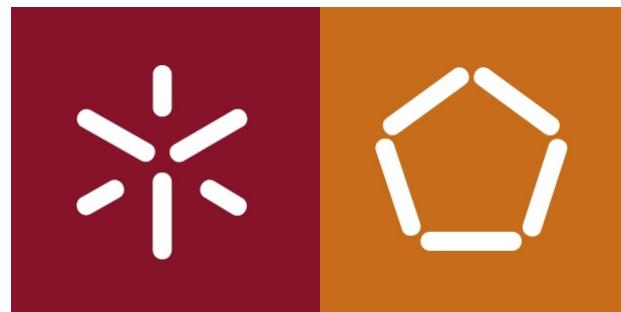


MESTRADO INTEGRADO EM ENGENHARIA
INFORMÁTICA

LABORATÓRIO EM ENGENHARIA INFORMÁTICA
PROJETO 45



A Model for the Quantification of
Running and Cycling Safety

João Miguel Freitas Palmeira, A73864
José Lopes Ramos, A73955
Rafael Machado Silva, A74264

7 de Julho de 2020



Agradecimentos

Gostaríamos de agradecer aos orientadores do projeto, o Professor Cesar Analide e o Professor Bruno Fernandes, pela disponibilidade demonstrada e por todo o tempo despendido no acompanhamento do nosso projeto, nomeadamente ao longo das várias reuniões realizadas onde nos prestaram auxílio na elaboração do mesmo, esclarecendo as dúvidas que o grupo ia encontrando ao longo do caminho e ajudando a definir um rumo adequado para este projeto.



Resumo

Neste projeto procura-se fornecer a todos os utilizadores vulneráveis da estrada (VRU) uma nova ferramenta para ajudá-los a deixar o conforto de sua casa ou qualquer outro qualquer local de conforto e sair para rua para praticar desporto, nomeadamente corrida ou ciclismo, sem incorrer uma série de riscos à saúde, uma vez que esses riscos serão analisados na aplicação móvel desenvolvida. Utilizando técnicas de Inteligência Artificial e *Machine Learning*, como *Random Forest* e *Convolutional Neural Networks*, treinadas com informações recebidas de várias fontes, desde clima, poluição e fontes de tráfego. O modelo de previsão final implementado no algoritmo da aplicação foi desenvolvida seguindo a metodologia CRISP-DM e, em seguida, inserido num servidor *web* da AWS que se conecta ao *front-end* da nossa aplicação. Cada utilizador poderá solicitar a avaliação de segurança que o nossa aplicação calcula para aquela determinada hora daquele determinado dia, após conectar-se ao servidor através da aplicação e solicite a avaliação.



Conteúdo

Agradecimentos	i
Resumo	ii
1 Introdução	1
1.1 Contextualização	1
1.2 Motivação	1
1.3 Objetivos	1
1.4 Estrutura do relatório	1
2 Estado da arte	3
2.1 <i>Data Mining</i>	3
2.2 <i>CRISP-DM</i>	4
2.2.1 Business understanding	4
2.2.2 Data understanding	4
2.2.3 Data preparation	5
2.2.4 Modeling	5
2.2.5 Evaluation	6
2.2.6 Deployment	6
2.3 <i>Machine Learning</i>	7
2.4 <i>Deep Learning</i>	8
2.5 Redes Neuronais	9
2.5.1 Redes Neuronais Recursivas	10
2.5.2 Vantagens e Desvantagens	10
2.5.3 Áreas de Aplicação	11
2.5.4 Ferramentas de Desenvolvimento	11
2.5.4.1 <i>TensorFlow</i>	11
2.5.4.2 <i>Keras</i> :	11
2.5.4.3 <i>Caffe</i> :	12
2.5.4.4 <i>PyTorch</i> :	12
2.5.4.5 <i>Theano</i> :	12
3 O problema e os seus desafios	13
3.1 Problema	13
3.1.1 Dados de entrada	13
3.1.2 Dados tratados	13
3.2 Desafios	13
3.3 Plano de Desenvolvimento	13
4 Exploração e Visualização dos dados	15
4.1 <i>Dataset Pollution</i>	15
4.1.1 Radiação Ultravioleta	17
4.1.2 Monóxido de Carbono	18
4.1.3 Dióxido de nitrogénio	19
4.1.4 <i>Particulate Matter 10</i>	20
4.1.5 Dióxido de enxofre	20
4.1.6 Ozono	21
4.2 <i>Dataset Traffic Flow</i>	21
4.3 <i>Dataset Traffic Incidents</i>	23
4.4 <i>Weather</i>	25
5 Tratamento dos dados	28
5.1 <i>Dataset Pollution</i>	28
5.2 <i>Dataset Traffic Flow</i>	29
5.3 <i>Dataset Traffic Incidents</i>	30
5.4 <i>Dataset Weather</i>	33
5.5 Agregação dos dados	34
5.6 Tratamento dados em falta	35
5.7 Criação do critério final	35
5.7.1 Critério da Poluição	36
5.7.2 Critério da Incidentes	36
5.7.3 Critério da <i>Speed_diff</i>	37
5.7.4 Critério do Tempo	37
5.7.5 Critério Final	38



5.8	<i>Dataset</i> final	38
6	Desenvolvimento e treino	40
6.1	<i>Random Forest</i>	40
6.1.1	<i>Tunning</i> inicial	40
6.1.2	<i>Cross-Validation</i>	41
6.1.3	<i>Tunning</i> final	42
6.2	<i>Artificial neural network</i> em <i>Python</i>	43
6.2.1	1 ^a Abordagem	43
6.2.2	2 ^a Abordagem	47
7	Análise dos resultados	49
7.1	<i>Random Forest</i>	49
7.2	<i>Artificial neural network</i> em <i>Python</i>	49
7.2.1	1 ^a Abordagem	49
7.2.2	2 ^a Abordagem	50
7.3	Comparação final	52
8	Criação da plataforma	53
8.1	Desenvolvimento do servidor	53
8.2	GoSafe	54
9	Conclusão	58
9.1	Conclusões e trabalho futuro	58



Lista de Tabelas

1	Tabela radiação UV	18
2	Tabela monóxido de carbono	19
3	Tabela dióxido de nitrogénio	20
4	Tabela PM10	20
5	Tabela dióxido de enxofre	21
6	Tabela com pesos da Poluição	36
7	Tabela com atribuições da Poluição	36
8	Tabela com pesos do Incidentes	36
9	Tabela com atribuições dos Incidentes	37
10	Tabela com pesos do Tempo	37
11	Tabela com atribuições do Tempo	38
12	Tabela com atribuições do Critério Final	38
13	Tabela com resultados 3 modelos	52



