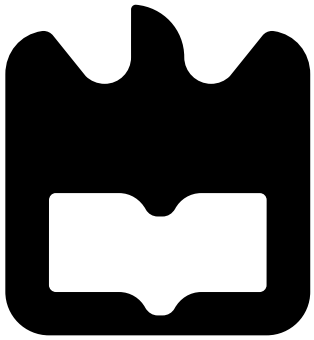
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2019

Cristiana
da Silva Carvalho

Atuação baseada em conhecimento nas cidades inteligentes



Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2019

Cristiana
da Silva Carvalho

Atuação baseada em conhecimento nas cidades inteligentes

Cristiana
da Silva Carvalho

Atuação baseada em conhecimento nas cidades inteligentes

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Ilídio Fernando de Castro Oliveira, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro e do Doutor Filipe Cabral Pinto, da Altice Labs (Aveiro).

o júri / the jury

presidente / president

Professor Doutor Luís Filipe de Seabra Lopes
Professor Associado, Universidade de Aveiro

vogais / examiners committee

Professora Doutora Ana Cristina Wanzeller Guedes de Lacerda
Professora Adjunta, Instituto Politécnico de Viseu - Escola Superior de Tecnologia

Professor Doutor Ilídio Fernando de Castro Oliveira
Professor Auxiliar, Universidade de Aveiro

agradecimentos / acknowledgements

A vocês pais, obrigada por me terem proporcionado todo este percurso e por nunca me terem deixado desistir.
A ti mana, obrigada por estares sempre presente, sempre com orgulho em mim.
Obrigada a toda a minha família por sempre acreditarem em mim e me apoiarem em todo este caminho.
Obrigada aos meus amigos e colegas, por todos os momentos juntos e bem passados, pelas trocas de ideias e ajudas necessárias. Obrigada a todos aqueles que ficam depois desta aventura, a todo o grupo QL que tão importante foi nesta fase final.
Obrigada a toda a empresa Altice Labs (Aveiro), que tão bem me acolheu ao longo destes meses e sempre se disponibilizou para ajudar quando necessário.
Obrigada aos meus orientadores, que sempre se mostraram disponíveis para qualquer dúvida e que prontamente sempre me ajudaram.
Sem vocês, não seria possível, obrigada a todos!

Resumo

O substancial aumento da população, a sua migração para centros urbanos e a desertificação de cidades de pequena e média dimensão por efeitos da migração referida têm colocado um conjunto de desafios ao planeamento e gestão das cidades. Surge a necessidade tanto de proteger recursos naturais e infraestruturas, como de otimizar processos e promover mais e melhores serviços digitais à população de modo a melhorar a qualidade de vida. Estes processos e serviços digitais, por exigência do dia-a-dia, necessitam de ter mecanismos automáticos que possibilitem a atuação célere no espaço urbano. Neste contexto, a utilização de lógicas de *Machine Learning* para mitigar em tempo útil eventuais contratempos, revela-se uma ferramenta útil para a fluídez diária da urbe, melhorando a qualidade de vida da população na cidade.

O principal objetivo desta dissertação era a criação de um modelo de *Machine Learning* a ser utilizado num sistema de gestão e automatização de uma cidade. A arquitetura foi criada de forma genérica, pelo que pode ser facilmente aplicada a outros cenários no âmbito das *Smart Cities*.

Este trabalho foi efetuado em colaboração com a empresa Altice Labs, no âmbito de um parceria entre a mesma e a Universidade de Aveiro.

Abstract

The substantial increase in population, their migration to urban centers and the desertification of small and medium-sized cities due to the effects of the mentioned migration have posed a number of challenges to city planning and management. There is a need both to protect natural resources and infrastructure, and to streamline processes and promote more and better digital services to the population in order to improve the quality of life. These processes and digital services, by day-to-day requirements, need to have automatic mechanisms that enable the rapid operation in urban space. In this context, the use of Machine Learning logics to timely mitigate any setbacks proves to be a useful tool for the daily fluidity of the city, improving the quality of life of the population in the city.

The main objective of this dissertation was to create a Machine Learning model to be used in city management and automation system. The architecture was created in a generic way, so it can easily be applied to other Smart Cities scenarios.

This work was done in collaboration with the company Altice Labs, in the framework of a partnership between it and the University of Aveiro.

Conteúdo

Conteúdo **i**

Lista de Figuras **iii**

Lista de Tabelas **v**

1 Introdução **1**
1.1 Motivação . . . 1
1.2 Objectivos . . . 2
1.3 Metodologia de trabalho . . . 3
1.4 Organização do documento . . . 4

2 Estado de Arte **5**
2.1 Cidades Inteligentes . . . 5
2.1.1 IoT: Internet of Things . . . 8
2.2 Machine Learning . . . 10
2.2.1 Pré-processamento . . . 11
2.2.2 Algoritmos . . . 12
2.3 *Machine Learning* aplicado às cidades . . . 19
2.4 Conjuntos de dados representativos . . . 20

3 Cenários **24**
3.1 Possíveis cenários . . . 24
3.2 História do cenário escolhido . . . 26

4 Arquitetura **27**
4.1 City Learning Model - Modelo de Aprendizagem da Cidade . . . 28
4.1.1 Offline Learning . . . 28
4.1.2 Runtime Assessment . . . 29
4.2 City Runtime . . . 30
4.2.1 Sensing . . . 30

4.2.2 Runtime Analitycs . . . 30
4.2.3 Actuation . . . 30
4.3 City Data Management Platform . . . 31
4.4 Arquitetura Detalhada . . . 31
4.5 Fluxo de Dados . . . 32
4.5.1 Fase da Aprendizagem . . . 32
4.5.2 Fase de Execução . . . 32
4.5.3 Fase da Avaliação . . . 33

5 Implementação **35**
5.1 Características do Conjunto de Dados Escolhido . . . 35
5.2 Pré-processamento . . . 35
5.2.1 Eliminanção de dados em branco ou *NaN* . . . 37
5.2.2 Feature Engineering . . . 38
5.2.3 Balanceamento do Conjunto de Dados . . . 41
5.2.4 Cross-validation . . . 49
5.3 Avaliação dos modelos . . . 57
5.4 Integração do Modelo . . . 58

6 Conclusões **61**
6.1 Desafios . . . 61
6.2 Trabalho futuro . . . 62

Bibliografia **63**

Lista de Figuras

1.1 Processo para o design de uma solução para um dado problema. 3

2.1 Modelo de referência conceptual de uma *Smart City* [1]. 6
2.2 Arquitetura Funcional TR-0057. [2] 8
2.3 Áreas de interesse do IoT (Adaptado de [3]). 9
2.4 Fluxo de Machine Learning. 10
2.5 Tipos de algoritmos de Machine Learning. 13
2.6 Árvore de Decisão adaptada de [4]. 15
2.7 Random Forest [5]. 16
2.8 K Nearest Neighbours. [6] 17
2.9 SVM [7]. 18
2.10 Comparação entre um neurónio humano e um neurónio artificial [8]. 18
2.11 Três primeiras fases do treino em Redes Neuronais [9]. 19

3.1 Cenário 1 - Priorização de acidentes. 24
3.2 Cenário 2 - Detetar causas de acidentes. 25
3.3 Cenário 3 - Qualidade do Ar. 25

4.1 Arquitetura genérica do sistema. 27
4.2 City Learning Model. 28
4.3 Construção do modelo de Machine Learning. 28
4.4 Offline Learning. 29
4.5 Runtime Assessment. 29
4.6 City Runtime. 30
4.7 Runtime Analytics. 31
4.8 Arquitetura detalhada do sistema. 31
4.9 Fluxo da Cognitive City Framework. 32

5.1 Número de atributos/colunas sem e com seleção. 40
5.2 Número de linhas com e sem tratamento. 40
5.3 Matriz de correlação. 41

5.4 Percentagem da frquência de ocorrências das diferentes classes. 41
5.5 Tomek's links. [10] 42
5.6 Edited Nearest Neighbours [10]. 43
5.7 Dados orginais.[11] 44
5.8 Dados aquando da aplicação do SMOTE. [11] 44
5.9 Dados orginais.[11] 45
5.10 Dados aquando da aplicação do ADASYN. [11] 45
5.11 Dados orginais.[11] 47
5.12 Dados aquando da aplicação do BorderlineSMOTE. [11] 47
5.13 Dados orginais.[11] 48
5.14 Dados aquando da aplicação do SVMSMOTE. [11] 48
5.15 Dados orginais.[11] 49
5.16 Dados aquando da aplicação do KMeansSMOTE. [11] 49
5.17 Representação da técnica *cross-validation*. [12] 50
5.18 Mockup da integração do modelo na plataforma. 58
5.19 Modelo integrado numa plataforma para monitorização. 60

Lista de Tabelas

1.1 População do mundo em 2017, 2030, 2050, 2100 [13]. 1

2.1 Exemplo de conjunto de dados adaptado de [4]. 15

5.1 Antes do pré-processamento. 37
5.2 Depois do pré-processamento. 37
5.3 Accuracy e *Recall* com e sem determinados atributos. 40
5.6 Resultado sem e com seleção de atributos, respetivamente para a técnica *Tomek's Links*. 43
5.7 Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *Tomek's Links*. 43
5.10 Resultado sem e com seleção de atributos, respetivamente para a técnica *Edited Nearest Neighbours*. 44
5.13 Resultado sem e com seleção de atributos, respetivamente para a técnica *SMOTE*. 44
5.14 Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *SMOTE*. 45
5.17 Resultado sem e com seleção de atributos, respetivamente para a técnica *ADASYN*. 46
5.18 Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *ADASYN*. 46
5.21 Resultado sem e com seleção de atributos, respetivamente para a técnica *BorderlineSMOTE*. 47
5.22 Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *BorderlineSMOTE*. 47
5.25 Resultado sem e com seleção de atributos, respetivamente para a técnica *SVMSMOTE*. 48
5.26 Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *SVMSMOTE*. 48
5.29 Resultado sem e com seleção de atributos, respetivamente para a técnica *KMeansSMOTE*. 49

5.30 Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *KMeansSMOTE*. 49
5.33 Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Naïve Bayes*. 50
5.34 Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Naïve Bayes*. 51
5.37 Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Árvores de Decisão*. 51
5.38 Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Árvores de Decisão*. 51
5.41 Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Random Forest*. 51
5.42 Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Random Forest*. 52
5.45 Resultado com e sem *cross-validation*, respetivamente para o algoritmo *KNN*. . 52
5.46 Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *KNN*. 52
5.49 Resultado com e sem *cross-validation*, respetivamente para o algoritmo *SVM*. . 52
5.50 Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *SVM*. 53
5.53 Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Neural Networks*. 53
5.54 Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Neural Networks*. 53
5.56 Resultado com PCA para o algoritmo *Naïve Bayes*. 54
5.57 Matriz de confusão com PCA para o algoritmo *Naïve Bayes*. 54
5.59 Resultado com PCA para o algoritmo *Árvores de Decisão*. 54
5.60 Matriz de confusão com PCA para o algoritmo *Árvores de Decisão*. 55
5.62 Resultado com PCA para o algoritmo *Random Forest*. 55
5.63 Matriz de confusão com PCA para o algoritmo *Random Forest*. 55
5.65 Resultado com PCA para o algoritmo *KNN*. 55
5.66 Matriz de confusão com PCA para o algoritmo *KNN*. 56
5.68 Resultado com PCA para o algoritmo *SVM*. 56
5.69 Matriz de confusão com PCA para o algoritmo *SVM*. 56
5.71 Resultado com PCA para o algoritmo *Neural Networks*. 56
5.72 Matriz de confusão com PCA para o algoritmo *Neural Networks*. 57
5.76 Resultados dos algortimos *Random Forest*, SVM e *Neural Networks*, respetivamente. 57

Capítulo 1

Introdução

1.1 Motivação

O aumento da população a nível mundial é uma realidade desde há alguns anos. Segundo o relatório das Nações Unidas, *"A Projeção da População Mundial: Revisão de 2017"*, estima-se que a população mundial possa chegar a 8,6 mil milhões de pessoas em 2030, 9,8 mil milhões em 2050 e 11,2 mil milhões até 2100 [13].

Região	População (Milhões)			
	2017	**2030**	**2050**	**2100**
World.....	7 550	8 551	9 772	11 184
África.....	1 256	1 704	2 528	4 468
Ásia.....	4 504	4 947	5 257	4 780
Europa.....	742	739	716	653
América Latina.....	646	781	780	712
América do Norte.....	361	395	435	499
Oceania.....	41	48	57	72

Tabela 1.1: População do mundo em 2017, 2030, 2050, 2100 [13].

O crescimento substancial da população e a sua migração para os centros urbanos bem como o isolamento e envelhecimento de cidades de pequenas dimensões têm colocado um conjunto de desafios ao planeamento e gestão das cidades. Existe uma cada vez maior necessidade de proteger os recursos naturais, de salvaguardar as infraestruturas e os equipamentos, de otimizar processos e de promover mais e melhores serviços digitais à população de modo a melhorar a qualidade de vida nas cidades. São muitos os exemplos do quotidiano que se podem referir: o aumento do trânsito, a escassez e ineficiência da rede de transportes públicos, o aumento da produção de resíduos sólidos urbanos, o maior número de problemas de saúde

associados à diminuição da qualidade ambiental, entre muitos outros.

Atualmente, com a agitação do dia-a-dia e com a necessidade de que tudo seja realizado de forma muito rápida, a população começa a exigir a digitalização dos processos e a criação de mecanismos automáticos que possibilitem a atuação célere no espaço urbano. O conceito de *Smart City* surge associado à utilização das tecnologias de informação e comunicação na gestão diária das cidades [14]. Com o advento da Internet das Coisas *(IoT - Internet of Things)* passa a ser possível conhecer de forma permanente o estado da cidade e atuar nas suas diferentes infraestruturas, contribuindo para a melhoria da vida de quem vive ou visita os diferentes municípios [3]. A IoT possibilita a visualização, o conhecimento e a atuação nas suas diferentes infraestruturas em "tempo real". Através da parametrização de regras será possível assegurar a adaptação da cidade para responder da melhor forma a determinadas ocorrências, permitindo dinâmicas baseadas no contexto. No entanto, tipicamente estas regras têm um cariz estático, estando muitas vezes dependentes de um só domínio, não dando resposta atempada a muitas situações. Neste contexto, a utilização de mecanismos de *Machine Learning* para antever determinadas ocorrências na cidade e mitigar em tempo útil os eventuais contratempos revela-se uma ferramenta útil para a fluidez diária da urbe. Assim, antecipando situações e atuando imediatamente será possível melhorar a dinâmica da cidade e promover um aumento da qualidade de vida do cidadão.

Esta dissertação tem como motivação principal a criação de um modelo de *Machine Learning* a ser utilizado num sistema de gestão e automatização de uma cidade, para que a esta consiga adaptar-se às necessidades da população, melhorando a qualidade de vida na cidade ao combinar IoT e *Machine Learning*.

A proposta deste trabalho teve origem na empresa Altice Labs em Aveiro, com a orientação empresarial do Engenheiro Filipe Cabral Pinto, tendo em conta os projetos em que a empresa está envolvida.

1.2 Objectivos

O objetivo principal desta dissertação é o desenvolvimento de uma componente funcional para um cenário em específico, que permita monitorizar uma cidade de forma inteligente e atuar em tempo útil de modo a evitar situações críticas à dinâmica da cidade. Serão utilizados mecanismos de *Machine Learning* para aprender um modelo que permita prever a ocorrência de eventos. De acordo com as situações detetadas, serão utilizados mecanismos para atuar nas infraestruturas da cidade para atenuar os efeitos. Para o seu desenvolvimento existem alguns objetivos específicos:

1. Analisar bases de dados abertas de diferentes municípios;

2. Definir o cenário e extrair requisitos;

3. Definir arquitetura a ser usada;

4. Preparar os dados para permitir a utilização de diferentes algoritmos de *Machine Learning*;

5. Aprender diferentes modelos e testar o seu sucesso;

6. Aplicar o modelo na gestão dinâmica de diferentes infraestruturas.

1.3 Metodologia de trabalho

Na elaboração da presente dissertação, a metodologia de trabalho adoptada segue o ***Engineering Design Process*** representado na Figura 1.1 [15]. São vários os passos que devem ser seguidos de forma a chegar a uma solução.

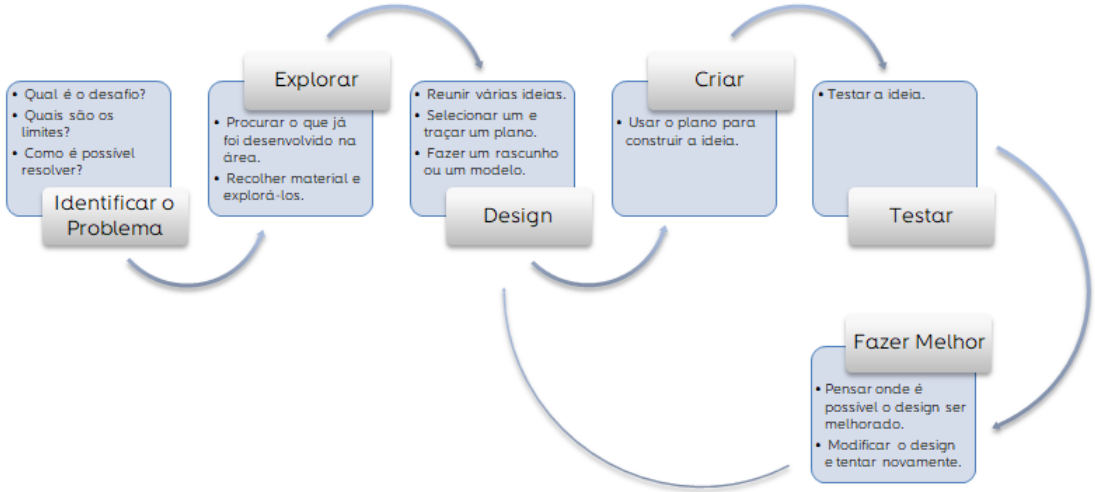


Figura 1.1: Processo para o design de uma solução para um dado problema.

Todas as etapas reunem um conjunto de pontos a ter em consideração e que servem como guia à resolução de cada uma delas.

Antes de qualquer outra coisa, é necessário **identificar o problema**, questionando qual o desafio presente, quais são os limites e como é que é possível resolver o problema identificado. Quando esta etapa estiver terminada, é necessário **explorar** o que já foi desenvolvido na área e recolher material necessário para a sua exploração. De seguida, passa-se então ao **design** de um modelo, reunindo diferentes ideias, selecionando e traçando um plano para esta. Desenvolvido o modelo, é altura de **criar**, usando o plano para construir a ideia para que se possa **testar**. Quando finalmente a ideia é testada, avalia-se o que é possível **fazer melhor** no design e modifica-se se necessário, passando novamente à etapa do **design**.

1.4 Organização do documento

Esta dissertação encontra-se dividida em cinco capítulos. Cada capítulo terá subsecções, permitindo que os temas possam ser abordados de forma organizada e facilmente compreensível. Neste sentido, esta dissertação é constituída pelos seguites capítulos:

- **Capítulo 1:** corresponde à Introdução, onde é apresentado o enquadramento do tema, assim como a descrição da motivação, objetivos e metodologia de trabalho utilizada na elaboração desta dissertação.
- **Capítulo 2:** é dedicado ao Estado de Arte, no qual são apresentados quatro subsecções: uma dedicada às *Smart Cities*, sendo apresentado e descrito detalhamente o conceito de *Smart Cities*; a seguinte decidada à *Machine Learning* e contém a explicação do processamento de dados, de diversos algoritmos e as suas diferentes aplicações; outra dedicada à *Machine Learning* aplicada às cidades e por fim uma secção com os conjuntos de dados representativos que foram analisados.
- **Capítulo 3:** denominado de Cenários, onde são apresentados possíveis cenários de utilização, bem como a história do cenário escolhido.
- **Capítulo 4:** denominado de Arquitetura, neste capítulo é feita a descrição detalhada da arquitetura desenhada para o sistema.
- **Capítulo 5:** denominado por Implementação, neste é descrito todo o processo de implementação da plataforma e os resultados obtidos.
- **Capítulo 6:** capítulo onde são descritas as conclusões finais na sequência da análise dos resultados obtidos, respondendo-se aos objetivos da investigação proposta.

