para o estudo. Podemos então eliminá-los,
juntamente
%com as colunas 1 e 2, referentes ao número do motor e ao número de
ciclo.

traindata = table2array(train_data);

train_data =
removevars(train_data,{'id_motor','cycles','op3','sensor1',...
'sensor5','sensor6','sensor10','sensor16','sensor18','sensor19'});

%Criação de classes para modelo de classificação

for numerolinha = 1:size(train_data,1)

    if train_data.RUL(numerolinha) <= 50
        train_data.CLASS(numerolinha) = 2;
```

```
    elseif train_data.RUL(numerolinha) > 50 &&
train_data.RUL(numerolinha) < 125
        train_data.CLASS(numerolinha) = 1;
    else
        train_data.CLASS(numerolinha) = 0;
    end
end

traindata = table2array(train_data);

%Guardar em ficheiro .txt

writematrix(traindata,'Dados de Treino');
```

Anexo B

Algoritmo de Regressão Linear – Regressão Linear Múltipla

```
%% Linear Regression Algorithm

%Iremos então analisar a aplicação de um algoritmo de regressão
linear.
%Uma vez que teremos valores na ordem das centenas e milhares e outros
na
%ordem das casas decimais, irá realizar-se primeiro uma Feature
%Normalization de forma a tornar todos os features na mesma ordem,
entre 0 e 1, tornando
%assim a convergência do gradient descent muito mais rápida. É um bom
%pré-processamento de dados para algoritmos de aprendizagem.

%% Carregamento de dados

clear ; close all; clc
fprintf('Loading data ...\n');


%Primeiro, iremos carregar os dados tratados em Data_Treatment


traindata = load('Dados de Treino.txt');
%Carregar como input os sensores
X = traindata(:,3:16);
%Carregar como output o RUL
y = traindata(:,17);
m = length(X);

fprintf('Data loaded. Press enter to continue.\n');
pause;

%% Feature Normalization

%Iremos então chamar a função Feature Normalization

fprintf('Normalizing Features ...\n');

[X mu sigma] = featureNormalize(X);

% Add intercept term to X
X = [ones(m, 1) X];

fprintf('Features Normalized. Press enter to continue.\n');
pause;

%% Gradient Descent

fprintf('Running gradient descent ...\n');

% Choose some alpha value
alpha = 0.2;
num_iters = 100;
```

```
% Init Theta and Run Gradient Descent

theta = zeros(15, 1);
[theta, J_history] = gradientDescentMulti(X, y, theta, alpha,
num_iters);

% Plot the convergence graph
figure;
plot(1:numel(J_history), J_history, '-b', 'LineWidth', 2);
xlabel('Number of iterations');
ylabel('Cost J');

% Display gradient descent's result
fprintf('Theta computed from gradient descent: \n');
fprintf(' %f \n', theta);
fprintf('\n');


% Calcular o ytheta (RUL previsto) baseado nos thetas calculados:

ytheta = X*theta;

%Calcular o erro

RMSE = sqrt(mean((y - ytheta).^2));

MAE = mean(abs(y-ytheta));
```

Feature Normalization

```
%% Função Feature Normalization

function [X_norm, mu, sigma] = featureNormalize(X)

X_norm = X;
mu = zeros(1, size(X, 2));
sigma = zeros(1, size(X, 2));

mu = mu + mean(X);
sigma = sigma + std(X);

m = length(X);
n = size(X,2);
for i = 1:m
    for j = 1:n
X_norm(i,j) = (X(i,j) - mu(1,j))/sigma(1,j);
    end
end

end
```

Compute Cost Multi

```
function J = computeCostMulti(X, y, theta)

%COMPUTECOSTMULTI Compute cost for linear regression with multiple
variables
%   J = COMPUTECOSTMULTI(X, y, theta) computes the cost of using theta
as the
%   parameter for linear regression to fit the data points in X and y

% Initialize some useful values
m = length(y); % number of training examples

% You need to return the following variables correctly
J = 0;

% ====================== YOUR CODE HERE ======================
% Instructions: Compute the cost of a particular choice of theta
%               You should set J to the cost.

hypothesis = X*theta;
sqrErrors = (hypothesis-y).^2;
J = 1/(2*m) * sum(sqrErrors);

%
=========================================================================
end
```

Gradient Descent

```
%% Gradient Descent

function [theta, J_history] = gradientDescentMulti(X, y, theta, alpha,
num_iters)
%GRADIENTDESCENTMULTI
%   theta = GRADIENTDESCENTMULTI(x, y, theta, alpha, num_iters)
updates theta by
%   taking num_iters gradient steps with learning rate alpha

% Initialize some useful values
m = length(y); % number of training examples
J_history = zeros(num_iters, 1);

for iter = 1:num_iters

    hypothesis = X*theta;

    for i = 1:size(X,2)
    theta(i) = theta(i) - alpha * (1/m) * sum((hypothesis-y).*X(:,i));
    end

    % ============================================================

    % Save the cost J in every iteration
    J_history(iter) = computeCostMulti(X, y, theta);

end
end
```

Anexo C

Algoritmo de Classificação – Support Vector Machine (com função de custo)

```
%% Multi-Class Suport Vector Machine


% Extract predictors and response
% This code processes the data into the right shape for training the
% model.
inputTable = train_data;
predictorNames = {'sensor2', 'sensor3', 'sensor4', 'sensor7',
'sensor8', 'sensor9', 'sensor11', 'sensor12', 'sensor13', 'sensor14',
'sensor15', 'sensor17', 'sensor20', 'sensor21'};
predictors = inputTable(:, predictorNames);
response = inputTable.CLASS;
isCategoricalPredictor = [false, false, false, false, false, false,
false, false, false, false, false, false, false, false];

% Set up holdout validation
cvp = cvpartition(response, 'Holdout', 0.2);
trainingPredictors = predictors(cvp.training, :);
trainingResponse = response(cvp.training, :);
trainingIsCategoricalPredictor = isCategoricalPredictor;

%% Função de Custo (Cost Matrix)

CostMatrix = ones(3)-eye(3);
CostMatrix(3,1:2) = 10;
CostMatrix(2,3) = 3;
CostMatrix(1,3) = 3;
disp(CostMatrix)

% Train a classifier
% This code specifies all the classifier options and trains the
classifier.
template = templateSVM(...
    'KernelFunction', 'gaussian', ...
    'PolynomialOrder', [], ...
    'KernelScale', 0.9399999999999999, ...
    'BoxConstraint', 1, ...
    'Standardize', true);
classificationSVM = fitcecoc(...
    trainingPredictors, ...
    trainingResponse, ...
    'Learners', template, ...
    'Coding', 'onevsone', ...
    'ClassNames', [0; 1; 2],...
    'Cost', CostMatrix);

% Create the result struct with predict function
svmPredictFcn = @(x) predict(classificationSVM, x);
validationPredictFcn = @(x) svmPredictFcn(x);
```

```
% Compute validation predictions
validationPredictors = predictors(cvp.test, :);
validationResponse = response(cvp.test, :);
[validationPredictions, validationScores] =
validationPredictFcn(validationPredictors);

% Compute validation accuracy
correctPredictions = (validationPredictions == validationResponse);
isMissing = isnan(validationResponse);
correctPredictions = correctPredictions(~isMissing);
validationAccuracy =
sum(correctPredictions)/length(correctPredictions);

%% Confusion Matrix

figure
cm = confusionchart(validationResponse,validationPredictions, ...
    'Title','Confusion Matrix', ...
    'RowSummary','row-normalized');
```