Lista de Figuras

1	Processo de <i>Data Mining</i>	3
2	Metodologia <i>CRISP-DM</i>	4
3	<i>Machine Learning</i> e os seus algoritmos	7
4	Enquadramento do <i>Deep Learning</i> na Inteligência Artificial [1]	8
5	Utilização de <i>Deep Learning</i> na <i>Google</i>	8
6	A inspiração biológica das redes neuronais	9
7	Exemplo de arquitetura de uma RNA	9
8	Exemplo de arquitetura de uma RNN	10
9	Ferramentas de Desenvolvimento	11
10	<i>Workflow</i> exploração de dados	15
11	Dados do <i>dataset Pollution</i>	16
12	Dados numéricos contidos no <i>dataset</i>	16
13	Dados relativos ao tipo de poluição	17
14	Média dos valores das partículas	17
15	Radiação UV ao logo do ano	18
16	Ozono ao logo do ano	21
17	<i>speed_diff</i> ao longo da semana	22
18	<i>speed_diff</i> ao longo da semana	22
19	Descrição dos incidentes	23
20	Categoria dos incidentes	24
21	Magnitude dos incidentes	24
22	Dados numéricos <i>dataset Weather</i>	25
23	Dados nominais <i>dataset Weather</i>	26
24	<i>Workflow</i> tratamento de dados	28
25	<i>Column Filter Pollution</i>	29
26	Tratamento do <i>dataset Pollution</i>	29
27	<i>Column Filter Traffic Flow</i>	30
28	Tratamento do <i>dataset Traffic Flow</i>	30
29	Análise das colunas <i>description</i> e <i>magnitude_of_delay_desc</i>	31
30	Valores de <i>description</i>	31
31	Valores de <i>magnitude_of_delay_desc</i>	32
32	<i>Column Filter Traffic Incidents</i>	32
33	Tratamento do <i>dataset Traffic Incidents</i>	32
34	26 valores do <i>weather_description</i>	33
35	<i>Column Filter Weather</i>	34
36	Tratamento do <i>dataset Weather</i>	34
37	Agregação dos dados	34
38	Tratamento dados em falta	35
39	Criação do critério final	35
40	<i>Workflow Random Forest</i>	40
41	Parte inicial do <i>tunning</i>	40
42	Definições <i>Table Creator</i>	41
43	Definições <i>Parameter Optimization Loop Start</i>	41
44	Tipos de modelos	41
45	<i>Cross-validation</i> no <i>Workflow</i>	42
46	<i>Cross-validation</i> com <i>k-fold=10</i>	42
47	Definições dos nodos <i>X-Partitioner X-Aggregator</i>	42
48	Parte final do <i>tuning</i>	43
49	Definições <i>Variable Loop End</i>	43
50	Processo de obtenção do modelo	44
51	Divisão dos dados	45
52	Modelo final da MLP	46
53	Processo de implementação	47
54	Divisão dos dados	47
55	Processo completo da 2 ^a abordagem	48
56	Processo de implementação	48
57	Resultados do modelo <i>Random Forest</i>	49
58	Métricas da MLP	49
59	Gráfico da <i>accuracy</i> da MLP	49
60	Gráfico da <i>loss</i> da MLP	50
61	<i>Accuracy</i> dos <i>k</i> modelos	50
62	<i>Loss</i> dos <i>k</i> modelos	51
63	Métricas da MLP	51



LISTA DE FIGURAS

64	Gráfico da <i>accuracy</i> da MLP	51
65	Gráfico da <i>loss</i> da MLP	52
66	<i>Amazon Web Services</i>	53
67	Processo de desenvolvimento do servidor	54
68	<i>React Native</i>	55
69	Logótipo da aplicação	55
70	Plataforma <i>mobile</i>	56



1 Introdução

Neste capítulo é feita uma contextualização do projeto e da motivação que levou ao desenvolvimento do mesmo. Os objetivos são expostos e é apresentada uma estrutura geral do relatório.

1.1 Contextualização

A computação está cada vez mais a marcar a sua presença no desporto, conseguindo atingir os mais diversos desportos, as mais diversas modalidades, o que leva aos criadores de software a desenvolver mais *software*, essencialmente, aplicações móveis. Esta tentativa de introduzir a tecnologia na prática de desporto ou exercício físico é bem aceite pela sociedade, ou seja, tanto profissionais como meros amadores, pois vêm estes avanços tecnológicos como auxiliares para este tipo de atividade.

O foco principal deste projeto está direcionado para um tipo específico de indivíduos, os VRUs, que são utilizadores vulneráveis da estrada, os ciclistas e os corredores. Definido o foco do projeto, pretende-se elaborá-lo de modo a criar uma plataforma que estes VRUs possam consultar para saber se aquele momento é seguro ou não para a prática de desporto.

A abordagem consiste no tratamento de dados recolhidos relativos à cidade de Braga, tais como previsões de fluxo de trânsito de 24 horas, previsões meteorológicas, níveis de poluição e quantidade de incidentes de trânsito. Este tratamento será elaborado seguindo o conceito de *Data Mining*, mais concretamente a metodologia *CRISP-DM*, com o intuito de realizar a análise destes dados e o posterior tratamento dos mesmos.

Com os dados processados, o passo seguinte da abordagem definida será, através de *Deep Learning*, mais concretamente as redes neurais, elaborar uma rede neuronal que possa aprender com estes dados tratados.

Por último estará a associação desta rede com a plataforma final, que se irá desenvolver em *React Native* e será onde os VRUs poderão solicitar recomendações.

1.2 Motivação

O envolvimento do grupo nesta iniciativa advém de vários fatores, como o atual crescimento exponencial da utilização de AI (Inteligência Artificial) e *Machine Learning*, áreas em que todos os elementos do grupo estão envolvidos, e o simples facto de estar relacionado com desporto, algo bastante diferente do normal para todos os elementos do grupo, algo que despertou ainda mais interesse.

Por um lado, o grupo tinha curiosidade em perceber melhor estes conceitos mencionados, bem como perceber ao certo o conceito de *Data Mining* e aquilo que envolve, sendo ao mesmo tempo encarado como uma oportunidade para adquirir conhecimentos nestas diversas áreas que estão em claro desenvolvimento e que são cada vez mais prevalentes e com tendência a continuar a sê-lo graças aos constantes desenvolvimentos a nível de hardware e software informático.

Por outro lado, tanto o desporto como a atividade física são áreas em que nenhum dos elementos do grupo tinha trabalhado ou desenvolvido algum projeto no âmbito destas áreas, mas que todos tinham vontade de explorar e aprender mais. Por isso, esta foi uma oportunidade de tentar juntar o útil ao agradável.

1.3 Objetivos

Tendo em conta o que foi dito anteriormente, o principal objetivo deste projeto o desenvolvimento de uma plataforma de recomendação e informação tanto para corredores como para ciclistas através dos dados que nos foram fornecidos sobre a cidade de Braga.

Uma análise completa e detalhada será um ponto fulcral deste projeto complementando com um tratamento assertivo, averiguando todos os potenciais riscos de perigosidade para os VRUs que contém todos os conjuntos de dados.

Outro objetivo passará pela criação de uma rede neuronal e consequentemente um modelo de treino onde deverá ser realizado o melhor *tunning* possível para que o treino seja o mais eficaz possível.

1.4 Estrutura do relatório

Este relatório aborda todas as etapas necessárias para a realização do projeto em questão, começando por um estado da arte em que é feita referência a vários temas de importância para o projeto em questão, tais como a metodologia de *Data Mining* a seguir, mais concretamente o *CRISP-DM*, *Machine Learning*, *Deep Learning* e redes neurais.



1 INTRODUÇÃO

De seguida, há uma reflexão acerca do problema que tínhamos em mãos, enumerando os seus desafios e a abordagem escolhida para a sua resolução. O passo seguinte foi a descrição do processo de desenvolvimento, desde as pesquisas informativas na exploração dos dados, às decisões tomadas face ao tratamento dos mesmos até ao treino da rede neuronal com estes dados tratados.

Antes de terminar com algumas conclusões e com uma análise para trabalho futuro, este relatório contém ainda uma fase dedicada às várias experiências que foram realizadas na procura para atingir os nossos objetivos e, ainda, a apresentação final da plataforma desenvolvida para o projeto.

2 Estado da arte

Neste capítulo, é feito um estado da arte relativamente ao conceito de *Data Mining*, mais concretamente a metodologia *CRISP-DM*, mas também relativamente a *Machine Learning*, a *Deep Learning* e às redes neurais, ponto fulcral para o desenvolvimento deste projeto.

2.1 Data Mining

O conceito de *Data Mining* é um processo de tratamento de dados. É uma forma de analisar e processar uma quantidade de dados sob diferentes perspetivas. Esses dados são transformados em informação, que serão úteis nas mais diversas áreas estratégicas.



Figura 1: Processo de *Data Mining*

Mas o que é o propósito do conceito de *Data Mining*? Qualquer tipo de dado necessita de algo que identifique os seus padrões, as suas consistências e os seus relacionamentos com outros dados, para que toda esta informação seja transformada em conhecimento para ser usado em decisões estratégicas no futuro.

Data Mining costuma estar associado à tecnologia. A tecnologia é parte fundamental para o desenvolvimento deste campo. A mineração muitas vezes dá-se por pesquisas na internet, sendo estas processadas por computadores robustos e softwares complexos desenvolvidos para análise de dados.