Capítulo 2

Estado de Arte

2.1 Cidades Inteligentes

"Smart City" é uma expressão muito utilizada hoje em dia podendo no entanto o seu significado ser analisado sob diferentes perspectivas e para o qual existem múltiplas definições. Não obstante esta diversidade, é geralmente compreendido que uma *Smart City* é uma cidade em que a eficiência e a otimização dos recursos, infraestruturas e serviços são potenciados pela utilização das tecnologias de informação e comunicação. A cidade, de acordo com *Zygiaris*, pode decompor-se em diferentes pilares: economia, mobilidade, ambiente, vida e governo [1]. Para melhorar o desempenho urbano em todos os seus domínios, a cidade recorre a dados e às Tecnologias da Informação (IT) para monitorizar, analisar e atuar no espaço urbano, de forma a aumentar a colaboração entre diferentes agentes económicos e incentivar modelos empresariais inovadores, tanto no sector privado, como no sector público, tendo como objetivo final a melhoria da qualidade de vida do cidadão [16].

A construção de uma *Smart City* pressupõe a evolução tendo em conta um conjunto de orientações concretas [1]:

- **Sustentáveis:** infraestruturas urbanas que conduzam à proteção do meio ambiente;
- **Interligadas:** revolução da economia de banda larga;
- **Inteligentes:** capacidade de produzir conhecimento através do processamento de dados em tempo real, recorrendo a dispositivos (sensores, atuadores, gateways) distribuídos pela cidade;
- **Inovação e Conhecimento:** capacidade de aumento da inovação com base na inteligência e na criatividade humana.

Zygiaris estruturou um modelo de referência conceptual para as *Smart Cities*, seguindo uma abordagem baseada em 7 camadas, conforme se ilustra na Figura 2.1.

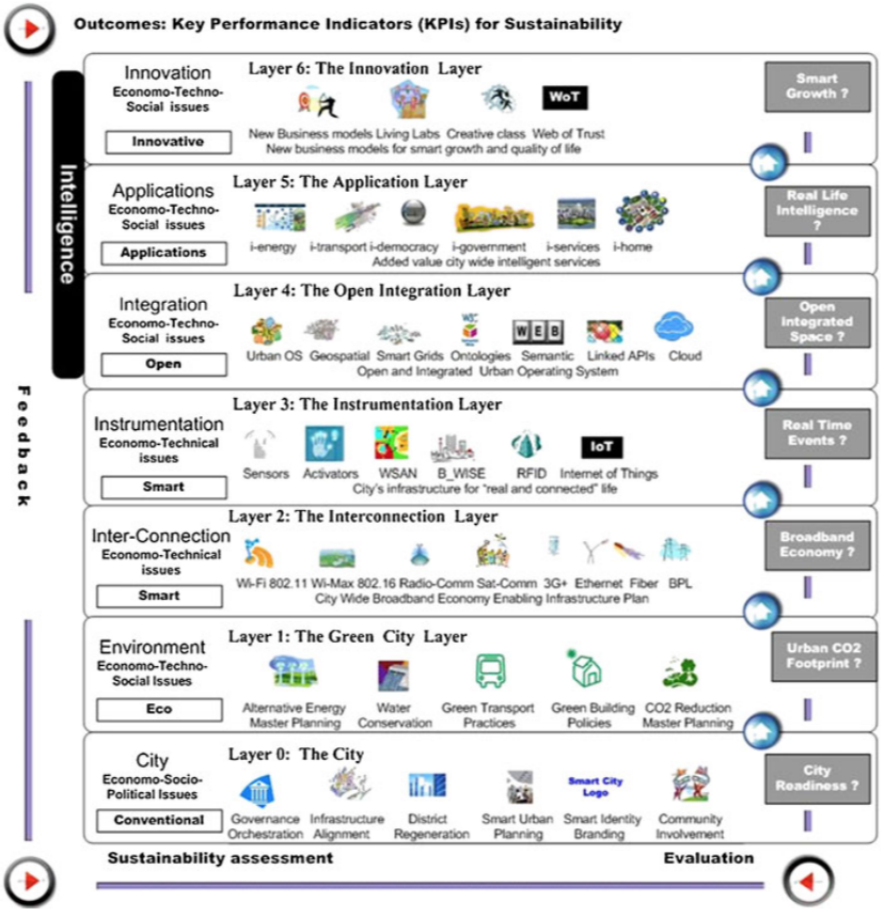


Figura 2.1: Modelo de referência conceptual de uma *Smart City* [1].

Camada 0: A Cidade

A cidade é definida por uma determinada área, caracterizada pela existência de uma densidade populacional e de determinadas infraestruturas [17]. Geralmente tem a sua própria administração para manter e fornecer serviços públicos [18].

Nesta camada estão representados todos os componentes tradicionais geralmente contidos numa urbe, como por exemplo o alinhamento de infraestruturas, o involvimento da comunidade, o planeamento de uma urbe inteligente, entre outros.

Camada 1: A Cidade Verde

A constituição de espaços urbanos sustentáveis tem subjacente a adoção de políticas e estruturas diretamente ligadas a *Smart Cities*. Nesta camada, são consideradas as políticas verdes que se traduzem na adoção de transportes ecológicos e construções verdes [1].

A camada da Cidade Verde refere-se a todas as políticas e infraestruturas verdes que são essenciais para a redução da pegada ecológica. A inovação e a inteligência, que podem ser

aplicadas nas cidades, são fatores que podem contribuir substancialmente para essa redução [19].

Camada 2: Interconexão

A interconexão, através de redes urbanas de fibra ótica e de redes sem fios, tem a capacidade de conectar as economias verdes e de banda larga, sendo este o primeiro passo para criar uma cidade inteligente e conectada [1]. Tal significa a comunicação entre os diferentes serviços de uma cidade [20], potencializando a sua capacidade económica e correspondente inclusão social, através da cobertura da urbe, conectando comunidades físicas à medida que estas realizam as suas tarefas de dia-a-dia, podendo, à distância manterem-se sempre em contacto [1].

Camada 3: Instrumentação

A disponibilidade de dados em tempo real é um elemento fundamental para a construção de uma *Smart City* [1]. Através da instrumentação é possível a recolha e a integração de dados do mundo real. Dispositivos como sensores, medidores, dispositivos pessoais, aparelhos, câmaras, *smartphones*, entre outros, são normalmente usados para este tipo de aplicação [20].

A camada da Instrumentação refere-se, assim, às diferentes formas de recolha e integração de dados.

Camada 4: Integração Aberta

Um processo importante para a monitorização de uma cidade é a capacidade de integrar e de tornar os recursos digitais abertamente disponíveis. Esta camada promove o desenvolvimento da camada 5, através da disponibilização de dados que podem ser utilizados por aplicações promovendo assim um ecossistema de inovação [1].

Camada 5: Aplicação

Várias são as aplicações possíveis, em diferentes serviços e infraestruturas que possibilitam a construção de uma cidade inteligente.

A camada de Aplicação consubstancia estas diversas aplicações que, no seu todo, representam não só a resolução de questões tecnológicas e económicas mas também questões sociais, promovendo a economia e o bem estar do cidadão [1].

Camada 6: Inovação

Considera-se que a inovação está muito ligada à inteligência, referindo-se esta a análises de processos de forma a facilitar a tomada das melhores decisões [20].

Embora não exista, ainda, capacidade para resolver todas as questões urbanas, grande parte destas podem enquadrar-se neste conceito em camadas para as *Smart Cities.*

2.1.1 IoT: Internet of Things

A IoT (*Internet of Things*) "estende"a Internet aos objectos potenciando uma miríade de novos serviços. Terá impacto na inovação, na criação de novos negócios e revolucionará a forma como se vive em sociedade. Mas a IoT é também um novo paradigma tecnológico caracterizado por ser uma rede de máquinas e dispositivos capazes de comunicar entre si [21], proporcionando às *"Coisas - Things"*, com capacidade computacional e de comunicação, a ligação à Internet. Assim sendo, é possível denomiar as *"Coisas - Things"*por possuírem capacidade de comunicação e processamento aliados a sensores e a atuadores, como objetos inteligentes [3].

A Figura 2.2 representa a arquitetura oneM2M, cujo principal objetivo é normalizar uma arquitetura global IoT de forma a terminar com a construção de silos, diminuindo a fragmentação de soluções verticais. Esta minimização é conseguida através da consolidação das atividades atuais destas e através do desenvolvimento de especificações globais [22].

Esta arquitetura está dividida em 3 camadas: Camada da Aplicação (***Application Layer***), Camada de Serviço (***Service Layer***), Camada de Conectividade (***Connectivity Layer***) [2].

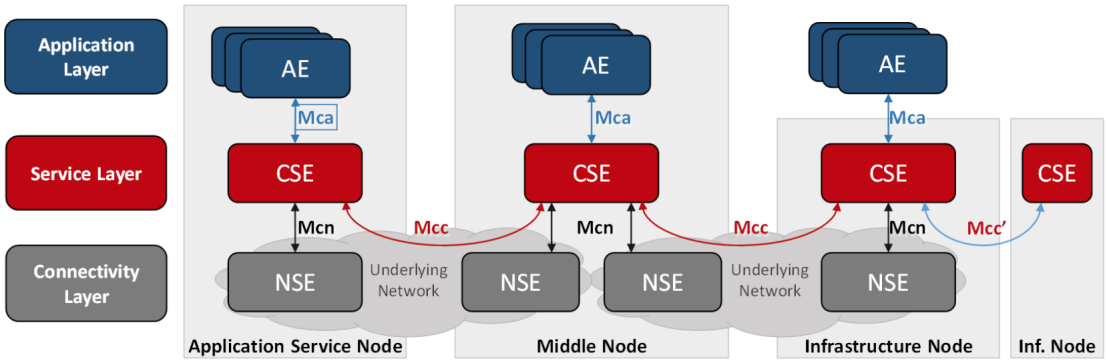


Figura 2.2: Arquitetura Funcional TR-0057. [2]

Cada uma das camadas tem diferentes entidades. Na Camada da Aplicação encontra-se a *AE* (***Application Entity***). Esta é uma entidade cujo objetivo é fornecer lógica aplicacional a soluções M2M *end-to-end* [22]. Podem existir em vários nós M2M e/ou num só nó múltiplos *AE* [2]. Na camada seguinte, a Camada de Serviço, encontra-se presente a *CSE* (***Common services entity***), uma entidade lógica que é instanciada num nó M2M e consiste num conjunto de funções de serviço, as chamadas *CSFs* (***Common Services Functions***), que podem ser usadas pelas *AE* [22]. Estas funções exteriorizadas para outras entidades através de pontos de referência. No caso de exposição a *AEs* são usados Mca; por outro lado, no caso de exposição a outros *CSEs* são usados Mcc. Finalmente, na Camada de Conectividade, está presente a *NSE* (***Network Service Exposure***). Esta é uma entidade que fornece serviços do *Underlying Network* para os *CSEs*, através dos pontos de referência Mcn [2]. Este tipo de arquitetura

M2M é funtamental no progresso eficiente de IoT.

Existem algumas tendências que podem ser usadas para o desenvolvimento IoT. De seguida, são apresentadas 5 dessas tecnologias consideradas fundamentais para o seu desenvolvimento [21].

- ***Devices e Gateways*:** permite a recolha de dados e atuação em sistemas;
- ***Wireless Sensor Networks* (WSN):** consistem em dispositivos equipadas com sensores autónomos, distríbuidos espacialmente para monitorar condições físicas ou ambientais, permitindo diferentes topologias de rede e comunicações *multihop*;
- ***Middleware:*** é uma camada de software entre as aplicações e os dispositivos, que elimina eventuais diferenças entre os protocolos de comunicação.
- ***Cloud Computing:*** modelo de armazenamento e processamento que recorre à partilha de um conjunto de recursos configuráveis, de forma eficiente;
- ***IoT Application Software:*** permite interações entre dispositivos (*device-to-device*) e entre o Humano e os dispositivos (*human-to-device*), de maneira robusta e confiável. É necessário que garantam que os dados enviados sejam recebidos e executados adequadamente.

Neste contexto, a Figura 2.3 representa algumas áreas de aplicação do IoT.

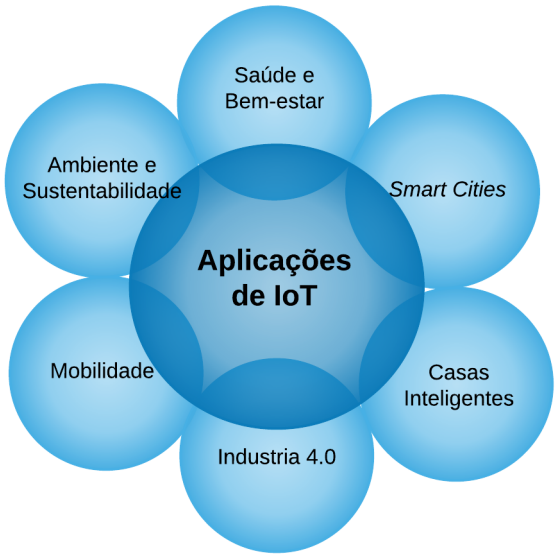


Figura 2.3: Áreas de interesse do IoT (Adaptado de [3]).

De entre as diferentes áreas representadas, o foco nesta dissertação são as *Smart Cities*. Neste domínio, podem considerar-se várias sub-áreas de interesse [3], no entanto foram selecionados dois tópicos mais concretos.

- **Mobilidade:** monitorização das condições de tráfego; estradas inteligentes com notificações e desvios, de acordo com as condições meteorológicas ou eventuais situações inesperadas como acidentes ou trânsito congestionado; e integração inteligente de plataformas de transporte.
- **Ambiente e Sustentabilidade:** controlo da iluminação inteligente e toda a energia elétrica produzida e consumida; monitorização de condições ambientais; controlo da poluíção; deteção dos níveis de resíduos nos pontos de recolha dos mesmos, de modo a otimizar as rotas a efetuar para esta recolha.

Neste âmbito, torna-se necessário direcionar vários esforços no planeamento apropriado, para que as cidades, atuais e futuras, possam albergar de modo robusto, criativo e sustentável o desenvolvimento necessário para a melhoria da qualidade de vida da sociedade, trabalho para o qual a tecnologia pode contribuir eficazmente, sobretudo através do IoT - *Internet of Things* [23]. Este conceito é, ainda, um grande desafio, tanto a nível conceitual, como a também a nível tecnológico, pois são várias as tecnologias juntas num sistema [3]. Posto isto, é importante perceber que embora existam obstáculos, eles não são intransponíveis [24].

2.2 Machine Learning

Machine Learning, também conhecida em português como Aprendizagem Automática, é um domínio de Inteligência Artificial que fornece aos sistemas a capacidade de aprenderem automaticamente. A sua aplicação foca-se no desenvolvimento de modelos, cuja sua própria aprendizagem é feita através de dados que foram previamente recolhidos e tratados [25].

O fluxo iterativo para a aplicação de *Machine Learning* a um dado conjunto de dados pode ser representado pela Figura 2.4.

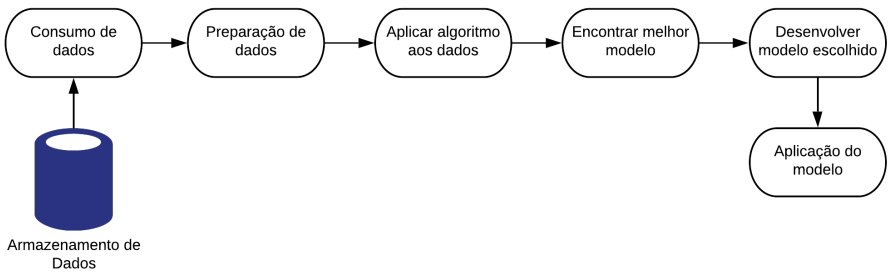


Figura 2.4: Fluxo de Machine Learning.

2.2.1 Pré-processamento

Para os programas obterem uma aprendizagem o mais eficiente possível é necessário que se preparem previamente os dados a serem utilizados. A esta preparação dá-se o nome de **pré-processamento**, podendo ter diferentes aplicações, tais como o tratamento de dados quando existe ausência de determinados valores, a codificação de atributos categóricos, a padronização e normalização dos dados, entre outras.

Para a realização deste pré-processamento são utilizadas a biblioteca **scikit-learn, pandas** e **numpy**, disponibilizada pela linguagem de programação **Python**.

Ausência de dados

É muito frequente os *datasets* utilizados terem valores em falta, sendo que geralmente são apresentados com espaços em branco, *NaN (Not a Number)* ou outro tipo de representação. Esta ausência de valores é incompatível com a utilização da biblioteca scikit-learn.

Uma estratégia básica de resolução deste problema é a remoção das linhas e/ou colunas de um determinado *dataset* com valores em falta. Esta aplicação tem, no entanto, um custo associado, concretamente a perda de dados se que podem revestir de elevada importância para a aprendizagem. Outra estratégia, mais interessante deste ponto de vista, é a atribuição de valor onde existe ausência dos mesmo. Esta atribução pode ocorrer de diversas formas, como por exemplo, através da inserção do valor da média, mediana ou moda de cada coluna cujo valor está em falta [26].

Codificação de Atributos Categóricos

Nem todos os *datasets* têm apenas características numéricas. Quando estão inseridos valores nominais, diz-se que estamos perante um conjunto de dados categórico [27].

Diversas técnicas de *Machine Learning* exigem que os valores de entrada sejam numéricos, sendo, por isso, necessário proceder a uma codificação prévia dos dados categóricos, a que se dá o nome de codificação categórica.

Este processo consiste em atribuir a cada valor categórico um dado numérico. Esta atribuição pode ser efetuada de diversas formas, tal como é sinteticamente apresentado de seguida.

(a) **Codificação Ordinal**

Este tipo de codificação transforma cada atribuito categórico num atributo inteiro de forma ordenada. No entanto, esta representação ordenada de valores inteiros pode não ser a mais adequada para determinados modelos de *Machine Learning* que esperam uma entrada contínua, interpretando as categorias como ordenadas [27].

(b) **Codificação 1-de-K**

Outro tipo de abordagem é a codificação 1-de-K, que transforma cada atributo categórico com N valores possíveis, em N atributos binários, sendo um deles igual a 1 e os restantes a igual a 0 [28][27].

Padronização de dados

Os algoritmos de *Machine Learning* dão, muitas vezes, como garantida a padronização dos dados [29]. Como se assume que os dados estão todos à mesma escala, quando tal não acontece, existe uma elevada probabilidade de determinados algoritmos se tornarem ineficientes. É, por isso, importante nestes casos proceder-se à padronização dos dados.

Na prática, esta técnica ignora a forma da distribuição e transforma os dados de modo a centralizá-los.

A padronização de um atributo j num objecto i é dada por:

$$x_{ij} = \frac{x_{ij} - \bar{x}_{.j}}{s_j} \tag{2.1}$$

sendo $\bar{x}_{.j}$ e s_j a média de j e o desvio padrão, respetivamente.

A média de cada atributo é colocada a zero e o desvio padrão a um [28]. Assume-se, assim, que não existem valores discrepantes e que tudo está normalizado.

Normalização

Ao processo de transformar amostras individuais com o objetivo destas terem uma norma unitária, dá-se o nome de normalização [30].

A normalização de um atributo j num objecto x_i é dada por:

$$x_{ij} = \frac{x_{ij} - min_j}{max_j - min_j} \tag{2.2}$$

sendo max_j e min_j o valor máximo e mínimo de j, respetivamente [28].