Contudo, o conceito de mineração de dados não se limita ao uso de tecnologias e ferramentas de computador, mas também é composta por várias áreas de estudo relacionadas entre si. Para compreender melhor o que este conceito contempla, veja-se as 5 principais características desta área:

- Extração, transformação e envio de dados para um sistema de *data warehouse*;
- Armazenamento de dados num sistema de base de dados;
- Acesso aos dados por analistas e aos demais utilizadores;
- Análise de dados através de ferramentas, softwares e tecnologias diversas;
- Apresentação de dados em informação útil e comprehensiva.

A extração e transformação de dados lida com conceitos como a recuperação da informação, a *Machine Learning* e programação de linguagem natural. O estudo baseia-se na ciência da informação e na inteligência artificial, que procura as melhores formas de encontrar as informações de uma fonte e extraí-las.

O estudo da base de dados é uma das principais prioridades de *Data Mining*, uma vez que é necessário lidar com o gestão e manutenção de grandes bases de dados. Os dados minerados precisam de ser armazenados de forma adequada para, posteriormente, serem recuperados e analisados durante a sua análise. Assim que são extraídos e armazenados, estes dados são finalmente disponibilizados aos utilizadores alvo. São estes que vão ser os encarregados de analisar o que foi recolhido. Neste ponto da pesquisa, encontram-se ferramentas de apoio ao profissional e a análise de dados é feita com conhecimentos em matemática, estatística e probabilidade.

Por fim, as apresentações dessas informações podem ser feitas através de relatórios, gráficos e até parâmetros para interpretação de outros softwares de decisão.

2.2 CRISP-DM

Em que consiste o *CRISP-DM*? O *CRISP-DM* é a abreviação de *Cross Industry Standard Process for Data Mining*, ou seja, é um Processo Padrão para Inter-Indústrias para Mineração de Dados. É uma metodologia de *Data Mining* que descreve abordagens mais comuns utilizadas por especialistas em mineração de dados para atacar vários problemas.[9]

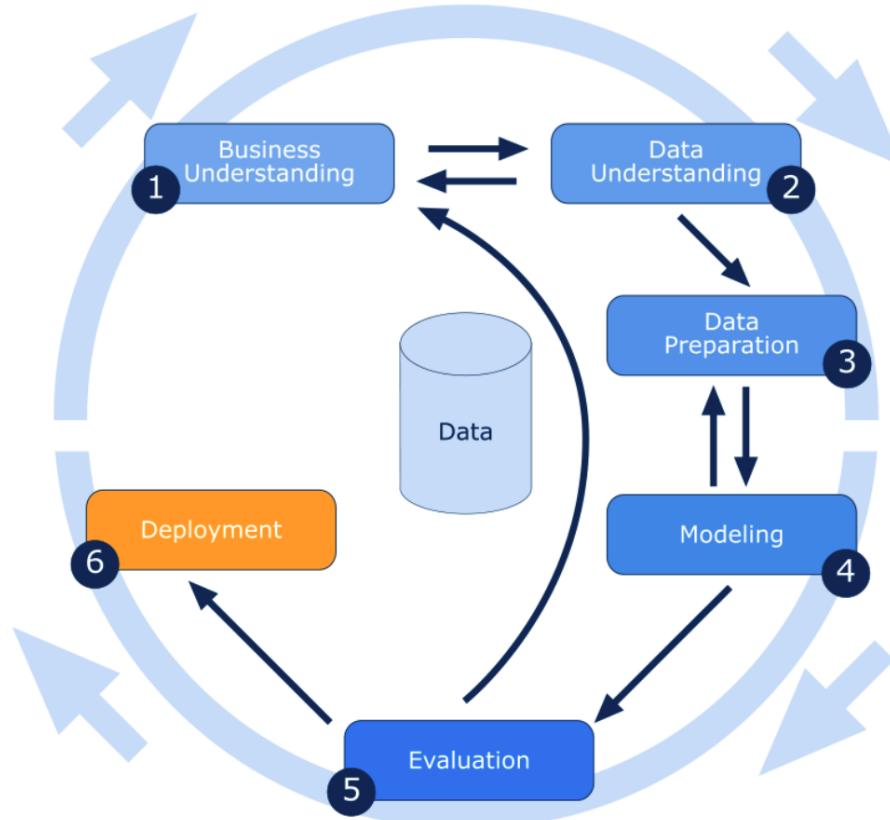


Figura 2: Metodologia *CRISP-DM*

Como se pode verificar pela figura 2, esta metodologia é composta por várias etapas.[19][20][21]

2.2.1 Business understanding

A primeira fase passa por identificar o problema a ser resolvido consoante o panorama atual em que fatores como stock de recursos, requisitos, riscos e custos e benefícios são colocados em cima de mesa para viabilização do projeto. Nesta primeira fase é necessário definir objetivos, plano de mineração de dados e plano de trabalhos.

- **Definição de objectivos:** Traduzir o problema para um cenário empresarial, estabelecer uma relação cliente/empresa. Definição de componentes a serem introduzidas no mercado, e análise e comparação de produtos provenientes de competidores.;
- **Plano Mineração de dados:** Interpretar o problema em objectivos, especificar o estilo e os critérios do problema a ser resolvido. Descrever os outputs esperados do projecto e a forma como devem ser definidos
- **Plano de trabalhos:** Definir um processo inicial, discutir a sua execução entre os intervenientes. Identificar os objectivos principais e as técnicas a serem utilizadas para a resolução desse problema. Nesta fase já são descritas as ferramentas a serem usadas para implementar o projecto. Realçar quais serão à partida os principais obstáculos na execução do projecto

2.2.2 Data understanding

Esta fase é o primeiro embate com os dados a serem trabalhados, primeiro passa por adquiri-los e passá-los para a ferramenta ideal que permita observá-los e ter uma noção que tipo de dados estão a ser coleccionados.



- **Colecção de dados:** indicar de forma transparente a origem dos dados, que métodos foram usados para obtê-los, p.e, se são de acesso privado, quais as credenciais fornecidas para ter acesso autorizado. Indicar quantitativamente os dados em cada campo e se cumprem critérios definidos anteriormente;
- **Exploração de dados:** Esta fase pode ser feita a “olho” ou de forma mais complexa, usar *queries* específicas para analisar em detalhe certas componentes, p.e, atributos chave, relações entre atributos, resultado de agregações, atributos com frequência redundante e estatísticas gerais;
- **Plano de trabalhos:** Definir um processo inicial e discutir a sua execução com os intervenientes. Identificar os objectivos principais e as técnicas a serem utilizadas para a resolução desse problema. Nesta fase já são descritas as ferramentas a serem usadas para implementar o projecto. Realçar quais serão à partida os principais obstáculos na execução do projecto;
- **Qualidade dos Dados:** Aqui são respondidas algumas questões sobre o estado dos dados, p.e, se os dados obtidos estão completos, se todos os casos estão abrangidos, se são valores admissíveis e não falsos, se há valores em falta e, se sim, se são frequentes?

2.2.3 Data preparation

Seleccionam-se os dados que vão ser usados para análise. O critério de escolha será com base nos dados que melhor se adequam a minerar. Esta componente envolve gastar bastante tempo e recursos durante a implementação de um produto, derivado á passagem de uma componente de dados crua para uma pronta para ser consumida.

- **Selecionar Dados:** Escolha e análise de dados de forma racional, bem como a explicação pela selecção de alguns dados e outros não;
- **Limpar Dados:** Subsets que não interessem são descartados, inserção ou adaptação de dados pode ocorrer para melhorar a qualidade dos dados, técnicas de modelação são recomendadas para este efeito. Estas acções devem ser sempre justificadas;
- **Construção de dados :** Operações que envolvem derivações de atributos ou criação de registo;
- **Derivação de Atributos:** Através de atributos já existentes, criar novos atributos a partir desse próprio registo;
- **Criação de registos:** Registos que não aparecem nos dados iniciais mas que são necessários para criar um modelo são introduzidos de modo a explicitar determinados casos;
- **Integração de Dados:** Após modificações nos dados é necessário juntar de forma correta e coerente, e existem duas formas de o fazer: unir ou agrregar. Unir dados - Tabelas são adicionadas de forma conjunta de acordo com o atributo em questão;
- **Agregações:** Após consulta dos valores referentes a múltiplos registo e/ou tabelas, agregá-los numa só entrada;

2.2.4 Modeling

Aqui é selecionada qual a técnica de modelação a ser usada. Apesar de nesta altura de implementação já haver uma ideia prévia de qual será a mais apropriada, aqui será escolhida com maior detalhe, p.e, regressão linear, clustering hierárquico, redes neurais etc... Mas caso seja usada mais que uma abordagem esta subfase deve ser aplicada independentemente.