2.2.2 Algoritmos

Existem diversos algoritmos de *Machine Learning* que podem ser aplicados, de acordo com os cenários de aplicação. A escolha do melhor algoritmo, a maioria das vezes, não é óbvia, nem linear, e, portanto, é necessário testar com os diferentes algortimos, até se obterem os melhores resultados.

Os algoritmos podem ser divididos em 3 grandes categorias, como ilustra a Figura 2.5.

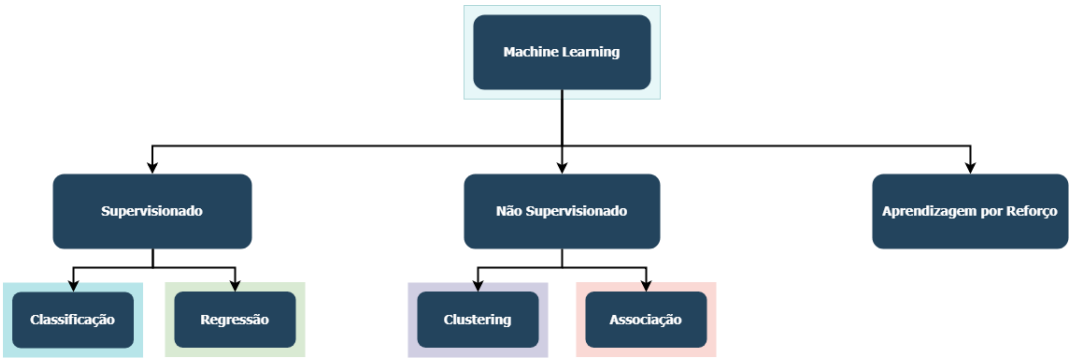


Figura 2.5: Tipos de algoritmos de Machine Learning.

Supervisonados

A aprendizagem supervisionada é aquela cujos os dados contém os atributos em que o valor será previsto, ou seja o valor de saída [31].

Neste caso, o objetivo é aprender através do atributos de entrada de forma a obter uma determinada saída, que pode ou não, estar correta [32].

Os algoritmos pertencentes ao tipo dos supervisionados estão divididos ainda em 2 categorias: classificação, regressão [33].

(a) **Classificação**

Neste tipo de algoritmos de *Machine Learning* as amostras pertencem a duas ou mais classes e o objetivo é que com os dados já classificados, se consiga aprender a fazer essa classificação a amostras que ainda não estão rotuladas [33].

(b) **Regressão**

Os algoritmos inseridos nesta categoria caracterizam-se pelas suas saídas consistirem em uma ou mais variáveis contínuas, isto é, variáveis numéricas que têm um número infinito de valores entre dois valores quaisquer [33].

Não Supervisonados

A aprendizagem não supervisionada é aquela em que apenas existem dados de entrada e, portanto, não existe nenhum dado de saída que diga qual deveria ser o resultado. Neste caso, o objectivo é encontrar um padrão nos atributos de entrada de forma a se obter um dado de saída [32].

Os algortimos pertencentes ao tipo dos não supervisionados estão divididos ainda em 2 categorias: clustering e associação.

(a) **Clustering**

Os algoritmos inseridos nesta categoria caracterizam-se pela divisão dos dados em grupos de objectos semelhantes [34].

(b) **Associação**

Uma forma de encontrar padrões num conjunto de dados é através da Associação. Neste caso, tenta-se encontrar relações entre os dados para que se consiga prever determinada acção.

Um exemplo prático, onde esta téncica é muito utilizada, é na disposição de artigos nos supermecados. São identificadas as relações entre o que normalmente as pessoas compram em conjunto. A entidade vendedora pode identificar assim que artigos devem estar juntos [35].

Aprendizagem por Reforço

A Aprendizagem por Reforço difere da Aprendizagem Supervisionada na medida em que nesta, os dados treinados têm a correspondente predição correta. O mesmo não acontece na Aprendizagem por Reforço, uma vez que aqui a aprendizagem é obtida pela experiência.

Exemplos de Algoritmos

São diversos os algoritmos que podem ser aplicados. De seguida é dada a explicação de alguns exemplos desses algoritmos, todos eles supervisionados, uma vez que o trabalho em questão se refere a aprendizagem supervisionada.

Para isto, será considerado um conjunto de dados pequeno e simples, de modo a que a explicação se torne mais eficiente.

Índice	**Meteorologia**	**Temperatura**	**Humidade**	**Vento**	**Jogar**
0	Ensolarado	Quente	Alta	Falso	Não
1	Ensolarado	Quente	Alta	Verdadeiro	Não
2	Nublado	Quente	Alta	Falso	Sim
3	Chuvoso	Ameno	Alta	Falso	Sim
4	Chuvoso	Frio	Normal	Falso	Sim
5	Chuvoso	Frio	Normal	Verdadeiro	Não
6	Nublado	Frio	Normal	Verdadeiro	Sim
7	Ensolarado	Ameno	Alta	Falso	Não
8	Ensolarado	Frio	Normal	Falso	Sim
9	Chuvoso	Ameno	Normal	Falso	Sim
10	Ensolarado	Ameno	Normal	Verdadeiro	Sim
11	Nublado	Ameno	Alta	Verdadeiro	Sim

12	Nublado	Quente	Normal	Falso	Sim
13	Chuvoso	Ameno	Alta	Verdadeiro	Não

Tabela 2.1: Exemplo de conjunto de dados adaptado de [4].

Com o conjunto de dados da Tabela 2.1 é possível, perante os diversos atributos, prever a decisão de jogar ou não.

Naive Bayes

O algoritmo de Naive Bayes consiste num modelo probabilistico de *Machine Learning*, usado como **classificador** e é baseado no teorema de Bayes.

Teorema de Bayes

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.3}$$

O Teorema de Bayes permite calcular a probabilidade de A, sabendo que já aconteceu B. Deve assumir-se, neste caso, que os atributos são independentes [36].

Considerando a Tabela 2.1, o objectivo é classificar se se vai jogar ou não, tendo em consideração os vários atributos relativos ao dia.

Se se tiver em conta o Teorema de Bayes, então diz-se que A corresponde à classe variável, neste caso o jogar ou não, e que B corresponde aos parâmetros.

Árvores de Decisão

Voltando ao exemplo da Tabela 2.1, é possível ser aprendida a árvore de decisão da Figura 2.6, através o cálculo da entropia e ganho de informação. Assim sendo, pode definir-se como árvore de decisão a representação de uma determinada tabela sob a forma de árvore. Este tipo de modelo pode ser usado tanto como **classificação** ou **regressão** [37].

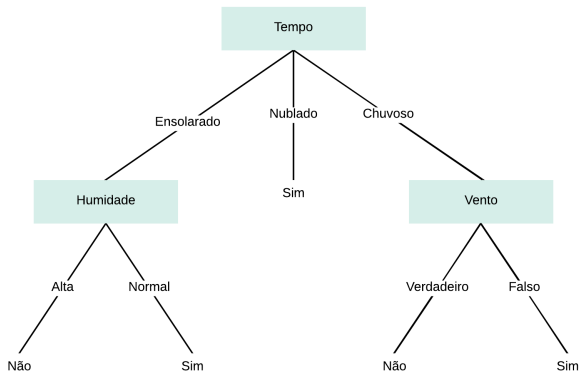


Figura 2.6: Árvore de Decisão adaptada de [4].

Para a construção da árvore representada na Figura 2.6 é necessário determinar qual o melhor atributo que será o ponto de partida para divisão dos ramos. A escolha deste atributo é feita através do cálculo do **ganho**, isto é, é escolhido aquele cujo o ganho é maior.

Antes de se efetuar o cálculo do ganho é necessário calcular a **entropia**. Define-se como entropia o grau de pureza de um determinado conjunto de dados e pode ser representada pela equação seguinte.

Entropia

Dado um conjunto S, com instâncias pertencentes à classe i, com probabilidade p_i, temos [38]:

$$Entropia(S) = \sum p_i \log_2 p_i \tag{2.4}$$

Ganho

Dado um conjunto S e um atributo A, o ganho de informação mede a redução da entropia de S causada pela divisão dos exemplos de acordo com os valores do atributo A [38].

$$Ganho(S, A) = Entropia(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \cdot Entropia(S_v) \tag{2.5}$$

Random Forest

Apresentado o algortimo das Árvores de Decisão, é possível agora compreender o algoritmo *Random Forest*, uma vez que este é apenas uma variante do primeiro referido.

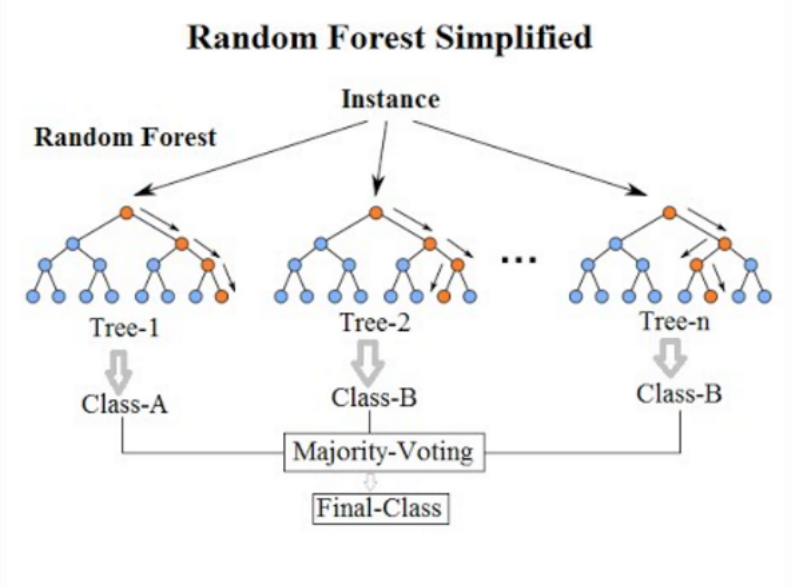


Figura 2.7: Random Forest [5].

O objetivo principal deste algoritmo é criar várias árvores de decisão dentro de um único modelo [5]. Ao treinar cada árvore em diferentes amostras, apesar de cada árvore poder ter uma grande variância em relação a um conjunto específico de dados, toda a floresta irá ter uma variância menor [39].

K Nearest Neighbours

K Nearest Neighbours (kNN) é um algoritmo de classificação simples que pode ser aplicado em multiplas situações. O **K** representa o número de pontos dos dados de treino próximos ao ponto dos dados de teste que serão usados para encontrar a classe [6].

O funcionamento do kNN consiste em escolher o melhor valor para K. De seguida, para cada valor de teste, é necessário encontrar a distância Euclidiana para todos os pontos de dados de treino, guardar essa distância numa lista e ordená-la, escolher os primeiros K pontos e por fim, atribuir uma classe ao ponto de teste baseado na maioria das classes presentes nos pontos escolhidos [6].

A Figura 2.8 representa um exemplo com valores de K diferentes.

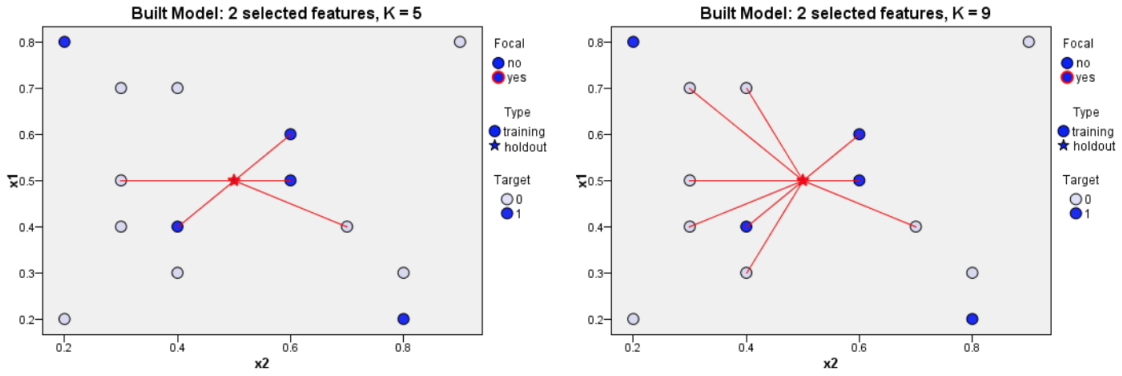


Figura 2.8: K Nearest Neighbours. [6]

SVM - Support Vector Machine

Para este algortimo a ideia é apenas criar uma linha ou um *hyperplane* que separe os dados em classes, como mostra a Figura 2.9 [40].

O objetivo neste algortimo é encontrar os pontos mais próximo das linhas das classes, a que se chamam *support vectors*. Depois dos pontos encontrados, é calculada a distâcia entre a linha e esses pontos - chama-se a essa distância **margem**. O *hyperplane* para que a margem tenha valor máximo é o ***hyperplane* ótimo** [40].

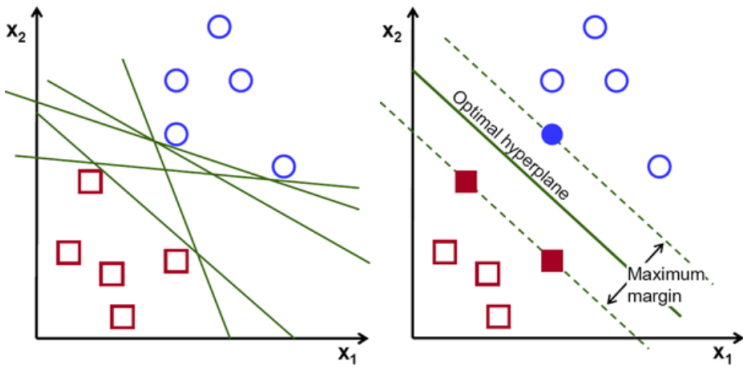


Figura 2.9: SVM [7].

Redes Neuronais

As redes neuronais artificiais são inspiradas no funcionamento do cérebro humano para o treino de modelos de *machine learning*. Tal como nos humanos, as redes neuronais estão interconectadas por um conjunto de elementos - **neurónios** - que trabalham em conjunto para responder a problemas que são tipicamente complexos [41]. Esta comparação é representada pela Figura 2.10.

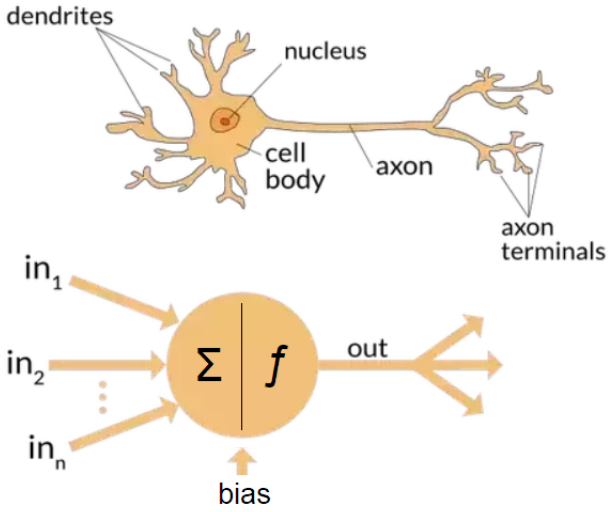


Figura 2.10: Comparação entre um neurónio humano e um neurónio artificial [8].

O treino em redes neuronárias ocorre em várias fases. A primeira fase - ***forwardpropagation*** - acontece quando os dados de treino são consumidos pela primeira camada escondida da rede. O resultado obtido desse camada é a entrada da próxima e assim sucessivamente, através de funções de ativação, até se obter uma predição. Para se calcular o erro obtido e poder melhorar os resultados, utiliza-se a ***loss function*** - segunda fase do treino. O objetivo é diminuir o erro entre os resultados das entradas e a saída do algoritmo. Num cenário

ideal, o erro seria nulo. Estando o erro calculado passa-se à próxima fase - ***backpropagation***. Aqui a informação é propagada para trás para todos os neurónios das camadas escondidas que contribuem diretamente para o resultado da saída [9].

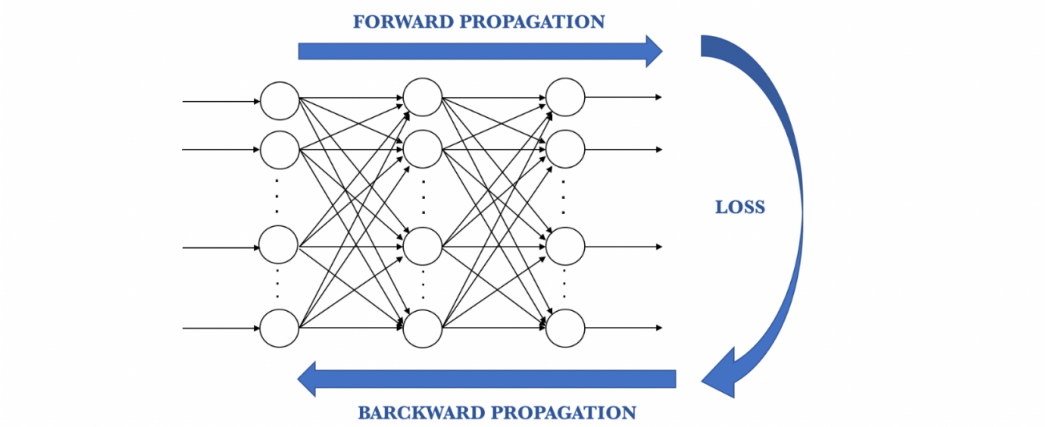


Figura 2.11: Três primeiras fases do treino em Redes Neuronais [9].

A última fase - ***weights update*** - tem como objectivo a atualização dos pesos de conecção entre os neurónios.

2.3 *Machine Learning* aplicado às cidades

De acordo com *N. Komninos*, todas as cidades inteligentes são cidades digitais, mas nem todas as cidades digitais são cidades inteligentes. Enquanto que a capacidade das cidades digitais é restrita à prestação de serviços através de canais digitais, nas cidades inteligentes é possível o planeamento, a operação e a atuação no espaço urbano de forma orquestrada [42].

A aprendizagem é um parte fundamental no processo de evolução das cidades. A capacidade de previsão permite antecipar eventos e mitigar os seus efeitos de forma a assegurar a segurança e a qualidade de vida dos cidadãos. Permitir que as cidades aprendam, usando dados históricos disponibilizados pelo ecossistema urbano, permite agilizar a resolução de vários problemas.

É importante que as novas aplicações IoT sejam criadas com recurso de Inteligência Artificial, para que a informação provinda de dispositivos e de outros sistemas possa ser utilizada na deteção de padrões de que se revelem cruciais para a gestão do espaço urbano [21].

A Inteligência Artificial aplicada às cidades tem bastante relevância e é muito promissora, se forem consideradas todas as variáveis, tecnológicas ou não, que envolvem a vida da sociedade.

A relação das Cidades Inteligentes, IoT e Machine Learning é bastante próxima, uma vez que todos estes se complementam. É através do IoT que é possível a comunicação e processamento entre sensores. São estes que executam a recolha dos dados necessários à

transformação de uma cidade numa cidade inteligente. Para esta dissertação em particular, estes são os dados também usados para a criação de um modelo de Machine Learning que é tão fundamental no processo de evolução das cidades.

As Cidades Inteligentes, ao estabelecerem uma ligação entre a tecnologia e a qualidade de vida da população em geral, podem tornar-se um meio para alcançar o equilíbrio da qualidade de vida da população [23].

2.4 Conjuntos de dados representativos

Para o desenvolvimento de um modelo de *Machine Learning* são necessários dados para a realização do seu treino. A tabela seguinte é representativa dos conjuntos de dados mais relevantes e interessantes, recolhidos a partir de uma pesquisa intensiva, onde é dada uma pequena descrição, assim como a indicação da sua origem e ano. Com base no conjunto de dados escolhido será criado um cenário da sua aplicação no contexto das *Smart Cities*.