A cada modelo deve ser interpretado consoante os conhecimentos na área, se ocorreu sucesso ou se ainda há parâmetros a modificar, de realçar que esta fase apenas considera modelos e não a respectiva avaliação. Os modelos testados nesta fase devem ser comparados e se possível criar um rank consoante a performance e/ou complexidade do mesmo

- **Técnica de Modelação:** Documentar qual a técnica a ser usada;
- **Afirmações do modelo:** Qualquer abordagem perante os dados e adaptações consoante o modelo a ser implementado devem ser explicadas aqui;



- **Gerar Protótipo:** Testar e avaliar a qualidade de um modelo protótipo antes de proceder a uma fase seguinte, só após testes a datasets específicos para treino e testes, e consequente validação, se constrói um modelo;
- **Definição de parâmetros:** Com a ferramenta previamente escolhida, explorar e adaptar os diferentes parâmetros oferecidos para melhor inclusão do dataset a ser usado;
- **Modelos:** Primeiro modelo criado por uma ferramenta numa componente prática e não teórica;
- **Descrição do Modelo:** Após testes, apontar os resultados obtidos e interpretá-los de acordo com expectativas e definir advertências encontradas ;
- **Revisão do modelo:** Resumir os resultados desta tarefa, através dos prós e contras através da performance observada;
- **Revisão dos parâmetros do modelo:** A cada iteração do modelo modificar os parâmetros definidos de modo a obter melhor performance do modelo;

2.2.5 Evaluation

Após os resultados obtidos por vários modelos, deve-se escolher qual o modelo que oferece melhores resultados para o dataset em questão, e analisarmeticulously cada desvantagem desse modelo, porque uma desvantagem forte pode impedir a execução do projecto, p.e, performance em tempo real, apesar de ser um modelo fiável, pode não ser eficiente em termos de performance e tempo e isto pode criar problemas de aplicação a nível real.

Caso os modelos sejam satisfatórios em todos os aspectos, deve-se rever todo o processo desde o ínicio até este ponto de modo a assegurar-se que todos os passos foram bem executados e nenhum interveniente foi ignorado.

Após todos estes checkpoints estarem analisados e validados o próximo passo será a implementação junto das entidades responsáveis e o ambiente indicado.

- **Análise dos resultados de data mining:** Comparar os resultados obtidos com os planos iniciais do projeto, se cumprem os requisitos implementados;
- **Modelos Aprovados:** Todos os modelos que estão conforme os critérios e requisitos são considerados aprovados para futuras fases;
- **Revisão do processo:** Resumir todos os processos de revisão e realçar todas actividades que possam ter sido ignoradas ou desprezadas;
- **Lista de possíveis acções:** Listar todas as potenciais acções, devidamente justificadas a aplicar nos próximos passos;
- **Decisão:** Descrever como se deve proceder para o próximo passo de forma racional;

2.2.6 Deployment

Com todas as análises dos resultados previamente obtidos, é agora altura de determinar uma estratégia de implementação no mercado, toda a documentação que servirá de suporte para manutenção e suporte do projecto, todos os aspectos empresariais tem que ser definidos junto com as entidades responsáveis, esta fase é crucial para o sucesso do projeto, porque aqui não depende da equipa que desenvolveu mas do mercado. Manutenção e suporte tem de ser acompanhados nesta fase dura de implementação, que pode ter prazo alargado.

E por fim fazer uma análise ao projeto em geral, realçar pontos que correram bem e outros nem tanto.

Plano de Implementação - Resumir a estratégia de implementação e todos os passos necessários para atingi-la.

- **Plano de Implementação:** Resumir a estratégia de implementação e todos os passos necessários para atingi-la;
- **Monitorização e Suporte:** Definir uma estratégia de suporte e manutenção;
- **Relatório Final:** Relatório escrito de acordo com o processamento de síntese de mineração de dados que inclui todos os passos do projecto descritos de forma resumida;

- **Apresentação Final:** Passo opcional, em que o projecto é apresentado ao cliente com resultados obtidos como prova de decisão;
- **Documentação:** Todos os pontos envolvidos durante o projecto, anomalias, bugs, imprevistos devem estar escritos num só sítio, para que intervenientes no futuro tenham noção da cronologia temporal;

É necessário referir que existem inúmeras vantagens na utilização desta metodologia como, por exemplo:

- Independente de Indústria: O mesmo processo pode ser aplicado ao analisar dados comerciais, financeiros, de recursos humanos, produção industrial, serviços prestados, entre outros;
- Independente da Ferramenta;
- Tem relação próxima com os modelos de processos de KDD.

2.3 Machine Learning

Machine Learning é um método de análise de dados que automatiza a construção de modelos analíticos. É um ramo da inteligência artificial que se baseia na ideia de que sistemas podem aprender com dados que lhe são fornecidos, capaz de identificar padrões e tomar decisões com o mínimo de intervenção humana possível.

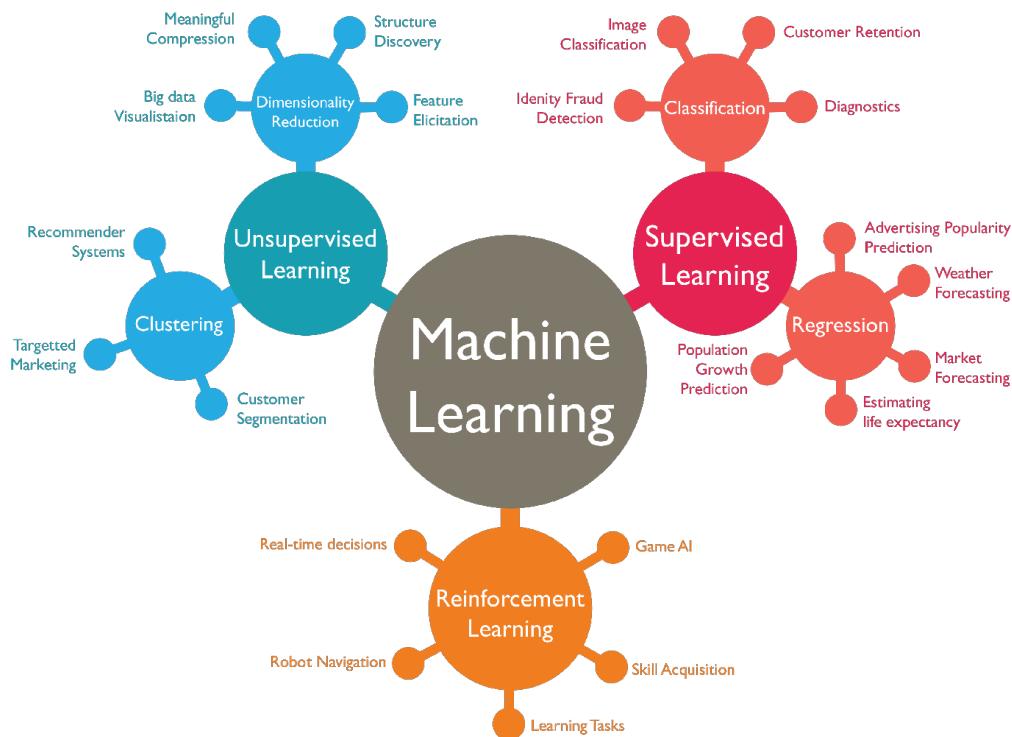


Figura 3: *Machine Learning* e os seus algoritmos

À medida que o tempo passa e graças aos avanços tecnológicos evidentes da atualidade, o *Machine Learning* também evoluiu. Nasceu do reconhecimento de padrões e da teoria de que os computadores podem aprender sem serem programados para realizar tarefas específicas até que pesquisadores tentaram que as máquinas aprendessem através de dados fornecidos. Isso não é uma ciência nova, mas uma ciência que está a ganhar um novo impulso na atualidade.

O aspecto iterativo de *Machine Learning* é importante, pois quando os modelos são expostos a novos dados, estes são capazes de se adaptar independentemente, aprendendo com computações anteriores para produzir decisões e resultados confiáveis e passíveis de repetição.

Inúmeros algoritmos de *Machine Learning* existem há muito tempo, embora a capacidade de aplicar vários cálculos matemáticos complexos à *big data* automaticamente e repetidamente é um desenvolvimento recente.[8]



2.4 Deep Learning

O *Machine Learning*, tal como foi referido anteriormente, está incluído no ramo da Inteligência Artificial e engloba vários sistemas de aprendizagem, podendo ser usado em tarefas diversificadas em que seja necessário que um computador ou uma máquina aprenda algo de modo a exibir um determinado comportamento. A título de exemplo, pode ser usado para tarefas simples, mas de grande importância, como detetar *spam* num *email* e, assim, automaticamente colocar esses *emails* numa pasta adequada ou eliminá-los.

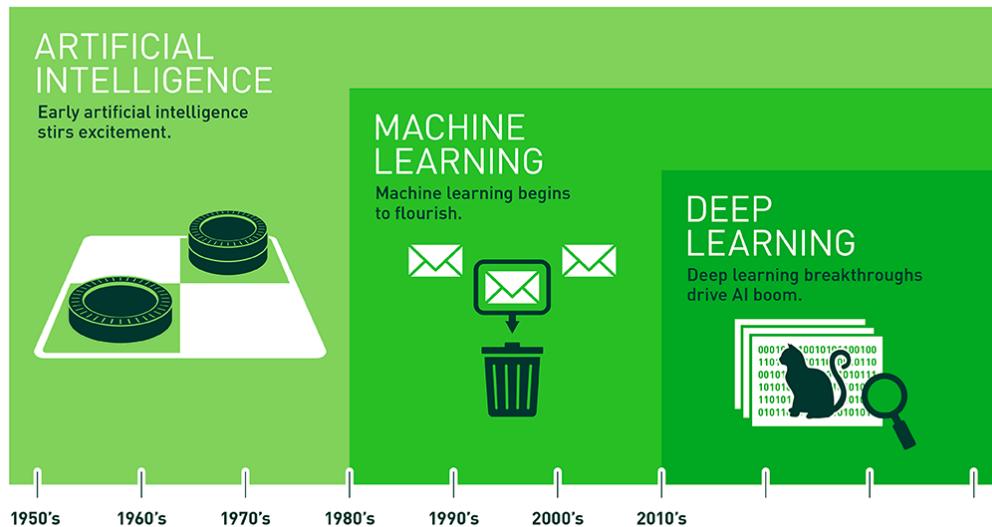


Figura 4: Enquadramento do *Deep Learning* na Inteligência Artificial [1]

Como se pode ver na figura 4, na última década, surgiu o *Deep Learning*, que é uma subárea do *Machine Learning*, em que os modelos possuem várias camadas, camadas essas que são compostas por unidades de processamento não-linear e precisam de grandes quantidades de dados para treino e de uma grande capacidade de processamento. Por esse motivo, apenas recentemente tem sido possível um maior desenvolvimento desta área. O *Deep Learning* destaca-se por poder ser usado para tarefas mais complexas, como por exemplo, a classificação de imagens.

Este desenvolvimento do *Deep Learning* tem sido acompanhado de uma grande utilização em várias áreas e aplicações de muitas empresas e talvez um dos melhores exemplos disso seja a Google, que tem assistido a um crescimento exponencial da utilização do *Deep Learning* nas suas aplicações, sendo isto visível no gráfico 5.

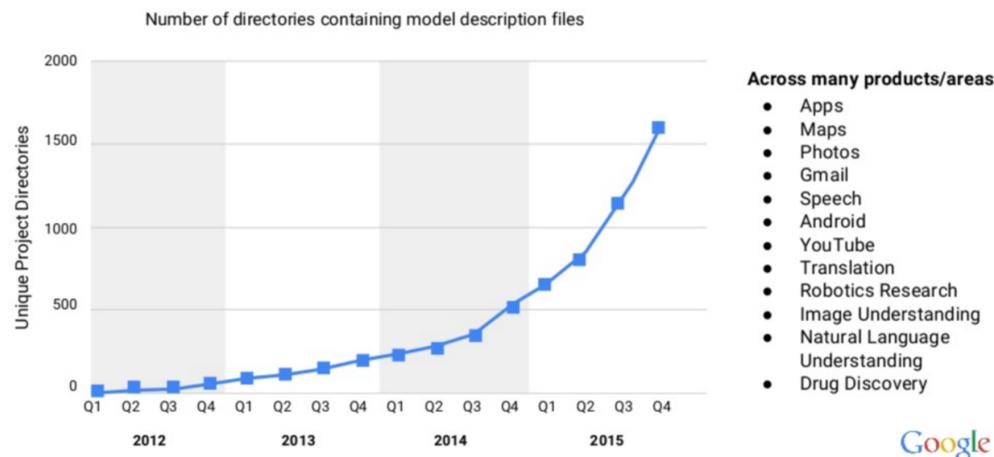


Figura 5: Utilização de *Deep Learning* na *Google*



2.5 Redes Neuronais

As redes neurais são inspiradas na estrutura e funcionamento do cérebro humano, sendo basicamente constituídas por várias unidades de processamento com capacidade de aprendizagem que são chamadas de neurónios e encontram-se interligadas. Têm, por isso, vários componentes fundamentais:

- **Neurónio:** é a unidade computacional das redes neurais e tem capacidade de aprendizagem;
- **Axónio:** via de comunicação entre os neurónios;
- **Sinapse:** ponto de ligação entre axónios e neurónios. A cada sinapse é atribuído um valor que simboliza o peso.

Na figura 6, é possível visualizar a estrutura biológica de onde surgiu o nome e a inspiração para esses componentes.

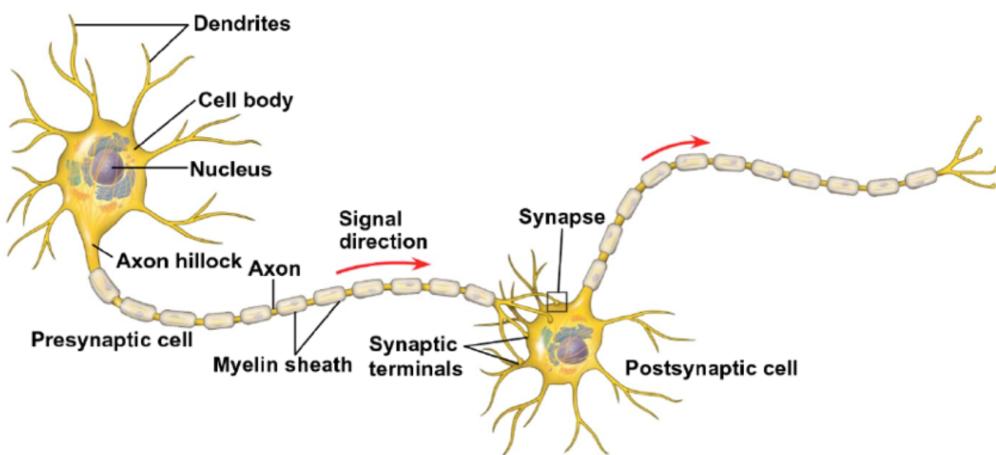


Figura 6: A inspiração biológica das redes neurais

Os neurónios interligados organizam-se em várias camadas por onde circula a informação na rede, mais concretamente, em uma camada de input, uma ou mais camadas intermédias (ou escondidas), e uma camada de output, tal como se pode observar na figura 7.