Dataset	**Ano**	**Descrição**
LASAN: Solid Resources Abandoned Waste Collection Activity [1]	2015	Informações sanitárias de LA sobre solicitações de recolha de resíduos abandonados. Os dados mais relevantes que podem ser obtidos através deste conjunto de dados são o número, o tipo e o estado do pedido; a data da sua criação e da sua conclusão; o número do distrito do concelho.
Annual Parking Study Data [2]	2014 2015 2016	Detalhes de estacionamento pago em toda a cidade de Seattle. Os dados mais relevantes que podem ser obtidos através deste conjunto de dados são a área de estudo; a hora de ponta; o número total de espaços no estacionamento; o número total de veículos no estacionamento; o ano do estudo; se é ou não a hora de ponta; a data e hora dos dados.

[1] `https://catalog.data.gov/dataset/lasan-solid-resources-abandoned-waste-collection-activit`
[2] `https://catalog.data.gov/dataset/annual-parking-study-data`

Historical Hourly Weather Data 2012-2017 [3]	2012 2013 2014 2015 2016 2017	Informações sobre a meterologia em diversas cidades. O conjunto de dados contém informação sobre a temperatura, pressão, direção e velocidade do vento, humidade e uma breve descrição da metereologia em várias cidades.
QualAr - Avenida da Liberdade, Lisboa [4]	2017	Detalhes correspondentes à qualidade do ar. São recolhidos valores para os paramêtros de Partículas $< 10\mu m$, Dióxido de Azoto, Óxidos de Azoto, Monóxido de Carbono e Monóxido de Azoto.
Road traffic accidents [5]	2017	Detalhes relativos a acidentes que permitem perceber o número de veículos envolvidos, a data e hora, as condições de luminosidade e de metereologia, tipo do veículo, a severidade do acidente, o género e a idade dos envolvidos.
Traffic - Annual volume of traffic [6]	2000 a 2017	Indica o volume total de tráfego num dado ano e região para diferentes tipos de veículos.
Road Casualties by Severity [7]	2010 2011 2012 2013 2014	Fornece informação relativa a acidentes indicando as coordenadas, o grau de severidade, o género e a idade dos envolvidos, a data, o tipo de veículo e ainda a localização do pedestre se existir algum envolvido.
Road Safety Data [8]	2017	Neste conjunto de dados estão detalhados vários acidentes tendo em consideração os veículos envolvidos, as possíveis causas, e os detalhes dos acidentes.

[3] https://www.kaggle.com/selfishgene/historical-hourly-weather-data

[4] https://qualar.apambiente.pt/qualar

[5] http://www.europeandataportal.eu/data/en/dataset/road-traffic-accidents/resource/124c194b-e0f5-411b-a234-6c9ebeb1506d?inner_span=True

[6] https://www.dft.gov.uk/traffic-counts/download.php

[7] https://data.london.gov.uk/dataset/road-casualties-severity-borough

[8] https://data.gov.uk/dataset/cb7ae6f0-4be6-4935-9277-47e5ce24a11f/road-safety-data

Air Quality in Madrid [9]	2001 a 2018	Informações sobre as concentrações de determinados componentes, que permitem identificar a qualidade do ar.
Bike Sharing Dataset [10]	2011 2012	Com este conjunto de dados é possível relacionar determinadas variáveis como as condições meterologicas, temperatura, dia da semana, hora do dia, entre outras, com o número de bicicletas que são alugadas.
Road Crash Data [11]	2012 2013 2014 2015 2016 2017	Detalhes sobre acidentes ocorridos na Austrália. Neste conjunto de dados é possível perceber quantos veículos estão envolvidos, assim como determinadas caracteristicas de quem está envolvido, entre outros.
Crash Stats - Data Extract [12]	2000 2001 2002 2003 2004 2005	Dados sobre determinados acidentes que indicam as condições metereológicas, as condições da estrada, detalhes sobre as pessoas evolvidas assim como dos veículos envolvidos, entre outros.
Killed or Seriously Injured (KSI) Toronto Clean [13]	2007 a 2017	Neste conjunto de dados são apresentados dados referentes a determinados acidentes, indicando que estes são ou não fatais. É possível ainda verificar as condições em que os acidentes ocorreram, como luminosidade, se as colisões se verificaram com veículos, motociclos ou peões, se se verificaram semáforos vermelhos ativos, presença ou não de álcool.
1.6 million UK traffic accidents [14]	2000 a 2016	Dados relativos a acidentes cujo os atributos apresentam as condições em que estes ocorreram, bem como a média do tráfego diário anual entre os anos de 2000 e 2016.

[9] https://www.kaggle.com/decide-soluciones/air-quality-madrid
[10] https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset
[11] https://data.sa.gov.au/data/dataset/road-crash-data
[12] https://www.data.vic.gov.au/data/dataset/crash-stats-data-extract
[13] https://www.kaggle.com/jrmistry/killed-or-seriously-injured-ksi-toronto-clean
[14] https://www.kaggle.com/daveianhickey/2000-16-traffic-flow-england-scotland-wales

UCI ML Air Quality Dataset [15]	2004 a 2005	Conjuntos de dados relativos à qualidade do ar em determinados dias e horas. Através destes dados, com base em diferentes concentrações de substâncias presentes no ar, é possível perceber qual o nível de CO.
U.S. Pollution Data [16]	2000 a 2016	Este conjunto de dados permite perceber como é que se encontra a qualidade do ar em determinada região e determinado dia, tendo em conta as concentrações de alguns dos maiores poluentes existentes.
UK Road Safety: Traffic Accidents and Vehicles [17]	2004 a 2016	Fornece informação relativa a acidentes indicando as coordenadas, o grau de severidade, o género e a idade dos envolvidos, a data, o tipo de veículo e ainda outro tipo de informações que permitem determinar a severidade do acidente.

[15] https://www.kaggle.com/nishantbhadauria/datasetucimlairquality

[16] https://www.kaggle.com/sogun3/uspollution

[17] https://www.kaggle.com/tsiaras/uk-road-safety-accidents-and-vehicles

Capítulo 3

Cenários

3.1 Possíveis cenários

Os cenários a seguir apresentados foram criados com base em alguns conjuntos de dados encontrados. No entanto, o cenário escolhido para este projeto acabou por ser o cenário apresentado na secção 3.2, por ser o que se adequava e fazia mais sentido para o conjunto de dados escolhido.

Priorização de acidentes

São frequentes as vezes em que há falta de meios para o socorro de vítimas em acidentes de viação. Quando são várias as ocorrências, pode acontecer ser necessário priorizar os meios de socorro, de modo a que sejam socorridas as vítimas dos acidentes com maior grau de severidade. No entanto, nem sempre é fácil, dados os imensos fatores que podem determinar este grau, perceber com a rapidez necessária, qual a ocorrência com maior severidade.

Ao aplicarmos um modelo de *Machine Learning* aos atributos que permitem fazer esta previsão, será possível priorizar mais eficazmente as ocorrências.

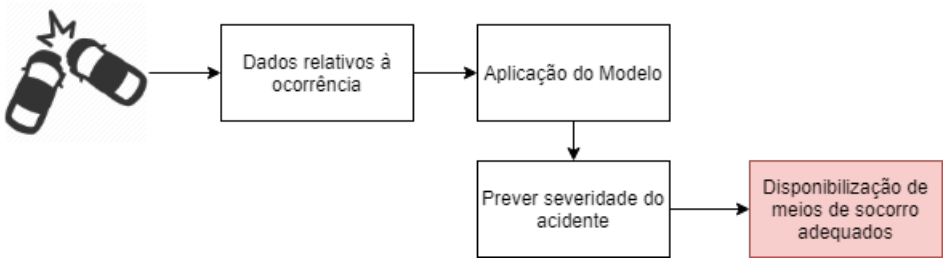


Figura 3.1: Cenário 1 - Priorização de acidentes.

Deteção de causas de acidentes

Ainda no âmbito dos acidentes de viação, é difícil por vezes determinar as suas causas. No entanto, se se analisar um conjunto de características, é possível que se consiga perceber as causas que originaram o acidente. Este tipo de predição é útil para casos em que existem dúvidas do culpado da ocorrência, por exemplo.

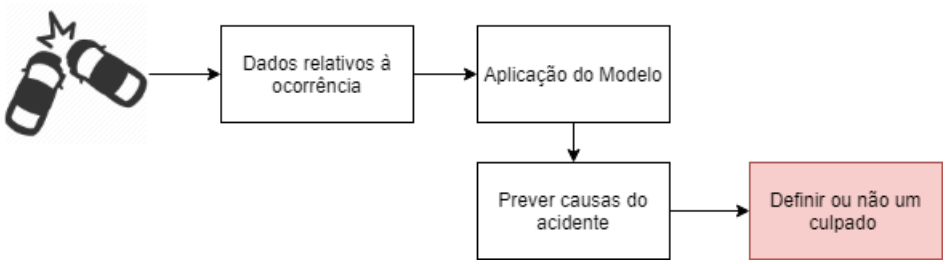


Figura 3.2: Cenário 2 - Detetar causas de acidentes.

Qualidade do ar

Cidades em que existe uma grande quantidade de tráfego e atividade fabril tendem a ser cidades cujo o ar é mais poluído, como são o caso as grandes cidades da China, por exemplo.

A qualidade do ar varia todos os dias, sendo que há dias em que se torna extremamente prejudicial à saúde da população a inalação do ar contaminado. Prevendo, com base em vários fatores, a qualidade do ar para determinado dia, é possível que sejam aplicadas medidas para que este não se torne tão prejudicial assim, e de modo a que se minimizem estes dias em que se torna irrespirável.

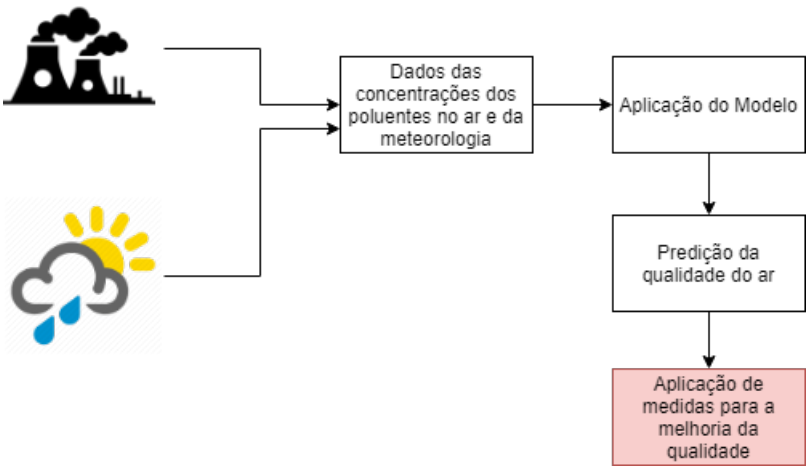


Figura 3.3: Cenário 3 - Qualidade do Ar.

3.2 História do cenário escolhido

O José é trabalhador numa empresa um pouco distante de sua casa e por isso utiliza todos os dias o seu veículo para se deslocar ao emprego. Num desses dias chovia bastante e o José estava atrasado para uma reunião importante, motivo este que o levou a exceder a velocidade, e como as condições meteorológicas não eram favoráveis, despistou-se.

Imediatamente, pessoas que passavam pela mesma estrada chamaram as autoridades necessárias para socorrer o José, informando como ocorreu o acidente, o local, entre outras informações que as autoridades de socorro questionam. Como também existem alguns sensores espalhados por esta cidade, as autoridades já têm acesso às condições meteorológicas, sabem qual seria o limite de velocidade da estrada e o tipo de estrada, as condições de luminosidade, entre outras coisas. É através de todos estes dados, bem como pelas informações dadas por quem chamou pelo socorro, que numa fase inicial, e até chegarem as autoridades ao local, se consegue prever a severidade do acidente.

Pela estrada existem alguns painéis informativos e nestes casos são utilizados para informar que existe um acidente por perto e dar indicações de como os condutores devem agir. Desta forma, é possível avisar todos os condutores que por ali passam que devem ter uma atenção redobrada ou sugerir novas rotas para que se evitem novos acidentes.

Capítulo 4

Arquitetura

Neste capítulo é apresentada a arquitetura do sistema, assim como a descrição das suas funcionalidades chave da mesma de modo a facilitar o dia-a-dia nas cidades.

A arquitetura das *Learning Cities* engloba duas entidades funcionais distintas que se complementam entre elas. A primeira entidade funcional - **City Learning Model** - inclui todos os componentes necessários para a construção do modelo de *Machine Learning* para ser adotado pela cidade, assim como a permanente evolução do desenvolvimento do mesmo. A segunda entidade funcional - **City Runtime** - é constituída por elementos necessários à gestão diária da cidade, permitindo agir nas infraestruturas e equipamentos tendo em conta o contexto do espaço urbano. Ambas estanciam uma plataforma de gestão de dados para guardar e aplicar persistência tanto aos dados da cidade que chegam dos sensores como aos dados analisados. A Figura 4.1 representa genericamente a arquitetura do sistema.

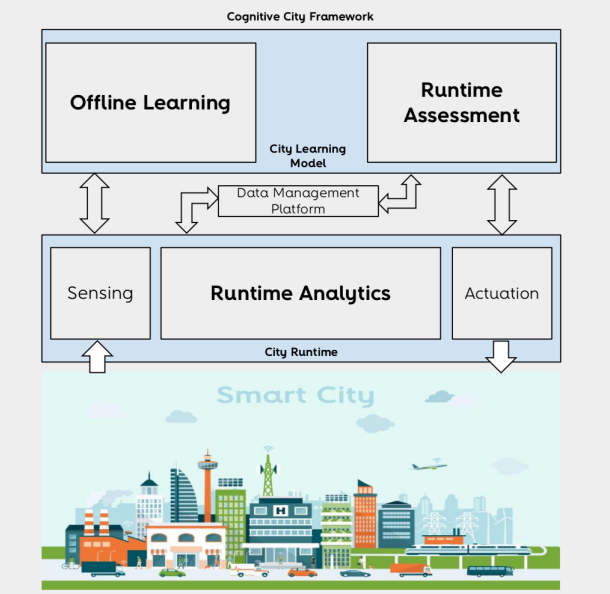


Figura 4.1: Arquitetura genérica do sistema.

4.1 City Learning Model - Modelo de Aprendizagem da Cidade

O *City Learning Model* tem o objectivo de gerir o ciclo de vida dos modelos a serem aplicados nas cidades. Engloba todo o processo de aprendizagem, a construção da componente nos sistemas das cidades, assim como a sua avaliação tendo em consideração as novas entradas de dados recolhidas pelos sistemas urbanos existentes.

No *City Learning Model* estão englobados dois grandes blocos: ***Offline Learning*** e o ***Runtime Assessment***, como mostra a Figura 4.2.

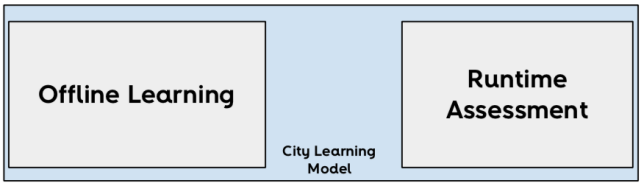


Figura 4.2: City Learning Model.

4.1.1 Offline Learning

A entidade funcional aqui apresentada (Figura 4.4) - *Offline Learning* - inclui as tarefas principais para as cidades com capacidade de aprendizagem. Esta é uma componente *offline* abrangendo todos os estágios para o **treino do modelo** baseado no conjunto de dados disponível. Assume-se como **preparação de dados** o módulo onde os dados são tratados, dando lugar a um conjuto de dados organizado e pronto a ser utilizado pelos algoritmos de *Machine Learning*, incluido a verificação de valores em falta, lidar com *outliers*, corrigir duplicados, normalização e divisão dos dados em treino e teste. Este módulo inclui ainda os processos adequados para a **criação de modelos** baseados num conjunto de algortimos de *Machine Learning* (Figura 4.3).

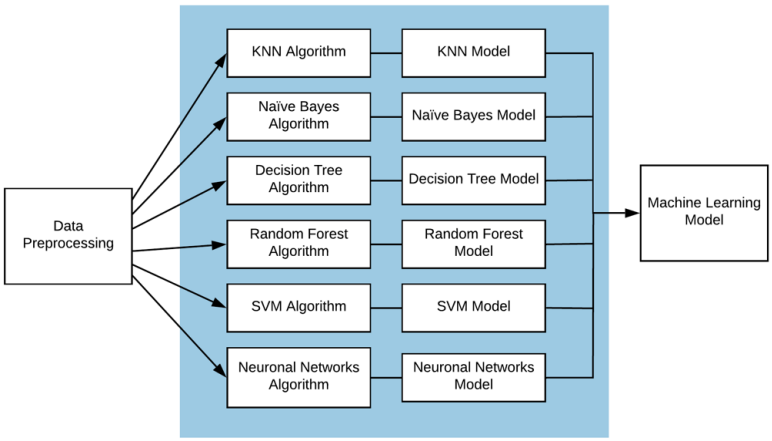


Figura 4.3: Construção do modelo de Machine Learning.

Os dados de treino são usados como entrada para os algoritmos *KNN, Naïve Bayes, Decision Trees, Random Forest, SVM* e *Neural Networks* resultando em diferentes modelos disponíveis para serem aplicados a predições.

Cada modelo é **avaliado** usando principalmente a matriz de confusão para determinar a prestação dos modelos de classificação criados.

Por fim, o modelo mais apropriado é selecionado e **implementado** num ambiente de execução em tempo real, tornando possível prever situações criticas da cidade e antecipar a mitigação de ações. Este módulo é representado ao detalhe pela Figura 4.4.

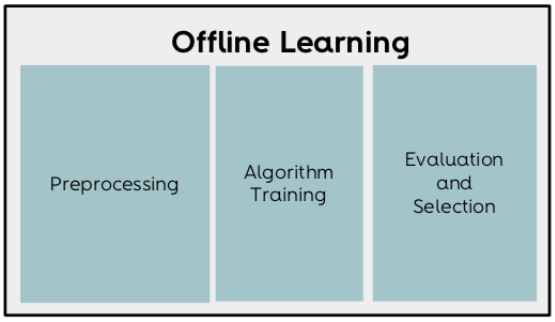


Figura 4.4: Offline Learning.

4.1.2 Runtime Assessment

O módulo *Runtime Assessment* visa garantir que o modelo a ser implementado permaneça adaptdado à dinâmica da cidade. Para isso, utiliza o módulo de **Verification** para comparar a previsão feita com o resultado real, ou seja, verifica se a previsão foi correta ou não. Além disso, executa o módulo **Evalutation** para verificar o nível de precisão do modelo. Quando esta começa a diminuir, é necessário refazer o modelo tendo em conta todos os novos dados recolhidos. Este módulo é representado pela Figura 4.5.

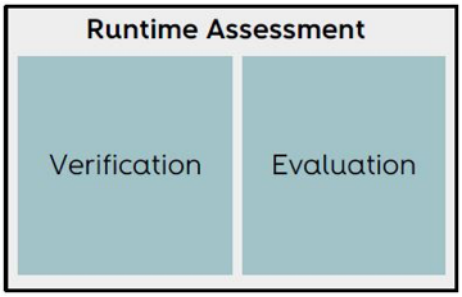


Figura 4.5: Runtime Assessment.

4.2 City Runtime

O *City Runtime* tem o objectivo de gerir a infraestrutura da cidade em tempo real, de forma facilitar a vida dos cidadãos. É utilizado o modelo que foi desenvolvido no módulo *Offline Learning* para a gerir as ocurrências na cidade. Os dados são recolhidos através de sensores espalhados pelo espaço urbano e analisados de modo a obter a predição de uma determinada ocurrência. De acordo com o resultado será feita uma recomendação de uma ação que mudará o estado de uma ou mais infraestruturas da cidade.