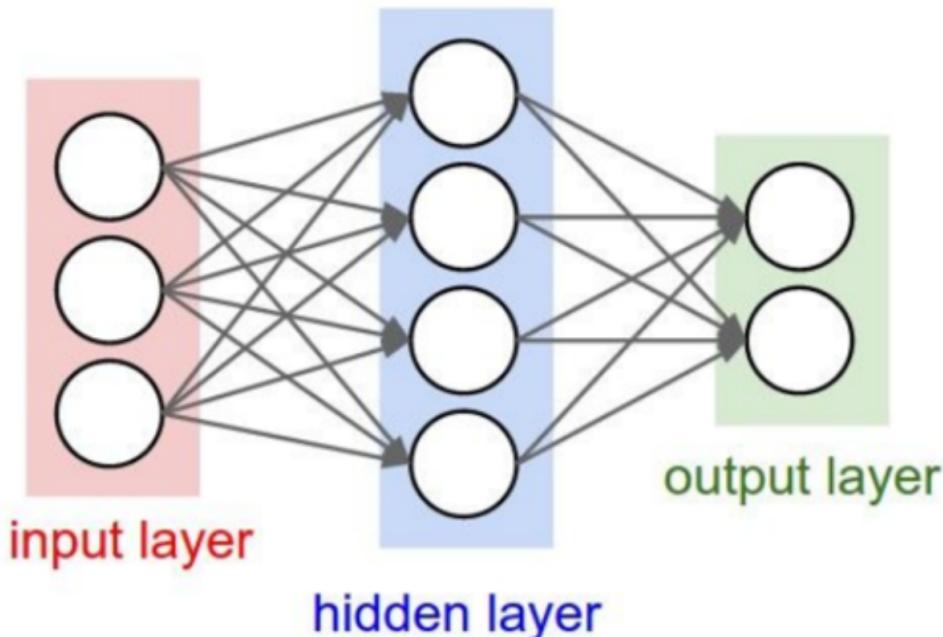


Figura 7: Exemplo de arquitetura de uma RNA

2.5.1 Redes Neuronais Recursivas

As Redes Neuronais Recursivas (RNNs) distinguem-se das redes habituais pelo facto de possuírem a noção de estado e pela existência de uma conexão com as várias passagens ao longo do tempo. Ou seja, cada neurónio recebe informação não apenas da camada anterior da rede, mas também informação dele próprio, proveniente da iteração anterior.

As RNNs têm um ciclo de alimentação em que o output de uma camada alimenta a seguinte, juntamente com o próximo *input*. Estas redes podem ter ciclos e são ideais para tarefas de processamento onde os dados das camadas anteriores devem ser considerados.

Além disso, têm a noção de memória interna, o que permite saber qual o comportamento temporal feito até ao momento, tornando-as ideais para conjuntos de dados em que as suas amostras sejam compostas por grandes sequências. Na figura 8 encontra-se uma possível arquitetura de uma RNN em que é visível uma recorrência característica destas redes.

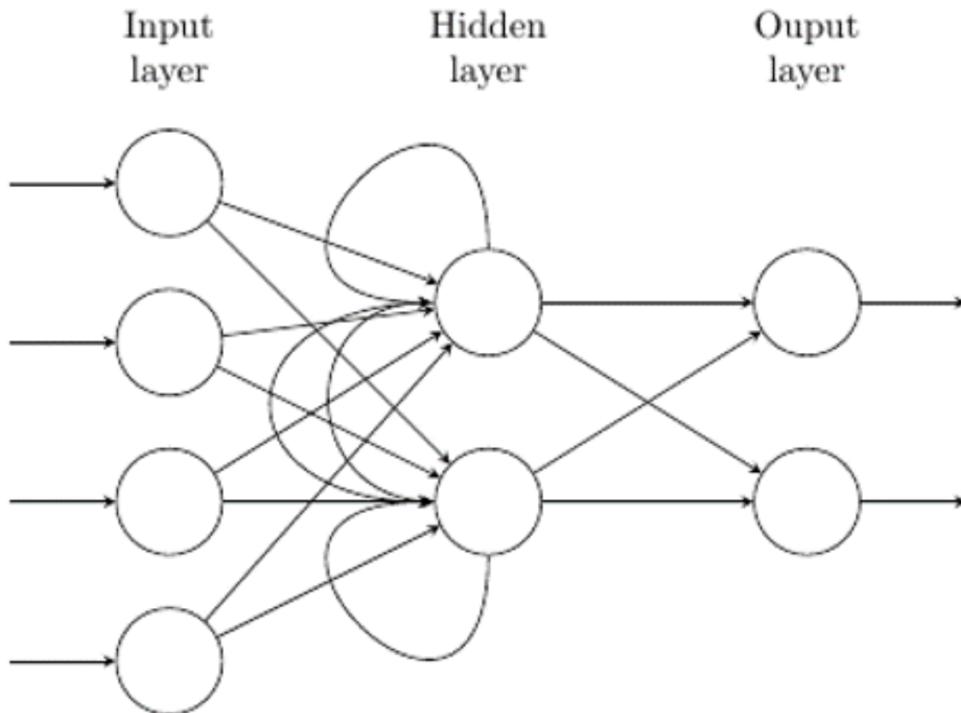


Figura 8: Exemplo de arquitetura de uma RNN

2.5.2 Vantagens e Desvantagens

As vantagens associadas ao uso das redes neuronais passam pela sua adaptabilidade e elevada tolerância a falhas, visto que a falha de um neurónio não inutiliza estas estruturas, continuando a rede a funcionar normalmente. Mais concretamente, as redes neuronais recursivas são, além disso, ideais quando os dados das épocas anteriores devem ser considerados.

Por outro lado, o facto de o treino das redes ser muitas vezes um processo bastante demorado e serem necessários grandes volumes de dados são alguns dos pontos menos positivos quando se avaliam os prós e os contras de usar este tipo de implementação no desenvolvimento de um projeto, tendo em conta as limitações de tempo e o facto de ser necessário um grande poder computacional que poderá nem sempre ser facilmente acessível. Além disso, as redes neuronais em muito se assemelham a "caixas negras", o que torna mais complexo perceber como utilizar e manipular as mesmas para atingir os objetivos desejados, levando, muitas vezes, a um processo de tentativa e erro em que não há garantias.



2.5.3 Áreas de Aplicação

A maioria dos modelos usados em *Machine Learning* e *Deep Learning* baseia-se em redes neuronais e, por isso, estas estão presentes em várias áreas tais como:

- *Internet e Cloud* (classificação de imagens, reconhecimento de emoções);
- Medicina e Biologia (descoberta de células cancerígenas, avaliação do grau de diabetes);
- Media e Entretenimento (tradução em tempo real, procura de vídeos);
- Segurança e Defesa (vídeo vigilância);
- Máquinas autónomas (reconhecimento de sinais de trânsito e de pedestres).

Tendo em conta isto, dá para perceber que as redes neuronais são já usadas para tarefas de elevada importância, o que pode ser encarado como uma prova da sua qualidade e eficiência quando usadas adequadamente.

2.5.4 Ferramentas de Desenvolvimento

Existem várias soluções no mercado para a implementação de redes neuronais e de seguida é feita a comparação de algumas dessas soluções para a linguagem de programação *Python*, que é também a linguagem escolhida para o desenvolvimento deste projeto.

Atualmente, encontra-se disponível uma grande variedade de bibliotecas para os diferentes *frameworks* existentes no mercado, tais como:



Figura 9: Ferramentas de Desenvolvimento

2.5.4.1 *TensorFlow* Este *framework* foi criado pela *Google* para substituir o *Theano* e é um dos mais populares atualmente, o que o torna numa das melhores escolhas para quem está numa fase inicial dada a quantidade de tutoriais e guias disponíveis. Outra vantagem importante é o facto de permitir paralelismos, podendo ser executado usando múltiplos CPUs ou GPUs. No entanto, é tipicamente mais lento que outros *frameworks* e o facto de ser uma biblioteca de baixo nível tornam-no também um pouco complexo.[2]

2.5.4.2 *Keras*: É uma biblioteca recente, mas que está a crescer rapidamente e já tem uma documentação com alguma qualidade.[3] Tenta resolver o problema da complexidade de utilização do *TensorFlow* de modo a que seja possível criar redes neuronais eficientes com poucas linhas de código. Por isso, pode ser configurada para funcionar com base no *TensorFlow* e especula-se que se venha a tornar na API de *Python* por defeito quando se quiserem usar RNAs.[4]



2 ESTADO DA ARTE

2.5.4.3 Caffe: O *Caffe* e o *Caffe2* (versão mais escalável e menos pesada) pertencem ao *Facebook* e são usados principalmente para Redes Neuronais Convulsionais que são muito úteis na área da Visão por Computador. É bastante bom para usar em redes com *feedforward* e em processamento de imagem e permite treinar modelos sem escrever qualquer código. No entanto, não é uma boa escolha para Redes Neuronais Recorrentes e tem tido um desenvolvimento lento, estando a sua utilização a diminuir e a ficar menos relevante.[5]

2.5.4.4 PyTorch: Tal como a anterior, a biblioteca *PyTorch* foi disponibilizada pelo *Facebook* e a partir de outra biblioteca bastante poderosa, *Torch*, mas que foi desenvolvida para a linguagem *Lua*. Assim, o *Python* recebeu uma biblioteca com muitos modelos pré-treinados e que é bastante boa para redes recursivas. As suas desvantagens são o facto de ser necessário para o utilizador escrever o seu próprio código de treino e a documentação não é tão boa como para alguns dos *frameworks* anteriores.[6]

2.5.4.5 Theano: O *Theano* era a biblioteca mais usada para *Deep Learning*, suportando tanto CPU como GPU e podendo até ser usado através do *Keras*, no entanto, desde o surgimento do *TensorFlow*, *Caffe* e *PyTorch*, tem vindo a perder importância até que foi anunciada uma versão final a partir da qual não seriam feitos novos desenvolvimentos, garantindo-se apenas uma manutenção mínima e, por isso, a sua utilização é cada vez menor.[7]



3 O problema e os seus desafios

Neste capítulo, começa-se por fazer uma abordagem aos dados de entrada e aos dados tratados, passando-se por todos os desafios com os quais o grupo se deparou, como foi o caso da investigação realizada face aos diferentes parâmetros dos dados fornecidos e o desenvolvimento de uma rede neuronal competente.

3.1 Problema

Uma vez que os temas *Deep Learning* e redes neurais são temas que o grupo não está muito familiarizado, adivinhariam-se alguns obstáculos ao longo da execução do projeto. No entanto, e sendo esse um dos motivos de o grupo ter escolhido este projeto, são temáticas que o grupo irá ter contacto neste período do ano letivo. Desde logo foi necessário fazer um estudo específico só destes novos temas.

3.1.1 Dados de entrada

A ideia base para o projeto é a de treinar a rede partindo de um conjunto de dados tratados. Logo os dados de entrada serão os dados que nos foram fornecidos que irão passar por um processo de análise detalhado para que o grupo determine quais deles são realmente necessários de manter, quais necessitarão de um tratamento/transformação e quais deles podem gerar novos tipos de dados. A exploração destes dados será abordada detalhadamente no Capítulo 4.

3.1.2 Dados tratados

Feita a análise dos dados de entrada, a ideia passa por se tratar os dados com base na mesma e, como tal, terá de ser feito um *parsing* dos ficheiros de entrada de modo a que se possa extrair o conteúdo de relevo para garantir a qualidade de um futuro treino de uma rede neuronal, tal como será explicado nos Capítulos 5 e 6.

O objetivo deste tratamento dos dados passará por criar um ficheiro com todos os dados essenciais para fornecer à rede neuronal que se irá desenvolver.

3.2 Desafios

Existem vários desafios na resolução deste problema, começando por um tratamento dos dados completo e minucioso. Esta abordagem mais cuidada ao tratamento de dados promove a possibilidade de alcançar-mos mais tarde um *dataset* capaz de satisfazer com qualidade as necessidades do projeto a desenvolver. Esta qualidade superior que procuramos alcançar permitirá uma aprendizagem mais eficiente e capaz de produzir melhores resultados através da rede neuronal que o utilizar para treino.

Um segundo desafio será o desenvolvimento da rede neuronal a utilizar, uma vez que esta se trata de uma temática recente ao grupo, tendo sido leccionada ao longo do desenvolvimento deste projeto, sobre a qual teremos de investigar para adquirir os conhecimentos necessários para obter os melhores resultados possíveis.

Por último, a criação da plataforma de recomendações para os VRUs que será construída em *React Native*, tecnologia que também é recente para nós. No entanto, o compromisso e dedicação de cada elemento do grupo permitir-nos-á ultrapassar todos estes desafios para terminar o projeto, conquistando cada um dos desafios que surjam.

3.3 Plano de Desenvolvimento

De modo a resolver este problema bem como responder a estes desafios, o desenvolvimento deste projeto será composto por 4 fases principais:

- Exploração de dados
- Tratamento de dados
- Desenvolvimento e treino
- Criação da plataforma

Numa primeira fase será realizada uma exploração (investigação) detalhada de todos os dados que nos são fornecidos de maneira a que possamos detetar quais dos dados são realmente relevantes, quais não o são e porquê. Esta análise prévia permitirá com que seja feita uma escolha mais consciente na fase seguinte.



3 O PROBLEMA E OS SEUS DESAFIOS

Numa segunda fase será realizado o tratamento dos dados, com este processo pretendemos, como os dados inicialmente fornecidos, gerar um ficheiro de dados final que possa ser utilizado para o desenvolvimento e treino da rede neuronal que queremos implementar.

Já numa terceira fase, com o ficheiro de dados final, será construída uma rede neuronal cujo o intuito desta, será aprender com o conjunto de dados tratados para que possa fazer as sugestões mais acertadas ao problema proposto.

Por último, numa última fase, pretende-se desenvolver uma plataforma que através do conhecimento que podemos retirar desta rede, possa recomendar ao utilizador, um *VRU*, se este deve ou não praticar o seu exercício físico exterior. A plataforma que terá o seu *front-end* desenvolvido em *React Native*, como já foi referido, fará a recomendação consoante a hora do dia baseando-se na informação proveniente da rede elaborada.



4 Exploração e Visualização dos dados

Neste capítulo iremos fazer uma exploração (análise) detalhada de todos os dados fornecidos para a realização deste projeto. Será utilizada a ferramenta KNIME onde será possível introduzir todos os conjuntos de dados e permitirá uma melhor análise dos mesmos.

Os conjuntos de dados fornecidos para este projeto continham informações sobre a cidade de Braga capturadas através do uso de sensores pertencentes a entidades públicas que nos emprestaram para o desenvolvimento deste projeto. Os dados de cada *dataset* foram o Fluxo de Tráfego, a Poluição, as Condições Climáticas e os Incidentes de Tráfego.

Levaremos algum tempo para explicar cada *dataset*, bem como as informações nele contidas. Algumas das informações provaram ser estranhas para nós, por isso tivemos que investigá-las para obtermos uma melhor compreensão das mesmas e começar a processar os dados em uma configuração em que concordamos que era prática e bem definida. Também pesquisamos as implicações de certos recursos para determinar melhor quais faixas de valores tinham um certo nível de perigo em termos de saúde e possíveis conexões entre os recursos dos diferentes conjuntos de dados.

Desta forma, podemos atribuir um valor relativamente seguro a cada recurso indicado que nos irá ajudar a treinar nossa rede neuronal. Na figura 10 vemos o *workflow* completo desenvolvido para a exploração de dados.