Estão inseridos nele três grandes blocos: ***Sensing***, ***Runtime Analytics*** e ***Actuation***, como se pode ver na Figura 4.6

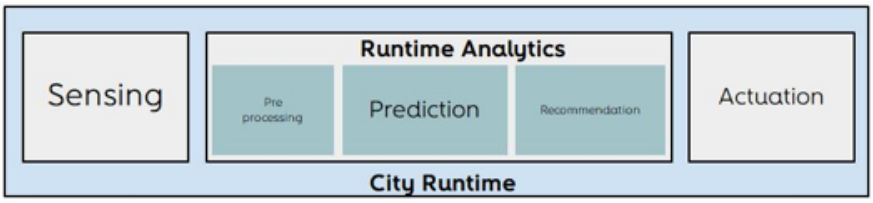


Figura 4.6: City Runtime.

4.2.1 Sensing

Esta entidade é a responsável por recolher medições físicas relacionadas com as infraestruturas da cidade. Os dados são recolhidos através de sensores disponíveis na cidade. O módulo de *Sensing* irá converter essas ocorrências (quandidade de tráfego, medição da temperatura, qualidade do ar, entre outros) em dados que serão guardados na ***Data Management Plataform*** para análise futura.

4.2.2 Runtime Analitycs

O *Runtime Analytics* é o "cérebro"do sistema de tempo de execução. Este obtém a informação detectada e a prepara-se para o módulo de machine learning. A amostra de dados tratada é usada para prever um evento específico; entra na entidade **Prediction**, que executa o modelo de machine learning implementado no sistema da cidade, permitindo obter o resultado da classificação ou regressão. Dependendo do resultado, é feita uma recomendação específica para atualizar o estado da infraestrutura da cidade. A Figura 4.7 apresenta o módulo de análise de tempo de execução.

4.2.3 Actuation

Esta entidade - *Actuation* é responsável pela aplicação das recomendações fornecidas. Aqui, é fechado o ciclo, enviando as instruções apropriadas a infraestrutura de modo a adaptar a cidade às suas necessidades instantâneas.

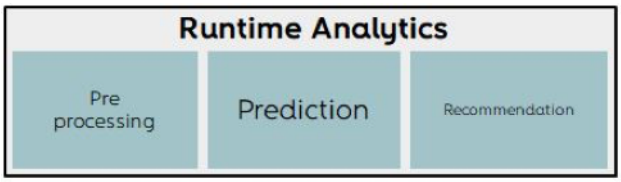


Figura 4.7: Runtime Analytics.

4.3 City Data Management Platform

A principal funcionalidade da *City Data Management Platform* é fazer a mediação de dados entre diferentes entidades do sistema. É uma API aberta baseada em *cloud* capaz de manter diferentes tecnologias e protocolos, facilitando a integração do sistema end-to-end. Suporta tanto padrões de troca de mensagens de *request* e *response* como *publish* e *subscribe*. A *City Data Management Platform* permite vincular diferentes domínios de cidades, permitindo o armazenamento e partilha de dados da cidade, resolvendo os problemas típicos da cidade. Todo o ciclo de vida de dados da cidade é gerido aqui e disponibilizado permanentemente para todas as entidades autorizadas.

4.4 Arquitetura Detalhada

A Figura 5.17 representa a arquitetura detalhada que foi apresentada anteriormente.

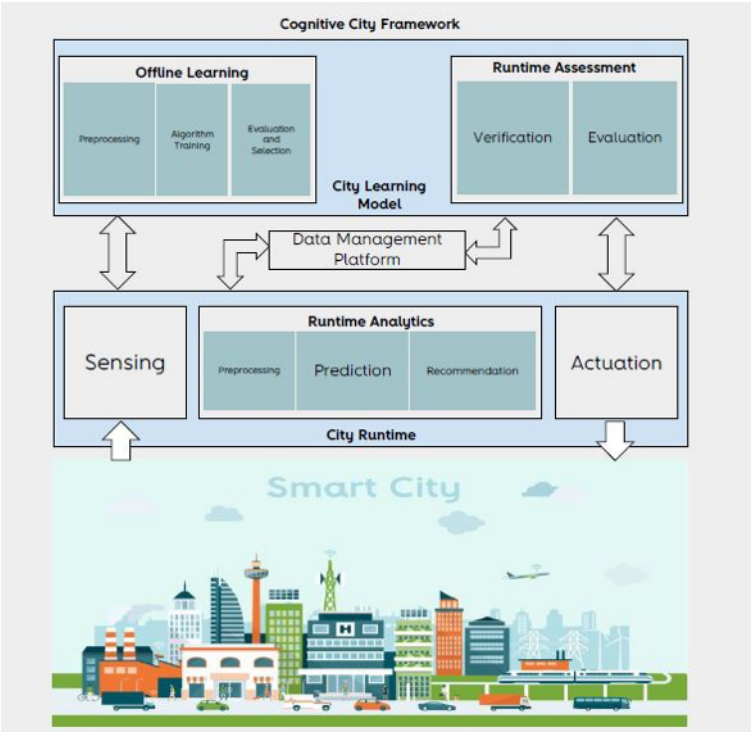


Figura 4.8: Arquitetura detalhada do sistema.

4.5 Fluxo de Dados

O fluxo de dados do sistema é reprensentado pela Figura 4.9.

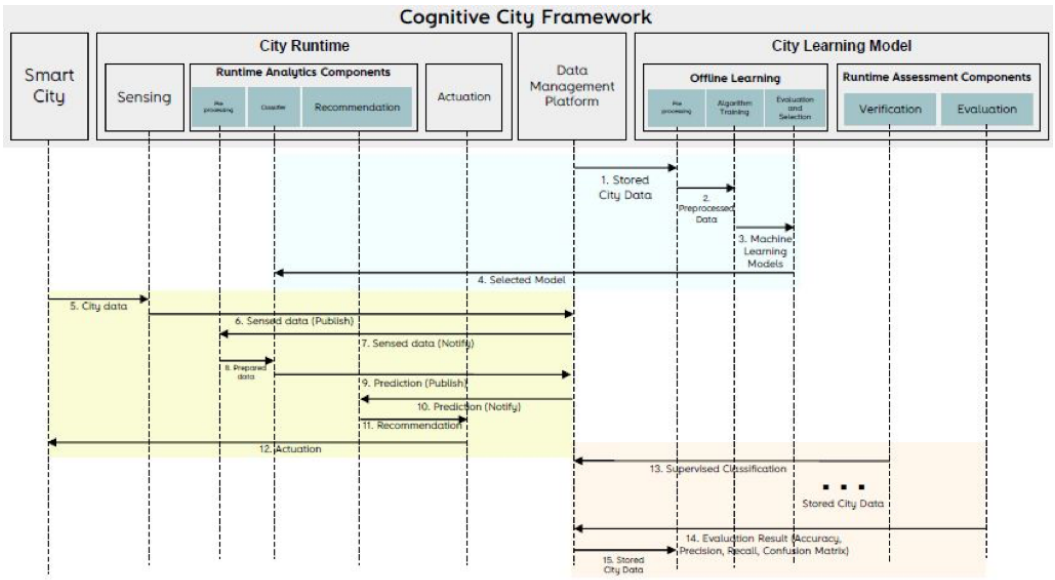


Figura 4.9: Fluxo da Cognitive City Framework.

4.5.1 Fase da Aprendizagem

Esta fase engloba todos os passos necessários para criar um modelo apropriado para prever situações específicas na cidade. O bloco *City Learning Model* usa os dados armazenados na *Data Management Platform* para o processo de modelação **(etapa 1)**. Na entidade *Offline Learning*, os dados recolhidos são tratados no pré-processamento, dando origem a um conjunto de dados preparados para a criação do modelo **(etapa 2)**. Como pode ser visto na Figura 4.9, o conjunto de dados funciona como entrada para diferentes algoritmos de *machine learning*, como Naïve Bayes, Random Forest, Redes Neuronais, entre outros. Para cada algoritmo, é criado um modelo pronto para avaliação **(etapa 3)**. Cada modelo é testado e avaliado, com base no desempenho, e o selecionado é implementado no bloco *City Runtime* para ser usado em tempo de execução **(etapa 4)**.

4.5.2 Fase de Execução

A fase de execução controla a infraestrutura da cidade de forma dinâmica. Ela afeta a vida quotidiana dos cidadãos, adaptando antecipadamente o espaço urbano às necessidades da cidade. Este recorre ao modelo de *Machine Learning* construído para prever situações da cidade e mitigá-las antecipadamente.

Os sensores espalhados pela cidade estão a disponibilizar periodicamente dados da cidade.

Esses dados são recolhidos e definidos num modelo de informações comum **(etapa 5)**, sendo então enviados para a plataforma de gestão de dados **(etapa 6)**, que armazena e mantém as informações. O *Data Management Platform* notifica o módulo de Pré-processamento no Runtime Analytics **(etapa 7)**, que trata os dados tornando-os prontos para o módulo *Prediction*. Este executa o modelo implementado; com base na entrada irá permitir antecipar situações críticas. A previsão feita é publicada no *Data Management Analytics* **(etapa 8)**, que, por sua vez, notifica o módulo da recomendação **(etapa 9)**. Com base na previsão recebida, a recomendação define as ações necessárias para mitigar o problema e envia-as ao módulo de atuação (etapa 10). Finalmente, a situação muda o estado da infraestrutura **(etapa 11)**, adaptando a cidade à dinâmica da vida quotidiana.

4.5.3 Fase da Avaliação

Este módulo é responsável pela verificação da veracidade da previsão, assim como pela avaliação do modelo. De forma supervisionada, verifica-se se a predição feita foi correta ou não **(passo 12)**, sendo o resultado armazenado na plataforma de gestão de dados. Periodicamente, o módulo de Avaliação reúne todas as informações **(etapa 13)** e compara a previsão com os valores verificados, a fim de verificar os resultados do modelo. Esse resultado é publicado na plataforma **(etapa 14)**, que notifica o *Offline Learning* sobre os novos resultados **(etapa 15)**. Dependendo das políticas do sistema, o resultado da avaliação do modelo pode acionar a criação de um novo modelo a ser implementado na cidade, de acordo com as necessidades dos cidadãos.

Dependendo do cenário, precisamos avaliar as métricas como exatidão, precisão, *recall* e matriz de confusão. Se no cenário é importante eliminar falsos negativos, a métrica mais importante é a *recall*. Essa métrica indica quão bem o modelo identifica casos positivos corretamente. [43] Por exemplo, na detecção de pacientes doentes, o custo associado ao falso negativo pode ser muito alto para o paciente. É importante analisar a matriz de confusão também. Se no cenário for importante prever a maioria dos casos corretamente, é importante analisar a exatidão e a precisão. Os melhores resultados são quando as métricas estão próximas de 1.

Capítulo 5

Implementação

Este caítulo pretende apresentar os detalhes da implementação deste projeto, incluíndo todos os processos pelo qual foi necessário passar até se obter o resultado final. Especificamente, será descrito o processo de seleção do conjunto de dados a ser utilizado, o pré-processamento desse mesmo conjunto de dados e ainda como se procedeu ao seu treino.

Seguindo a metedologia de trabalho apresentada anteriormente, este capítulo irá incluir tanto parte da fase **Design** como a fase **Criar**.

5.1 Características do Conjunto de Dados Escolhido

Após uma pesquisa exaustiva de diversos conjuntos de dados, escolheu-se o conjunto de dados relativo ao cenário da **Priorização de Acidentes**. Um dos factores que pesou na altura da escolha foi o facto de este ser uma amostra representativa. Tem cerca de dois milhões de entradas, sendo possível ter dados suficientes para se tirarem conclusões válidas e neste caso em específico treinar de modo mais assertivo os algoritmos.

O conjunto de dados é a junção de duas tabelas: uma que contém informação sobre os acidentes e outra sobre os veículos envolvidos nos mesmos. As duas são juntas através de uma *feature* em comum (***Accident_Index***).

5.2 Pré-processamento

Inicialmente, e mesmo antes de se conhecerem os dados que seriam utilizados, fez-se um módulo genérico, que pudesse funcionar com diferentes conjuntos de dados.

Este módulo consiste numa classe **PreProcessing** que contém várias funções, representando cada uma delas diferentes fases do pré-processamento dos dados. A função ***get_data*** não só chama as restantes funções da classe como faz a divisão dos dados em descritivos e resultados (*descriptive* e *target*).

Numa primeira fase, procedeu-se ao tratamento dos dados onde existiam campos em branco ou *NaN*. Em dados cujo o tipo é *datetime64[ns]* e estes se encontrem com as caracteristicas anteriormente referidas, é feita a eliminação dessa linha, por se considerar não ter informação sobre o momento em que foi feita a recolha, dado este que é relevante na maioria dos cenários. Por outro lado, quando o tipo de dados é *int64* ou *float64* é feita a substituição desse valor inválido pela média de todos os outros valores.

```
def fill_nan(self, data):
  data.fillna(data.mean(), inplace=True)
```

Se o tipo for *object*, então o campo inválido é substituído por 0.

Quando a divisão dos dados em *descriptive* e *target* é feita, é preciso passar às próximas fases do pré-processamento. No entanto, agora apenas os dados *descriptive* necessitam desse pré-processamento. Posto isto, sucedeu-se a conversão dos elementos categóricos em numéricos. A função ***categorical_to_numeric*** é a responsável por isso.

```
def categorical_to_numeric(self, descriptive):
  labelEncoder = LabelEncoder()
  for index, i in enumerate(descriptive[0]):
    if type(i) is str:
      descriptive[:,index] = labelEncoder.fit_transform(descriptive[:,index])
  return descriptive
```

Após a conversão feita, o próximo passo é realizar a *demultiplexing*. É necessário que os valores de um determinado atributo, que passou de categórico a numérico, possuam a mesma magnitude, de forma a não influenciarem na altura do treino do algoritmo. É na função ***data_demultiplexing*** que este processo acontece.

```
def data_demultiplexing(self, descriptive, i, categorical_feature_mask):
  oneHotEncoder = OneHotEncoder(categorical_features = categorical_feature_mask
    [:])
  descriptive = oneHotEncoder.fit_transform(descriptive).toarray()
  return descriptive
```

Concluída esta etapa, é feita então a padronização dos atributos, responsabilidade da função ***scalling***, sendo que agora a variância passa a ser unitária.

```
def scalling(self, descriptive):
  scaler = StandardScaler()
  descriptive[:] = scaler.fit_transform(descriptive[:])
  return descriptive
```

Por fim, é crucial a divisão dos dados em dados de treino e teste. Para o caso, optou-se por 75% dos dados serem de treino e 25% de teste, por ser tipicamente a divisão que resulta melhor com a maioria dos conjuntos de dados. A função ***train_test*** é a responsável por essa divisão.

Nesta fase, os dados estão prontos a serem utilizados pelos algoritmos.

Aquando da escolha do conjuntos de dados, é normal ser necessário adaptar algumas características.

Escolhido o conjunto de dados a ser utilizado, foi aplicado o módulo de pré-processamento, com as devidas alterações necessárias.

5.2.1 Eliminanção de dados em branco ou *NaN*

Como descrito acima, a primeira fase foi o tratamento dos dados em que existiam campos em branco ou *NaN*. As tabelas seguintes (Tabela 5.1 e Tabela 5.2) descrevem o conjunto de dados antes e depois de terem sido tratados os dados com campos com as características referidas. Para efeitos de apresentação apenas estão representados nas tabelas os atributos que tinham pelo menos um campo em branco ou *NaN*.

Colunas	Total
1st_Road_Number	1
2nd_Road_Class	827937
2nd_Road_Number	18927
Did_Police_Officer_Attend _Scene_of_Accident	114
Latitude	124
Location_Easting_OSGR	124
Location_Northing_OSGR	124
Longitude	125
LSOA_of_Accident _Location	139207
Pedestrian_Crossing-Human_Control	654
Pedestrian_Crossing-Physical_Facilities	1370
Speed_limit	65
Time	146
InScotland	44
Age_of_Vehicle	337962
Driver_IMD_Decile	689290
Engine_Capacity_.CC.	250317
make	110845
model	299357
Propulsion_Code	233598
Vehicle_Location.Restricted _Lane	1124

Tabela 5.1: Antes do pré-processamento.

Colunas	Total
1st_Road_Number	0
2nd_Road_Class	0
2nd_Road_Number	0
Did_Police_Officer_Attend _Scene_of_Accident	0
Latitude	0
Location_Easting_OSGR	0
Location_Northing_OSGR	0
Longitude	0
LSOA_of_Accident _Location	0
Pedestrian_Crossing-Human_Control	0
Pedestrian_Crossing-Physical_Facilities	0
Speed_limit	0
Time	0
InScotland	0
Age_of_Vehicle	0
Driver_IMD_Decile	0
Engine_Capacity_.CC.	0
make	0
model	0
Propulsion_Code	0
Vehicle_Location.Restricted _Lane	0

Tabela 5.2: Depois do pré-processamento.

Como é possível observar nestas, deixou de se ter dados com estas características, promo-

vendo posteriormente mais eficiência no treino dos algortimos.

5.2.2 Feature Engineering

Para o conjunto de dados em questão, tendo este mais de 50 atributos, fazia sentido selecionar aqueles que mais podiam influenciar o resultado do algoritmo - *manual feature engineering*.

Usando para efeitos de teste dos atrbituos o algoritmo KNN, pode verificar-se uma *accuracy* de 0.8936 para quando não existe seleção de atributos. Para o cenário em questão para além da *accuracy*, uma métrica bastante importante de analisar é a *Recall*.

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{5.1}$$

$$Recall = \frac{(TP)}{(TP + FN)} \tag{5.2}$$

Sendo que:

- **TP:** corresponde aos verdadeiros positivos;
- **TN:** corresponde aos verdadeiros negativos;
- **FP:** corresponde aos falsos positivos;
- **FN:** corresponde aos falsos negativos;

A métrica *Recall* indica o quanto o modelo identifica os casos positivos corretamente [43]. Por este motivo é muito eficaz para eliminar os falsos negativos, que no caso podem tornar-se bastantes críticos, ao classificar um acidente como ligeiro quando na verdade é grave. É também por este motivo que, ao utilizar a função ***recall_score*** disponível na biblioteca *sklearn*, se deve utilizar *macro-averaged* e não *micro-averaged*. Ao utilizar-se *micro-averaged* não se está a ter em conta o facto do existirem muito mais dados de uma determinada classe do que das outras. Por este motivo, o resultado, neste caso, é relativamente bom, com um *recall* de cerca de 0.89 ao invés de quando se usa *macro-averaged* em que se obtém cerca de 0.33, o que é um resultado péssimo, principalmente para o cenário em questão. Estes resultados são obtidos sem qualquer seleção de atributos.

Apesar do mau resultado, é com base neste que o estudo da seleção dos atributos foi feito, de forma a não só melhorar o resultado como a perceber que atributos influenciam mais no resultado final. Os valores do *recall* são arredondados às 5 casas decimais.

Região	Accuracy | Recall			
	Com	**Sem**	**Com**	**Sem**
1st_Road_Class	0.8936	0.894	0.33461	0.33476
1st_Road_Number	0.894	0.894	0.33476	0.33476
2nd_Road_Class	0.894	0.894	0.33476	0.33476

2nd_ Road_ Number	0.894	0.8936	0.33476	0.33461
Carriageway_ Hazards	0.894	0.894	0.33476	0.33476
Day_ of_ Week	0.894	0.894	0.33476	0.33476
Did_ Police_ Officer_ Attend_ Scene _ of_ Accident	0.894	0.894	0.33476	0.33476
Junction_ Control	0.894	0.8948	0.33476	0.33629
Junction_ Detail	0.8948	0.8944	0.33629	0.33367
Light_ Conditions	0.8948	0.894	0.33629	0.33476
Local_ Authority_ (District)	0.8948	0.8948	0.33629	0.33505
LSOA_ of_ Accident_ Location	0.8948	0.8896	0.33629	0.33559
Number_ of_ Casualties	0.8948	0.8948	0.33629	0.33629
Number_ of_ Vehicles	0.8948	0.8948	0.33629	0.33629
Pedestrian_ Crossing-Human_ Control	0.8948	0.8948	0.33629	0.33629
Pedestrian_ Crossing-Physical_ Facilities	0.8948	0.8948	0.33629	0.33629
Road_ Surface_ Conditions	0.8948	0.8944	0.33629	0.33491
Road_ Type	0.8948	0.8944	0.33629	0.33491
Special_ Conditions_ at_ Site	0.8948	0.8948	0.33629	0.33629
Speed_ limit	0.8948	0.8944	0.33629	0.33491
Urban_ or_ Rural_ Area	0.8948	0.8948	0.33629	0.33629
Weather_ Conditions	0.8948	0.8936	0.33629	0.33338
Age_ Band_ of_ Driver	0.8948	0.8956	0.33629	0.33783
Age_ of_ Vehicle	0.8956	0.8956	0.33783	0.33783
Driver_ Home_ Area_ Type	0.8956	0.8956	0.33783	0.33783
Driver_ IMD_ Decile	0.8956	0.896	0.33783	0.33797
Hit_ Object_ in_ Carriageway	0.896	0.896	0.33797	0.33797
Hit_ Object_ off_ Carriageway	0.896	0.896	0.33797	0.33797
Journey_ Purpose_ of_ Driver	0.896	0.8804	0.33797	0.35904
Junction_ Location	0.8804	0.8728	0.35904	0.35991
make	0.8728	0.866	0.35991	0.35739
model	0.8728	0.8936	0.35991	0.33461
Propulsion_ Code	0.8728	0.8788	0.35991	0.33899
Sex_ of_ Driver	0.8728	0.8724	0.35991	0.35977
Latitude	0.8728	0.8728	0.35991	0.35991
Local_ Authority_ (Highway)	0.8728	0.8728	0.35991	0.35745
Location_ Easting_ OSGR	0.8728	0.872	0.35991	0.35839
Location_ Northing_ OSGR	0.8728	0.8728	0.35991	0.35991
Longitude	0.8728	0.8716	0.35991	0.35700
Police_ Force	0.8728	0.8932	0.35991	0.33323
Year_ x	0.8728	0.8728	0.35991	0.35991
InScotland	0.8728	0.8728	0.35991	0.35991
Engine_ Capacity_ .CC.	0.8728	0.872	0.35991	0.35838
Skidding_ and_ Overturning	0.8728	0.8728	0.35991	0.35869

Towing_and_Articulation	0.8728	0.8728	0.35991	0.35991
Vehicle_Leaving_Carriageway	0.8728	0.8728	0.35991	0.35869
Vehicle_Location.Restricted_Lane	0.8728	0.8728	0.35991	0.35991
Vehicle_Manoeuvre	0.8728	0.8948	0.35991	0.33629
Vehicle_Reference	0.8728	0.8728	0.35991	0.36115
Vehicle_Type	0.8728	0.8792	0.36115	0.35736
Was_Vehicle_Left_Hand_Drive	0.8728	0.8728	0.36115	0.36115
X1st_Point_of_Impact	0.8728	0.8732	0.36115	0.36007
Year_y	0.8728	0.8732	0.36115	0.36007

Tabela 5.3: Accuracy e *Recall* com e sem determinados atributos.

Através da análise dos resultados obtidos para a *Recall* com e sem cada um dos atributos, é possível peceber aqueles cuja sua presença influência positivamente o resultado. Por este motivo, foram retirados todos os atributos que não faziam variar o valor desta métrica, assim como aqueles que pioravam o mesmo.

Desta forma, foi criada uma função - ***clean_dataset*** - que tem como objetivo a limpeza dos dados, retirando tanto linhas cuja a falta de dados não pode ser substituída bem como todos os atributos que se pensam não ter tanta influência na predição.

Como se pode verificar atráves os gráficos das Figuras 5.1 e 5.2, diminui-se assim o número de colunas de 58 para 26 e o número de linhas de 2058408 para 1541884.

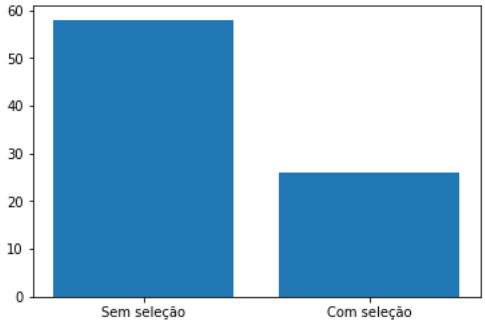


Figura 5.1: Número de atributos/colunas sem e com seleção.

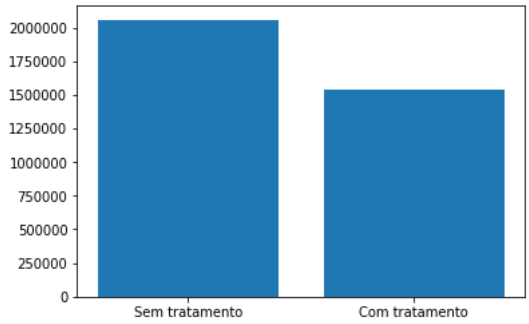


Figura 5.2: Número de linhas com e sem tratamento.

Testou-se ainda a seleção de atributos atráves da matriz de correlação (Figura 5.3), eliminando-se os atributos que tivessem uma correlação igual ou superior a 0.9.

No entanto, a matriz de correlação induz, por vezes, a erros nesta seleção por eliminar atributos com muita relevância para a predição. Neste caso, obteve-se um pior resultado com esta seleção do que sem e portanto este método foi descartado.

Tendo isto concluído e feita a divisão entre os dados descritivos e os resultados, é feita a conversão dos dados de categoricos para numéricos, a desmultiplexação e a sua normalização , como foi explicado anteriormente.

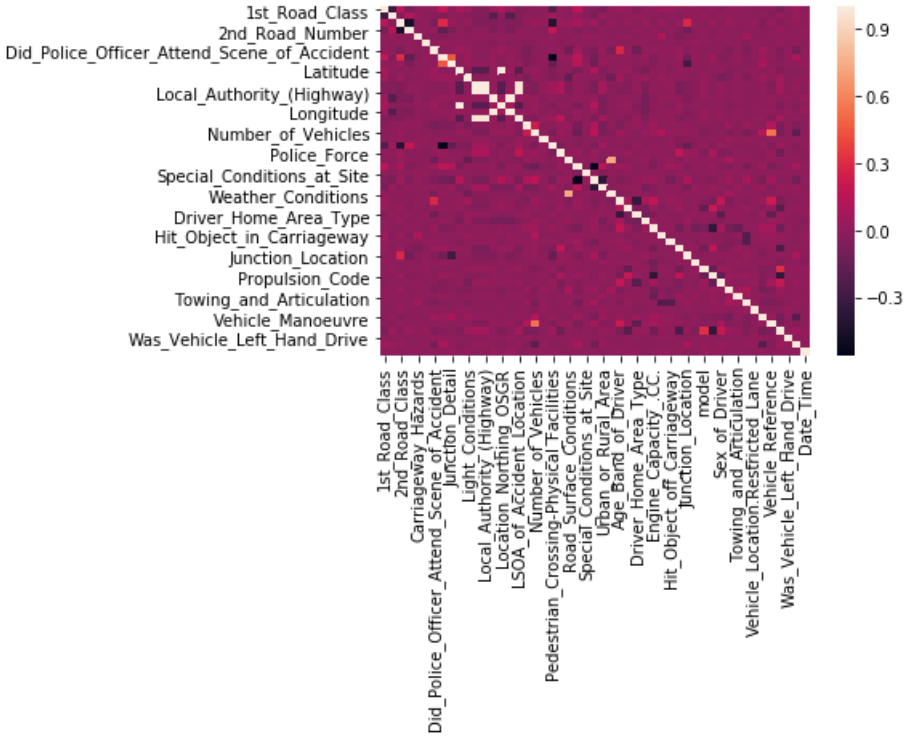


Figura 5.3: Matriz de correlação.

5.2.3 Balanceamento do Conjunto de Dados

Se se verificar a percentagem da quantidade dos diferentes resultados (*Slight, Serious* e *Fatal*) é possível perceber que estamos perante um conjunto de dados que **não é balanceado** (Figura 5.4), isto é, neste conjunto de dados existem ocorrências que são raras, mas muito importantes de serem detetadas [44], como acontece no caso de os acidentes serem *serious* ou *fatal*.

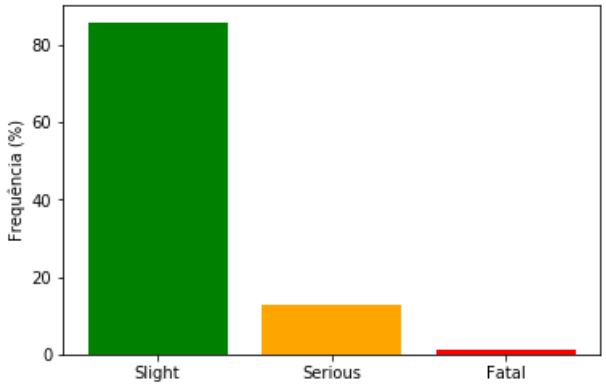


Figura 5.4: Percentagem da frquência de ocorrências das diferentes classes.

Torna-se mais importante conseguir classificar um acidente fatal ou severo como tal, do que ter mais predições certas para acidentes ligeiros. O importante é tentar minimizar ao

máximo as classificações obtidas como ligeiras, quando na verdade são ou fatais ou severas.

Este facto é muito importante para o treino eficaz do algoritmo e por isso há várias formas de se conseguir fazer esse balanceamento. Um conjunto de dados não balanceado caracteriza-se por ter mais amostras para uma dada classe do que para as outras. Uma das formas de poder balancear o conjunto de dados é eliminando amostras da classe maioritária.

A técnica ***Tomek's links*** faz precisamente isso. Se a distância entre duas amostras de classes diferentes for muito reduzida, é eliminada a amostra da classe maioritária. Considerando duas amostras de classes diferentes x e y, define-se como *Tomek's link*, para qualquer amostra z:

$$d(x, y) < d(x, z) \tag{5.3}$$

e

$$d(x, y) < d(y, z) \tag{5.4}$$

em que d é a distância entre as duas amostras. [10] A Figura 5.5 representa a explicação anterior.

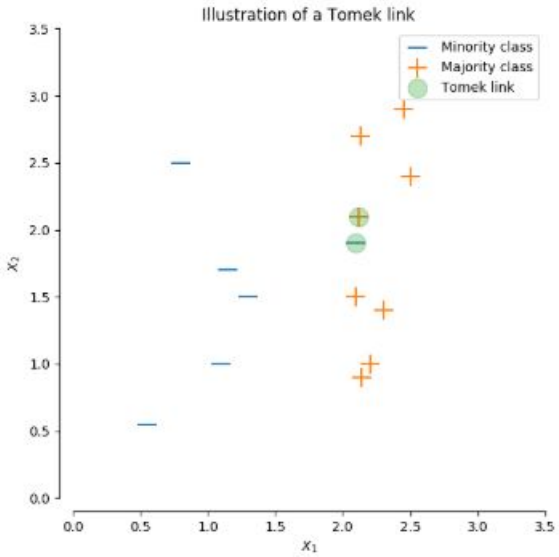


Figura 5.5: Tomek's links. [10]

Ao aplicar esta técnica ao conjunto de dados em questão, obtiveram-se os seguintes resultados:

Accuracy	0.91437
Precisão	0.30531
Recall	0.33270

Accuracy	0.91465
Precisão	0.63821
Recall	0.33511

Tabela 5.6: Resultado sem e com seleção de atributos, respetivamente para a técnica *Tomek's Links*.

	Sem seleção			Com seleção		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	0	0	12	0	0	12
`Serious`	0	0	183	0	1	186
`Slighty`	0	4	2125	0	4	2121

Tabela 5.7: Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *Tomek's Links*.

Apesar da *accuracy* aumentar, assim como a precisão no caso em que há seleção de atributos, o recall diminuiu. Ao observar a matriz de confusão, pode observar-se também que poucos ou nenhuns casos **fatais** ou **severos** são identificados, sem e com a seleção de atributos.

Outra técnica que pode ser aplicada para a diminuição da classe maioritária é a ***Edited Nearest Neighbours***. Neste caso, é feita a eliminação das amostras cuja classe difere da maioria da vizinhança, aplicando o algoritmo *nearest-neighbours* [10].

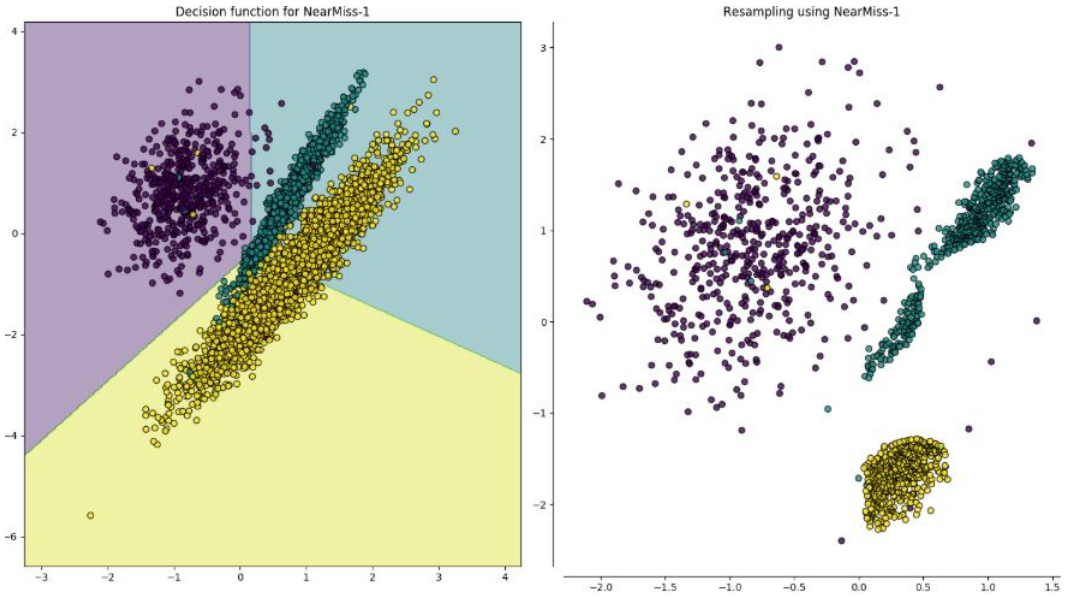


Figura 5.6: Edited Nearest Neighbours [10].

Quando se aplicou esta técnica os resultados obtidos foram os seguintes:

Accuracy	0.98871
Precisão	0.49435
Recall	0.5

Accuracy	0.98904
Precisão	0.49451
Recall	0.5

Tabela 5.10: Resultado sem e com seleção de atributos, respetivamente para a técnica *Edited Nearest Neighbours*.

Os resultados da *accuracy*, precisão e *recall* são visivelmente melhores em relação à técnica anterior. No entanto, a matriz de confusão passa a ser uma matriz 2x2, o que significa que uma das classes foi perdida e por isso não é uma técnica a considerar aplicar.

As técnicas acima descritas são consideradas ***under-sampling***, por reduzirem o número de amostras no conjunto de dados onde são aplicadas. No entanto, existem também técnicas ***over-sampling***, às quais foram feitos igualmente testes.

A primeira técnica *over-sampling* a ser testada foi a **SMOTE (Synthetic Minority Over-sampling TEchnique)**. Com base nas amostras já presentes no conjunto de dados são criadas novas amostras da classe minoritária, podendo desta forma equilibrar todas as classes, como mostra a Figura 5.8 [11].

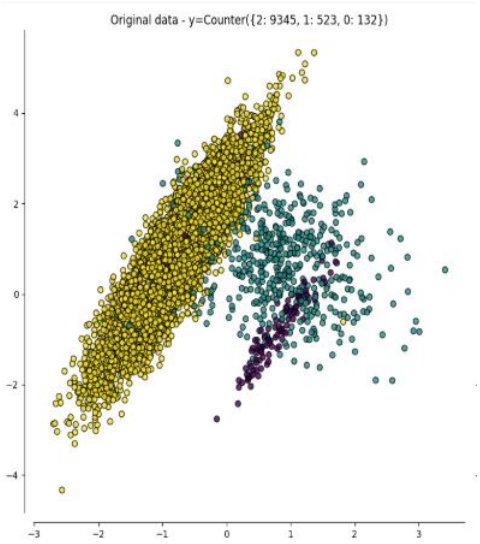


Figura 5.7: Dados orginais.[11]

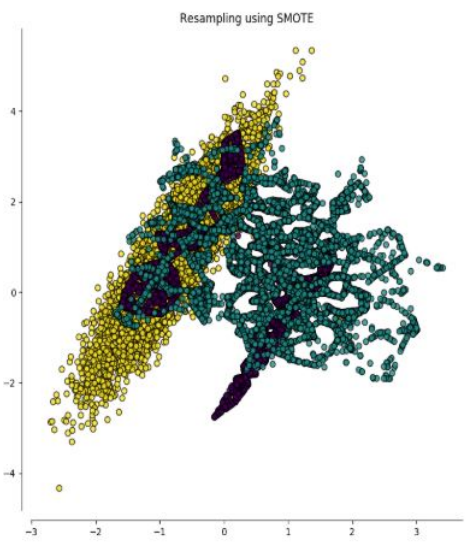


Figura 5.8: Dados aquando da aplicação do SMOTE. [11]

Quando se aplicou esta técnica os resultados obtidos foram os seguintes:

Accuracy	0.75119
Precisão	0.84810
Recall	0.74884

Accuracy	0.75480
Precisão	0.84440
Recall	0.75248

Tabela 5.13: Resultado sem e com seleção de atributos, respetivamente para a técnica *SMOTE*.

	Sem seleção			Com seleção		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2243	0	0	2243	0	0
`Serious`	1	2219	6	6	2211	13
`Slighty`	62	1591	550	76	1545	582

Tabela 5.14: Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *SMOTE*.

Em relação às técnicas *under-sampling*, esta produz resultados substâncialmente melhores nas métricas da precisão e recall. Também a matriz de confusão apresenta agora valores melhores no que diz respeito às predições *Fatal* e *Serious*.

Pode concluir-se desde já, que para casos como este, técnicas *over-sampling* produzem resultados melhores.

Aplicou-se ainda a técnica ***ADASYN (Adaptive Synthetic)***. Tal como a anterior esta é over-sampling, fazendo aumentar o número de amostras da classe minoritária, como apresenta a Figura 5.10.

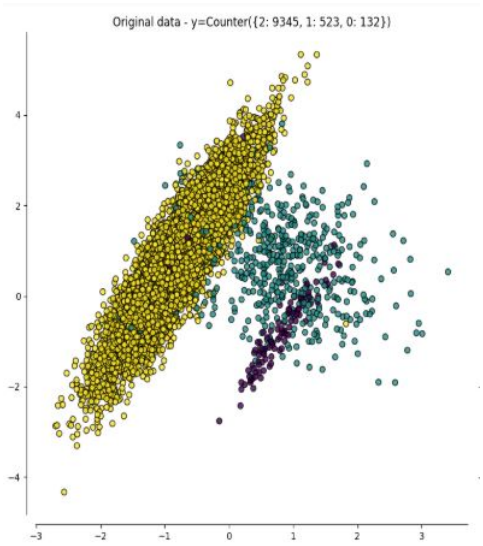


Figura 5.9: Dados orginais.[11]

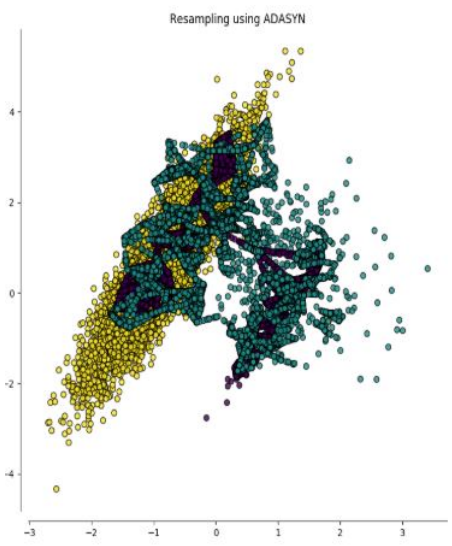


Figura 5.10: Dados aquando da aplicação do ADASYN. [11]

Quando se aplicou esta técnica os resultados obtidos foram os seguintes:

Accuracy	0.75416
Precisão	0.85196
Recall	0.75

Accuracy	0.75564
Precisão	0.85205
Recall	0.75308

Tabela 5.17: Resultado sem e com seleção de atributos, respetivamente para a técnica *ADASYN*.

	Sem seleção			Com seleção		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2279	0	0	2250	0	0
`Serious`	0	2200	0	1	2199	0
`Slighty`	63	1575	546	62	1559	570

Tabela 5.18: Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *ADASYN*.

De entre todas as técnicas testadas até ao momento esta é, sem dúvida, a melhor. Consegue-se através desta obter quase 100% das predições corretas para as classes *Fatal* e *Serious*, sendo estas muito importantes para o cenário em questão.

Ambas usam o mesmo algoritmo para a geração de novas amostras. Considerando uma amostra x_i, o calculo da nova (x_{new}) é feito através da seguinte equação:

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i) \tag{5.5}$$

sendo λ um número aletório entre 0 e 1. No entanto, no caso do *ADASYN* o número de amostras gerado para cada x_i é proporcional ao número de amostras que não são da mesma classe de x_i numa determinada vizinhança [11].

Dentro do *SMOTE* existem algumas variantes que podem também ser testadas, como é o caso do ***BorderlineSMOTE***, ***SVMSMOTE***, e o ***KMeansSMOTE***.

No primeiro as amostras serão classificadas como **ruído** (o vizinho mais próximo é de uma classe diferente da de x_i), **em perigo** (pelo menos metade dos vizinhos mais próximos são da mesma classe de x_i), ou **a salvo** (todos os vizinhos mais próximos são da mesma classe de x_i). O *BorderlineSMOTE* irá usar as amostras em perigo para gerar novas, como se pode ver na Figura 5.12.

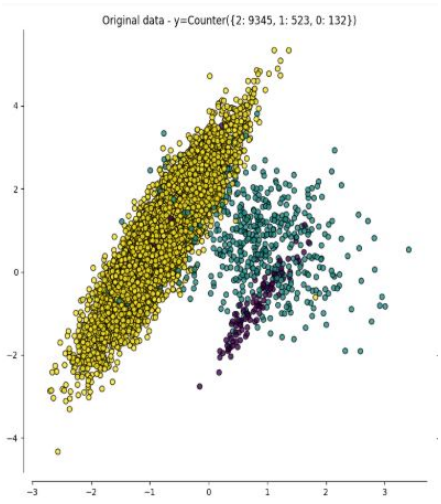


Figura 5.11: Dados orginais.[11]

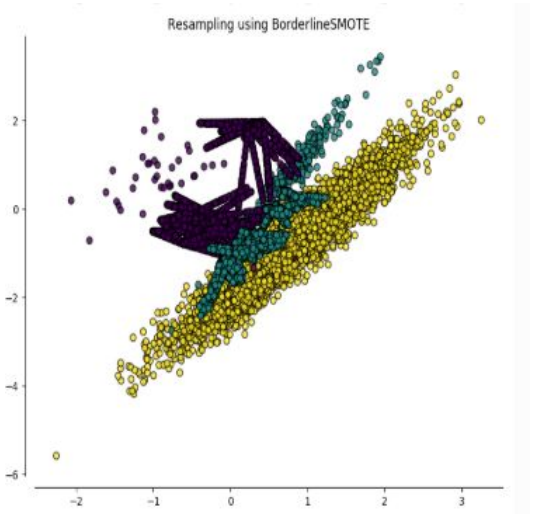


Figura 5.12: Dados aquando da aplicação do BorderlineSMOTE. [11]

Quando se aplicou esta técnica os resultados obtidos foram os seguintes:

Accuracy	0.75659
Precisão	0.83365
Recall	0.75437

Accuracy	0.75990
Precisão	0.83563
Recall	0.75769

Tabela 5.21: Resultado sem e com seleção de atributos, respetivamente para a técnica *BorderlineSMOTE*.

	Sem seleção			Com seleção		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2237	4	2	2240	2	1
`Serious`	2	2176	48	3	2177	46
`Slighty`	20	1548	653	30	1520	653

Tabela 5.22: Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *BorderlineSMOTE*.

Na segunda variante do *SMOTE* é utilizado o classificador SVM para gerar novas amostras [11].

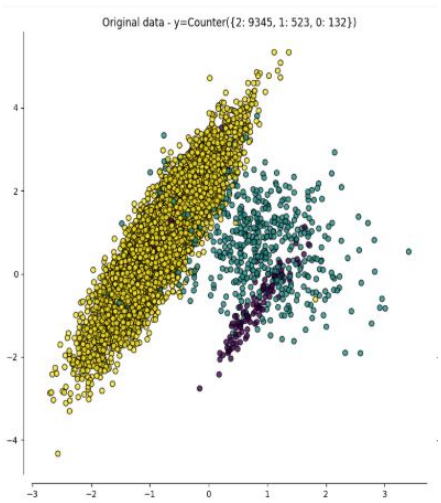


Figura 5.13: Dados orginais.[11]

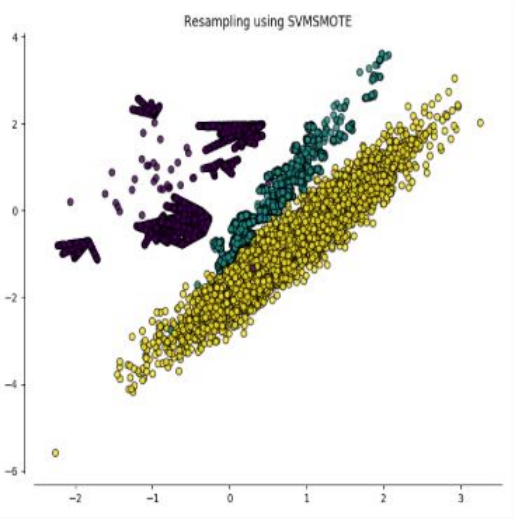


Figura 5.14: Dados aquando da aplicação do SVMSMOTE. [11]

Quando se aplicou esta técnica os resultados obtidos foram os seguintes:

Accuracy	0.69263
Precisão	0.82302
Recall	0.75297

Accuracy	0.70967
Precisão	0.82911
Recall	0.76247

Tabela 5.25: Resultado sem e com seleção de atributos, respetivamente para a técnica *SVMSMOTE*.

	Sem seleção			Com seleção		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	953	4	1	1097	1	1
`Serious`	3	2140	58	2	2150	46
`Slighty`	20	1570	659	37	1531	708

Tabela 5.26: Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *SVMSMOTE*.

Por último, na variante KMeans é utilizado o método *KMeans Clustering* antes de ser aplicado o SMOTE. Serão agrupados as amostras juntas e geradas novas dependendo da densidade do cluster [11].

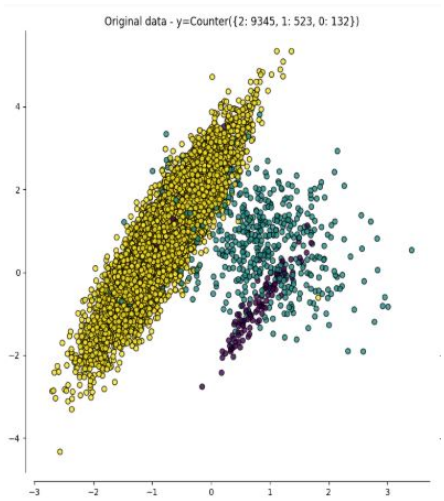


Figura 5.15: Dados orginais.[11]

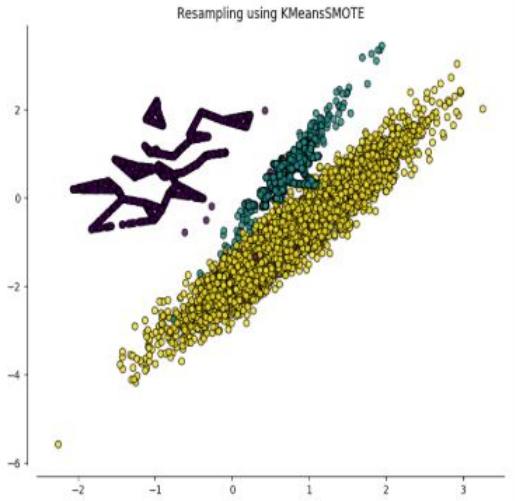


Figura 5.16: Dados aquando da aplicação do KMeansSMOTE. [11]

Quando se aplicou esta técnica os resultados obtidos foram os seguintes:

Accuracy	0.75060
Precisão	0.84529
Recall	0.74824

Accuracy	0.75044
Precisão	0.84643
Recall	0.74809

Tabela 5.29: Resultado sem e com seleção de atributos, respetivamente para a técnica *KMeansSMOTE*.

	Sem seleção			Com seleção		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2243	0	0	2243	0	0
`Serious`	2	2213	11	2	2215	9
`Slighty`	58	1593	552	57	1597	549

Tabela 5.30: Matriz de confusão sem e com seleção de atributos, respetivamente para a técnica *KMeansSMOTE*.

Ainda que o *SMOTE* tenha algumas variantes é no *ADASYN* onde se obtém os melhores resultados para o caso em questão, como já foi explicado anteriormente. Por este motivo, para o balanceamento do conjunto de dados optou-se por se utilizar a técnica ***ADASYN***.

5.2.4 Cross-validation

Uma das divisões típicas do conjunto de dados é de **treino** e **teste**. No entanto, desta divisão pode advir aquilo a que se chama **overfitting**. Overfitting ocorre quando os resultados do modelo quando aplicados aos dados de treino são muito bons, mas quando surgem dados que o modelo não conhece, os resultados são são os esperados.

Uma das formas de evitar este fenómeno é através da utilização de ***cross-validation*** (CV). *Cross-validation* é um método estatístico de avaliação e comparação de algoritmos de aprendizagem através da divisão dos dados em dois tipos: dados treino e validação. [45] Esta técnica consiste em manter os dados de teste de fora para a avaliação final e dividir os dados de treino em k conjuntos de dados mais pequenos (*k-fold CV*) [12]. O modelo passa a ser treinado utilizando k - 1 *folds* como dados de treino e o resultado do desempenho é avalidado com os restantes dados, os já referidos **dados de validação**. A avaliação final é executada através dos dados de teste.

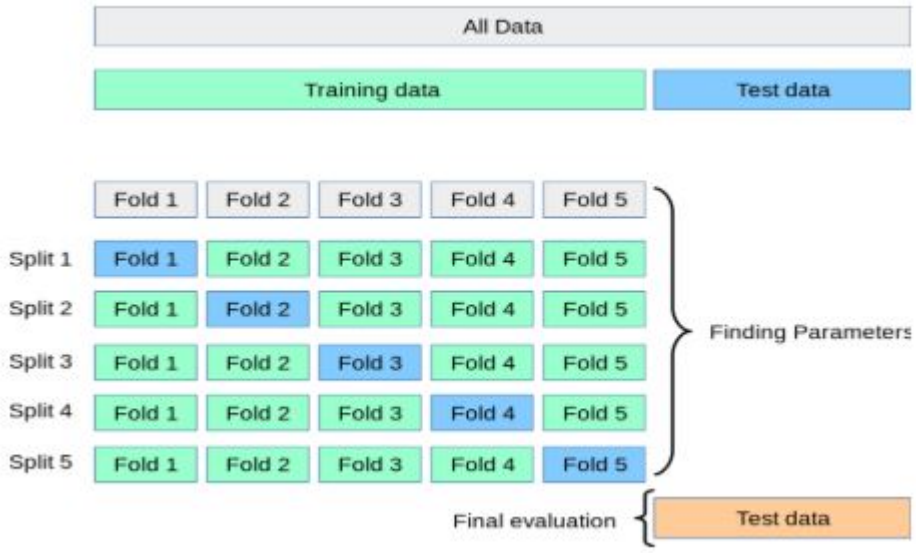


Figura 5.17: Representação da técnica *cross-validation*. [12]

O conjunto de dados foi inicialmente dividido 75% dos dados para treino e 25%. Com a aplicação do cross-validation optou-se por um k = 4, de forma a no final o conjunto de dados ser dividido em 55% para treino, 20% para validação e 25% para teste.

Ao aplicar esta técnica ao conjuntos de dados já devidamente pré-processado, obtiveram-se os seguintes resultados para os seguintes algoritmos:

Naïve Bayes

Accuracy	0.89069
Precisão	0.91295
Recall	0.88955

Accuracy	0.91026
Precisão	0.92693
Recall	0.90933

Tabela 5.33: Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Naïve Bayes*.

	Com *cross-validation*			Sem *cross-validation*		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2250	0	0	2250	0	0
`Serious`	3	2192	6	3	2198	0
`Slighty`	55	662	1474	36	557	1598

Tabela 5.34: Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Naïve Bayes*.

Árvores de Decisão

Accuracy	0.93933
Precisão	0.93919
Recall	0.93889

Accuracy	0.94113
Precisão	0.94072
Recall	0.94069

Tabela 5.37: Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Árvores de Decisão*.

	Com *cross-validation*			Sem *cross-validation*		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2240	1	9	2242	1	7
`Serious`	18	1969	223	16	1981	204
`Slighty`	17	135	2039	16	147	2028

Tabela 5.38: Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Árvores de Decisão*.

Random Forest

Accuracy	0.95799
Precisão	0.96108
Recall	0.95776

Accuracy	0.96161
Precisão	0.96431
Recall	0.96140

Tabela 5.41: Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Random Forest*.

	Com *cross-validation*			Sem *cross-validation*		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2242	0	8	2242	0	8
`Serious`	0	1950	251	0	1970	231
`Slighty`	0	20	2171	0	16	2175

Tabela 5.42: Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Random Forest*.

KNN

Accuracy	0.72553
Precisão	0.83636
Recall	0.72265

Accuracy	0.75504
Precisão	0.85200
Recall	0.75247

Tabela 5.45: Resultado com e sem *cross-validation*, respetivamente para o algoritmo *KNN*.

	Com *cross-validation*			Sem *cross-validation*		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2250	0	0	2250	0	0
`Serious`	1	2196	4	0	2201	0
`Slighty`	94	1724	373	67	1560	564

Tabela 5.46: Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *KNN*.

SVM

Accuracy	0.95453
Precisão	0.95424
Recall	0.95417

Accuracy	0.96296
Precisão	0.96276
Recall	0.96264

Tabela 5.49: Resultado com e sem *cross-validation*, respetivamente para o algoritmo *SVM*.

	Com *cross-validation*			Sem *cross-validation*		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2250	0	0	2250	0	0
`Serious`	0	2028	173	0	2102	99
`Slighty`	8	121	2062	6	141	2044

Tabela 5.50: Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *SVM*.

Neural Networks

Accuracy	0.90921
Precisão	0.92077
Recall	0.90827

Accuracy	0.95423
Precisão	0.95716
Recall	0.95375

Tabela 5.53: Resultado com e sem *cross-validation*, respetivamente para o algoritmo *Neural Networks*.

	Com *cross-validation*			Sem *cross-validation*		
	Fatal	Serious	Slighty	Fatal	Serious	Slighty
`Fatal`	2250	0	0	2250	0	0
`Serious`	0	2191	10	0	2192	5
`Slighty`	181	432	1598	111	188	1892

Tabela 5.54: Matriz de confusão com e sem *cross-validation*, respetivamente para o algoritmo *Neural Networks*.

Como se pode observar através das tabelas anteriores, quando não se utilizou *cross-validation* os resultados obtidos foram melhores, o que só comprova a existência de *overfitting* antes desta aplicação. Desta forma, esta técnica foi aplicada para se obterem resultados "reais", isto é, a evitar que só se obtenham bons resultados quando se utilizam as amostras do conjunto de dados inicial e quando chegam novas amostras, os resultados não sejam bons.

PCA (Principal Component Analysis)

Na tentativa de melhorar o desempenho dos modelos de *machine learning* aplicou-se a técnica do PCA, que tem como objetivo a redução do número de atributos a ser utilizado no treino do modelo. O excerto de código seguinte mostra a função responsável pela aplicação do PCA.

```
1 def apply_pca(self, descriptive):
2         from sklearn.decomposition import PCA
3         pca = PCA(n_components = 0.95,random_state = 0)
4 
5         descriptive = pca.fit_transform(descriptive)
6         plt.plot(np.cumsum(pca.explained_variance_ratio_))
```

```
7
8         return descriptive
```

Após a aplicação desta técnica o número de atributos diminuiu de **6819** para **5093**. Os resultados para os diferentes algortimos são os apresentados de seguida e antes da aplicação da técnica de *cross-validation*.

Naïve Bayes

Quando se aplicou esta técnica neste algoritmo os resultados obtidos foram os seguintes:

Accuracy	0.57514
Precisão	0.58029
Recall	0.57257

Tabela 5.56: Resultado com PCA para o algoritmo *Naïve Bayes*.

	Com *PCA*		
	Fatal	Serious	Slighty
`Fatal`	1967	5	306
`Serious`	1198	523	498
`Slighty`	452	396	1375

Tabela 5.57: Matriz de confusão com PCA para o algoritmo *Naïve Bayes*.

Árvores de Decisão

Quando se aplicou esta técnica neste algoritmo os resultados obtidos foram os seguintes:

Accuracy	0.84330
Precisão	0.84272
Recall	0.84217

Tabela 5.59: Resultado com PCA para o algoritmo *Árvores de Decisão*.

	Com *PCA*		
	Fatal	Serious	Slighty
`Fatal`	2232	18	28
`Serious`	44	1846	329
`Slighty`	71	563	1589

Tabela 5.60: Matriz de confusão com PCA para o algoritmo *Árvores de Decisão*.

Random Forest

Quando se aplicou esta técnica neste algoritmo os resultados obtidos foram os seguintes:

Accuracy	0.90848
Precisão	0.91010
Recall	0.90773

Tabela 5.62: Resultado com PCA para o algoritmo *Random Forest*.

	Com *PCA*		
	Fatal	Serious	Slighty
`Fatal`	2278	0	0
`Serious`	0	2028	191
`Slighty`	8	416	1799

Tabela 5.63: Matriz de confusão com PCA para o algoritmo *Random Forest*.

KNN

Quando se aplicou esta técnica neste algoritmo os resultados obtidos foram os seguintes:

Accuracy	0.74270
Precisão	0.83798
Recall	0.74073

Tabela 5.65: Resultado com PCA para o algoritmo *KNN*.

	Com *PCA*		
	Fatal	Serious	Slighty
`Fatal`	2270	0	0
`Serious`	0	2199	12
`Slighty`	82	1627	514

Tabela 5.66: Matriz de confusão com PCA para o algoritmo *KNN*.

SVM

Quando se aplicou esta técnica neste algoritmo os resultados obtidos foram os seguintes:

Accuracy	0.95401
Precisão	0.95363
Recall	0.95363

Tabela 5.68: Resultado com PCA para o algoritmo *SVM*.

	Com *PCA*		
	Fatal	Serious	Slighty
`Fatal`	2278	0	0
`Serious`	0	2084	135
`Slighty`	8	166	2049

Tabela 5.69: Matriz de confusão com PCA para o algoritmo *SVM*.

Neural Networks

Quando se aplicou esta técnica neste algoritmo os resultados obtidos foram os seguintes:

Accuracy	0.95714
Precisão	0.95887
Recall	0.95680

Tabela 5.71: Resultado com PCA para o algoritmo *Neural Networks*.

	Com *PCA*		
	Fatal	Serious	Slighty
`Fatal`	2278	0	0
`Serious`	0	2179	40
`Slighty`	22	226	1975

Tabela 5.72: Matriz de confusão com PCA para o algoritmo *Neural Networks*.

Mesmo com a aplicação do PCA o número de atributos continua bastante elevado, e em alguns dos algoritmos os resultados pioraram. Ao analisar novamente o conjunto de dados, verificou-se que o atributo *model* poderia contribuir para este aumento isto porque este tem dezenas de valores diferentes, o que faz com que, quando se faz a conversão dos dados categóricos para numéricos este aumento seja significativo.

Quando não foi considerado este atributo, conseguiu-se reduzir para mais de metade dos atributos, passando de **6819** para **3325** e após a aplicação do PCA para **2981**. Os resultados, considerando apenas os três melhores algoritmos até ao momento (**Random Forest**, **SVM** e ***Neural Networks***), foram os seguintes:

Accuracy	0.92439
Precisão	0.92545
Recall	0.92433

Accuracy	0.89949
Precisão	0.89927
Recall	0.89969

Accuracy	0.96190
Precisão	0.96203
Recall	0.96202

Tabela 5.76: Resultados dos algortimos *Random Forest*, SVM e *Neural Networks*, respetivamente.

Em termos de eficiência, após a eliminação do atributo *model*, a diferença é bastante notória, sendo que o treino dos modelos é feito de forma muito mais rápida. Para além disso, os resultados até conseguiram ser melhores no caso das *Neural Networks* e do *Random Forest*. Conseguiu-se assim baixar a complexidade do modelo e manter a qualidade do seu desempenho.

5.3 Avaliação dos modelos

Para uma escolha do modelo a ser utilizado automática, foi criado em pequeno *script*, que, com base nas métricas acima referidas *(Accuracy, Precisão e Recall)*, irá avaliar a eficiência do modelo e efetuar a escolha do que apresentar melhores resultados. Esta avalição terá em com os seguintes pontos:

- Deve ser mantida uma **accuracy** igual ou superior a **90%**.
- Deve ser mantida uma **precisão** igual ou superior a **85%**.
- Deve ser mantida uma ***recall*** igual ou superior a **90%**.

Incialmente escolhe-se o modelo tendo em conta o que tem melhor valor na métrica *recall*. De seguida, esse modelo é avaliado. Verifica-se se esse modelo tem as métricas de acordo com

os pontos referidos anteriormente. Estes valores foram escolhidos tendo em contas as matrizes de confusão que foram sendo apresentadas.

Tendo em conta os resultados apresentados na secção anterior, pode concluir-se que o algoritmo a ser utilizado para o cenário em questão é o **Neural Networks**. De uma forma geral os algoritmos testados obtiveram bons resultados, sendo que estes se devem sobretudo ao pré-processamento, que foi a parte mais importante para o desenvolvimento deste modelo. Embora outros modelos estivessem dentro dos padrões referidos anteriormente o que realmente se mostra melhor é o escolhido para ser utilizado.

5.4 Integração do Modelo

Concluída a fase da criação do modelo de *Machine Learning* é necessário integrar o mesmo na plataforma *IoT*. Para isso foi criado um mockup dos dados que seriam apresentados na mesma. A Figura 5.18 serve como rascunho de como será a disposição dos mesmos na plataforma.

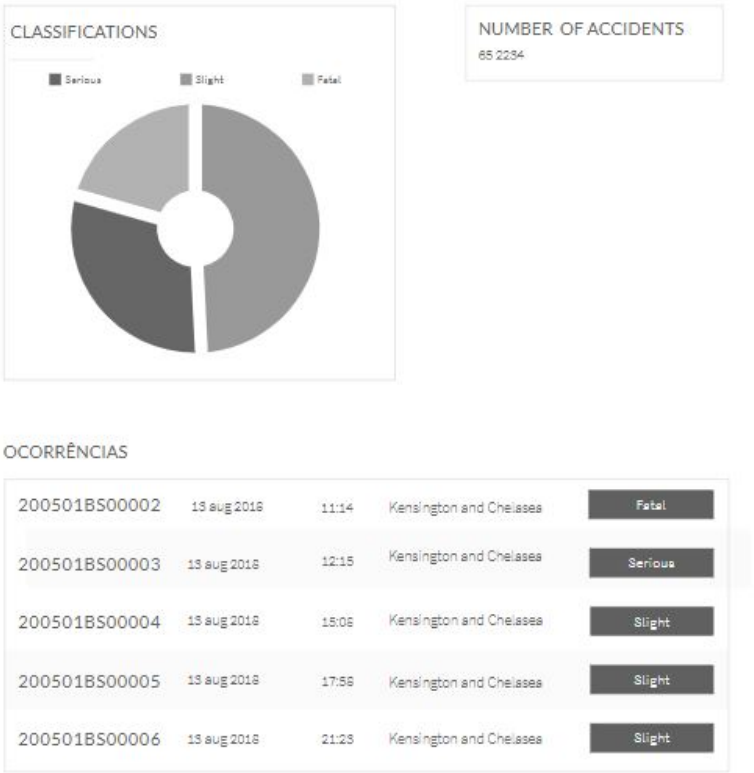


Figura 5.18: Mockup da integração do modelo na plataforma.

O dados que vão chegando à plataforma são simulados através do conjunto de dados que se obteve.

Para a integração do modelo na plataforma foi necessário criar duas tabelas na base de dados, onde são guardados as novas amostras que vão chegando sempre que existe uma nova ocorrência. O excerto de código seguinte mostra como são criadas essas tabelas e quais as

variavéis que são efetivamente guardadas.

```
class Evaluation(db_accounts.Model):
    __tablename__ = 'evaluation_model'
    id = db_accounts.Column(db_accounts.Integer, primary_key=True)
    recall = db_accounts.Column(db_accounts.Float, nullable=False)
    precision = db_accounts.Column(db_accounts.Float, nullable=False)
    accuracy = db_accounts.Column(db_accounts.Float, nullable=False)

    def __init__(self, recall, precision, accuracy):
        self.recall = recall
        self.precision = precision
        self.accuracy = accuracy


class Accidents(db_accounts.Model):
    __tablename__ = 'accidents'
    id_accident = db_accounts.Column(db_accounts.String(100), primary_key=True)
    date = db_accounts.Column(db_accounts.String(100), nullable=False)
    dtime = db_accounts.Column(db_accounts.String(100), nullable=False)
    local = db_accounts.Column(db_accounts.String(100), nullable=False)
    classification = db_accounts.Column(db_accounts.String(100), nullable=False
    )
    real_classification = db_accounts.Column(db_accounts.String(100), nullable=
    False)

    def __init__(self, id_accident, date, dtime, local, classification,
    real_classification):
        self.id_accident = id_accident
        self.date = date
        self.dtime = dtime
        self.local = local
        self.real_classification = real_classification
        self.classification = classification
```

Os dados guardados na base de dados, são apresentados numa pagina web na qual será possível a visualização das ocorrências e as correspondentes classificações obtidas através no modelo de *Machine Learning* através do *endpoint* ***'/accidents'***.

Este invoca uma função, de nome *sensing*, cujo o objetivo é adicionar as novas amostras à base de dados.

Por fim são criados componentes, também pela *framework Angular*, que vão atualizando a página de 10 em 10 segundos, de forma a simular a chegada de novos dados, uma vez que os dados obtidos não estão a ser recolhidos em tempo real.

O resultado final dados na plataforma é o representado na Figura 5.19.

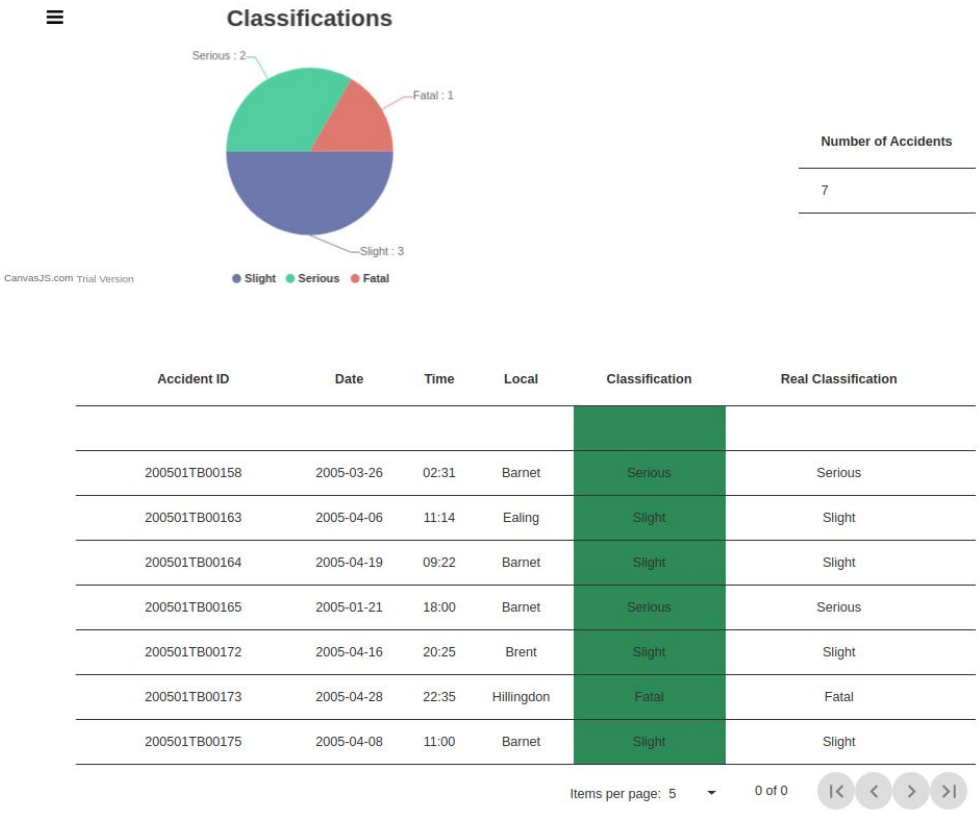


Figura 5.19: Modelo integrado numa plataforma para monitorização.

Capítulo 6

Conclusões

O presente trabalho consistiu na criação de uma arquitetura para dar respostas às necessidades das cidades do futuro, tendo por base a aplicação de técnicas de *Machine Learning* ao conjunto de dados escolhido e a integração desse modelo numa plataforma para monitorização das ocorrências. O trabalho prático foi realizado tendo em conta estes objetivos, podendo afimar-se que foram cumpridos com sucesso.

A plataforma disponibiliza várias informações sobre as ocorrências, nomeadamente, e a mais relevante, a previsão da classificação de um dado acidente. Esta previsão é feita por um modelo que foi criado através de *Machine Learning*, sendo que o seu desenvolvimento obtendo os melhores resultados possíveis foi o foco de todo o trabalho apresentado e também onde foi despendido mais tempo. A arquitetura desenhada é genérica e portanto serve não só para o cenário em questão, mas também para qualquer outro que se pretenda desenvolver.

Para o cenário em questão, tanto a arquitetura como o modelo de *Machine Learning* desenvolvido resultaram bastante bem e por isso os resultados obtidos são tão bons.

6.1 Desafios

Ao longo do desenvolvimento do presente projecto foram encontrados alguns desafios que por vezes dificultaram o avanço do trabalho.

Cenário

A escolha de um cenário que pudesse fazer sentido, aplicável às *Smart Cities* e que ao mesmo tempo fosse possível arranjar dados para o mesmo, foi talvez o que mais tempo dispendeu numa fase inicial do projecto. No entanto, o trabalho foi sendo desenvolvido tendo em conta um cenário geral e com o passar do tempo este foi ganhando forma, após ter sido escolhido o conjunto de dados. Posto isto, foi então adaptado o cenário ao conjunto de dados que se arranjou e que se podia adequar ao objetivo final deste projeto.

Treino dos modelos

Relativamente ao treino dos modelos, apesar do conjunto de dados escolhido ter muitas amostras, por questões de *hardware* optou-se apenas usar 10000 desses dados. Considerou-se ser um número já significativo e com o qual se pudesse trabalhar. As restantes amostras foram utilizadas para simulação da receção de novas à plataforma.

6.2 Trabalho futuro

Como trabalho futuro, propõe-se a aplicação da mesma arquitetura a outros cenários que possam fazer sentido no âmbito das *Smart Cities* e que melhorem a qualidade de vida dos cidadãos. Outra situação seria que a criação de novos cenários fosse com base em dados recolhidos em tempo real, com sensores que já existam espalhados pela cidade, sem que haja a necessidade de simulação de novas amostras como foi feito neste projeto. Por fim, propõe-se o melhoramento da plataforma em termos de usabilidade e design, onde são apresentados os dados para a monitorização das ocorrências bem como a sua integração em plataformas IoT em desenvolvimento e uso na Altice Labs.

Bibliografia

[1] S. Zygiaris, "Smart City Reference Model: Assisting Planners to Conceptualize the Building of Smart City Innovation Ecosystems," *Journal of the Knowledge Economy 4: 2 (2013) 217–231*, 2012.

[2] "Tr-0057 functional architecture," acedido a 2019-06-17. [Online]. Available: http://www.onem2m.org/getting-started/onem2m-overview/introduction/functional-architecture

[3] M. Mancini, "Internet das Coisas: História, Conceitos, Aplicações e Desafios," 2018.

[4] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, Mar 1986. [Online]. Available: https://doi.org/10.1007/BF00116251

[5] "Random forest simple explanation," acedido a 2019-06-17. [Online]. Available: https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

[6] "K nearest neighbours - introduction to machine learning algorithms," acedido a 2019-06-17. [Online]. Available: https://towardsdatascience.com/k-nearest-neighbours-introduction-to-machine-learning-algorithms-18e7ce3d802a

[7] "Support vector machine - introduction to machine learning algorithms," acedido a 2019-06-17. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[8] "The differences between artificial and biological neural networks," acedido a 2019-06-18. [Online]. Available: https://towardsdatascience.com/the-differences-between-artificial-and-biological-neural-networks-a8b46db828b7

[9] "Learning process of a neural network," acedido a 2019-06-18. [Online]. Available: https://towardsdatascience.com/how-do-artificial-neural-networks-learn-773e46399fc7

[10] G. Lemaître, F. Nogueira, and C. K. Aridas, "Under-sampling," acedido a 2019-07-11. [Online]. Available: https://imbalanced-learn.readthedocs.io/en/stable/under_sampling.html

[11] ——, "Over-sampling," acedido a 2019-07-15. [Online]. Available: https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html

[12] Scikit-learn, "Cross-validation: evaluating estimator performance," acedido a 2019-07-17. [Online]. Available: https://scikit-learn.org/stable/modules/cross_validation.html

[13] D. of Economic and S. A. of United Nations. Population Division (2017)., *World Population Prospects: The 2017 Revision, Key Findings and Advance Tables*, 2017, p. 2. [Online]. Available: https://population.un.org/wpp/Publications/Files/WPP2017_KeyFindings.pdf

[14] I. Pramanik, R. Y. Lau, H. Demirkan, and A. K. Azad, "Smart health: Big data enabled health paradigm within smart cities," *Expert Systems with Applications*, vol. 87, pp. 370–383, 2017. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1086093472

[15] "Engineering design process," acedido a 2019-06-14. [Online]. Available: https://theworks.org/educators-and-groups/elementary-engineering-resources/engineering-design-process/

[16] M.-L. Marsal-Llacuna, J. Colomer-Llinàs, and J. Meléndez-Frigola, "Lessons in urban monitoring taken from sustainable and livable cities to better address the Smart Cities initiative ," p. 621, 2014.

[17] D. P. da Língua Portuguesa, "Cidade," 2008-2013, acedido a 2019-02-05. [Online]. Available: https://dicionario.priberam.org/cidade

[18] V. Dictionary, "City," acedido a 2019-02-05. [Online]. Available: https://www.vocabulary.com/dictionary/city

[19] D. Antrobus, "Smart green cities: from modernization to resilience?" *Urban Research & Practice, 4:2, 207-214, DOI: 10.1080/17535069.2011.579777*, 2011.

[20] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. A. Pardo, and H. J. Scholl, "Understanding Smart Cities: An Integrative Framework," *45th Hawaii International Conference on System Sciences*, 2012.

[21] I. Lee and K. Lee, "The Internet of Things (IoT): Applications, investments, and challenges for enterprises," pp. 431–440, 2015.

[22] J. Swetina, G. Lu, P. Jacobs, F. Ennesser, and J. Song, "Toward a standardized common m2m service layer platform: Introduction to onem2m," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 20–26, June 2014.

[23] M. R. Santiago and J. V. Payao, "INTERNET OF THINGS AND SMART CITIES: TECHNOLOGY, INNOVATION AND THE PARADIGM OF SUSTAINABLE DEVELOPMENT," pp. 787–805, 2018.

[24] D. Evans, "A Internet das Coisas Como a próxima evolução da Internet está mudando tudo," pp. 1–13, 2011.

[25] E. System, "What is Machine Learning? A definition," acedido a 2019-02-01. [Online]. Available: https://www.expertsystem.com/machine-learning-definition/

[26] Scikit-learn, "Imputation of missing values," acedido a 2019-02-01. [Online]. Available: https://scikit-learn.org/stable/modules/impute.html

[27] ——, "Encoding categorical features," acedido a 2019-02-01. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html#encoding-categorical-features

[28] V. A. Padilha and A. C. P. de Leon Ferreira de Carvalho, "MINERAÇÃO DE DADOS EM PYTHON: Pré-processamento," pp. 18–32, 2017, acedido a 2019-02-01. [Online]. Available: https://edisciplinas.usp.br/pluginfile.php/4052836/mod_resource/content/4/mineracaodadosbiologicos-parte3.pdf

[29] Scikit-learn, "Standardization, or mean removal and variance scaling," acedido a 2019-02-01. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html#standardization-or-mean-removal-and-variance-scaling

[30] ——, "Normalization," acedido a 2019-02-01. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html#normalization

[31] ——, "An introduction to machine learning with scikit-learn," acedido a 2019-03-25. [Online]. Available: https://scikit-learn.org/stable/tutorial/basic/tutorial.html

[32] E. Alpaydin, *Introduction to machine learning*, acedido a 2019-03-25. [Online]. Available: https://kkpatel7.files.wordpress.com/2015/04/alppaydin_machinelearning_2010.pdf

[33] Towards Data Science, "Introduction to machine learning for beginners," acedido a 2019-03-25. [Online]. Available: https://towardsdatascience.com/introduction-to-machine-learning-for-beginners-eed6024fdb08

[34] J. Kogan, C. Nicholas, and M. Teboulle, *Grouping Multidimensional Data: Recent Advances in Clustering*.

[35] "Complete guide to association rules (1/2)," acedido a 2019-06-17. [Online]. Available: https://towardsdatascience.com/association-rules-2-aa9a77241654

[36] Towards Data Science, "Naive bayes classifier," acedido a 2019-03-25. [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

[37] ——, "Decision trees in machine learning," acedido a 2019-03-28. [Online]. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

[38] "Árvores de decisão," acedido a 2019-04-03. [Online]. Available: http://web.tecnico.ulisboa.pt/ana.freitas/bioinformatics.ath.cx/bioinformatics.ath.cx/indexf23d.html?id

[39] "An implementation and explanation of the random forest in python," acedido a 2019-06-17. [Online]. Available: https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76

[40] "Support vector machines(svm) - an overview." [Online]. Available: https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

[41] J. Pagel and P. Kirshtein, "Neural Networks," 2017.

[42] N. Komninos, *Intelligent Cities: Innovation, Knowledge Systems, and Digital Spaces*. Spon Press, 2002. [Online]. Available: https://books.google.pt/books?id=psQq2PJp07gC

[43] Scikit-learn, "Recall_score," acedido a 2019-07-09. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html

[44] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, pp. 25–36, 11 2005.

[45] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation.* Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565