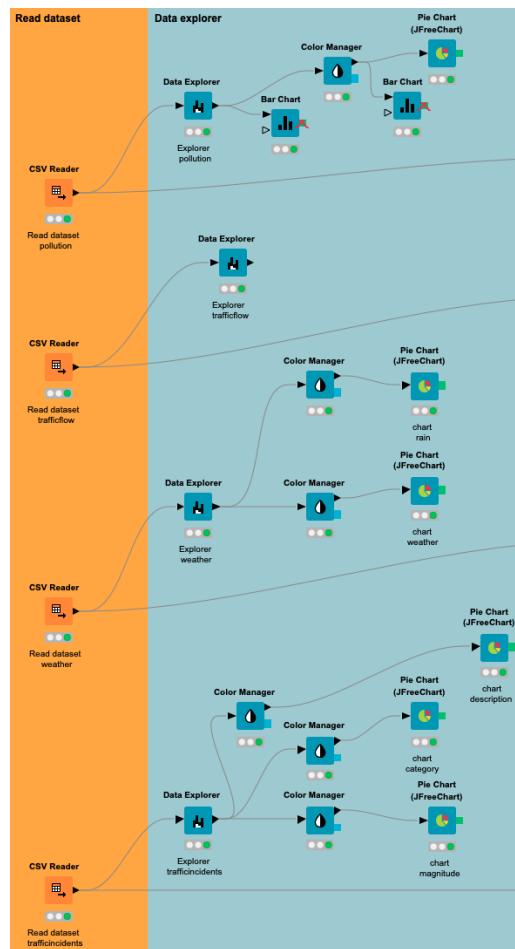


Figura 10: *Workflow* exploração de dados

4.1 Dataset Pollution

Analisaremos primeiro o *dataset* da Poluição que, para além da conter dados sobre a data de criação, a última atualização e o número de sequência, oferece-nos uma visão geral dos valores de concentração do volume de algumas partículas que, em concentrações suficientemente altas, podem representar um risco à saúde de qualquer pessoa, que faça uma corrida ou um percurso de bicicleta.



4 EXPLORAÇÃO E VISUALIZAÇÃO DOS DADOS

Column	Exclude Column	No. missings	Unique values	All nominal values	Frequency Bar Chart
pollution_type	<input type="checkbox"/>	0	6	Nitrogen Dioxide, Particulate Matter 10, Ultraviolet, Ozone, Carbon Monoxide, Sulfur Dioxide	
city_name	<input type="checkbox"/>	0	1	Braga	
last_updated	<input type="checkbox"/>	20234	>999	2019-01-27T01:00:00.000Z, 2019-02-05T09:00:00.000Z, 2019-01-19T05:00:00.000Z, 2019-03-05T09:00:00.000Z, 2019-02-15T1:00:00.000Z, [...], 2019-02-22T17:00:00.000Z, 2019-02-12T00:00:00.000Z, 2019-04-11T14:00:00.000Z, 2019-01-30T18:00:00.000Z, 2019-04-23T16:00:00.000Z	Not all nominal values calculated.
creation_date	<input type="checkbox"/>	0	>1000	2019-03-05 09:05:03.000000, 2019-02-14 07:05:01.000000, 2019-03-05 05:05:06.000000, 2019-04-22 01:58:31.000000, 2019-02-10 01:05:05.000000, [...], 2019-05-09 11:48:41.000000, 2019-03-04 07:05:02.000000, 2019-01-29 03:05:01.000000, 2019-01-20 19:05:02.000000, 2019-01-20 19:05:01.000000	Not all nominal values calculated.

Figura 11: Dados do *dataset Pollution*

A partir deste conjunto de dados, percebemos que será necessário realizar uma análise detalhada das condições de concentração atmosférica de partículas (ou radiação), uma vez que estas podem não apenas criar um ambiente perigoso, como também podem ser o resultado de outra situação retratada num conjunto de dados diferente.

Desta feita, e tal como foi referido anteriormente, face aos dados deste conjunto de dados, tendo em consideração que informações como a data e número de sequência são auto-explicativos, foi necessário dar mais relevância ao tipo de poluição.

Numeric	Nominal	Data Preview	Search: <input type="text"/>																														
<hr/>																																	
<hr/>																																	
<table border="1"> <thead> <tr> <th>Column</th><th>Exclude Column</th><th>Minimum</th><th>Maximum</th><th>Mean</th><th>Standard Deviation</th><th>Variance</th><th>Skewness</th><th>Kurtosis</th><th>Overall Sum</th></tr> </thead> <tbody> <tr> <td>seq_num</td><td><input type="checkbox"/></td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>43845</td></tr> <tr> <td>value</td><td><input type="checkbox"/></td><td>-0.000</td><td>638</td><td>7.724</td><td>17.210</td><td>296.198</td><td>7.573</td><td>173.767</td><td>338661.174</td></tr> </tbody> </table>				Column	Exclude Column	Minimum	Maximum	Mean	Standard Deviation	Variance	Skewness	Kurtosis	Overall Sum	seq_num	<input type="checkbox"/>	1	1	1	0	0	0	0	43845	value	<input type="checkbox"/>	-0.000	638	7.724	17.210	296.198	7.573	173.767	338661.174
Column	Exclude Column	Minimum	Maximum	Mean	Standard Deviation	Variance	Skewness	Kurtosis	Overall Sum																								
seq_num	<input type="checkbox"/>	1	1	1	0	0	0	0	43845																								
value	<input type="checkbox"/>	-0.000	638	7.724	17.210	296.198	7.573	173.767	338661.174																								
Showing 1 to 2 of 2 entries																																	

Figura 12: Dados numéricos contidos no *dataset*

No gráfico da figura 13 podemos verificar quais são as partículas que teremos de explorar.

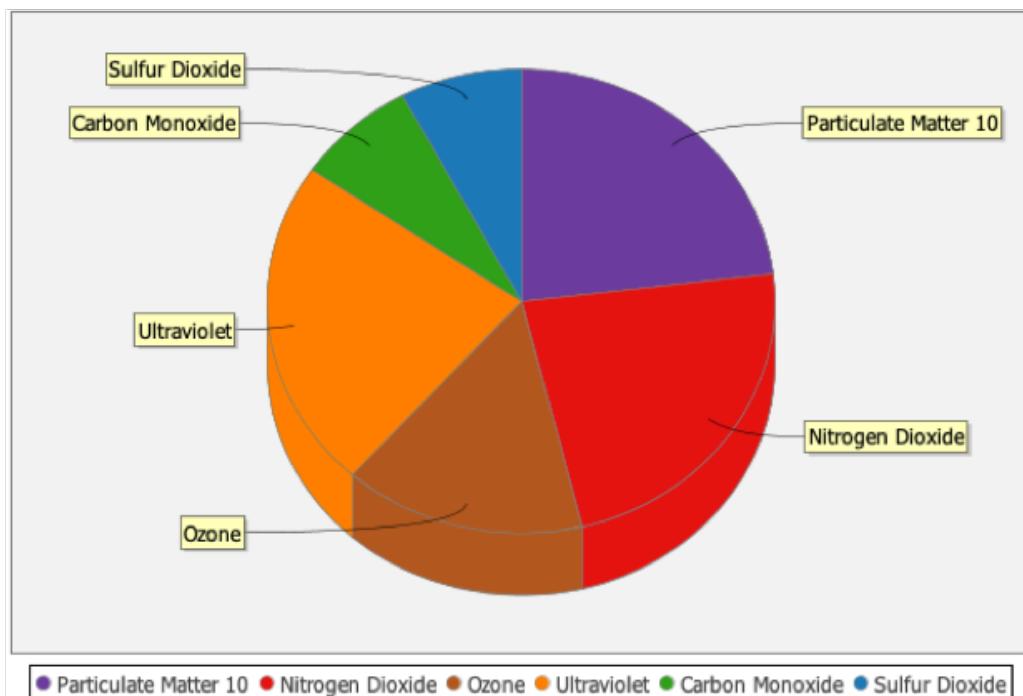


Figura 13: Dados relativos ao tipo de poluição

Destes gases disponíveis para investigação, podemos concluir pelo gráfico da figura 14 que possivelmente os dados a considerar serão o *Particulate Matter 10*, o *Nitrogen Dioxide*, o *Ozone* e o *Ultraviolet*, uma vez que são dados que têm dados consideráveis.

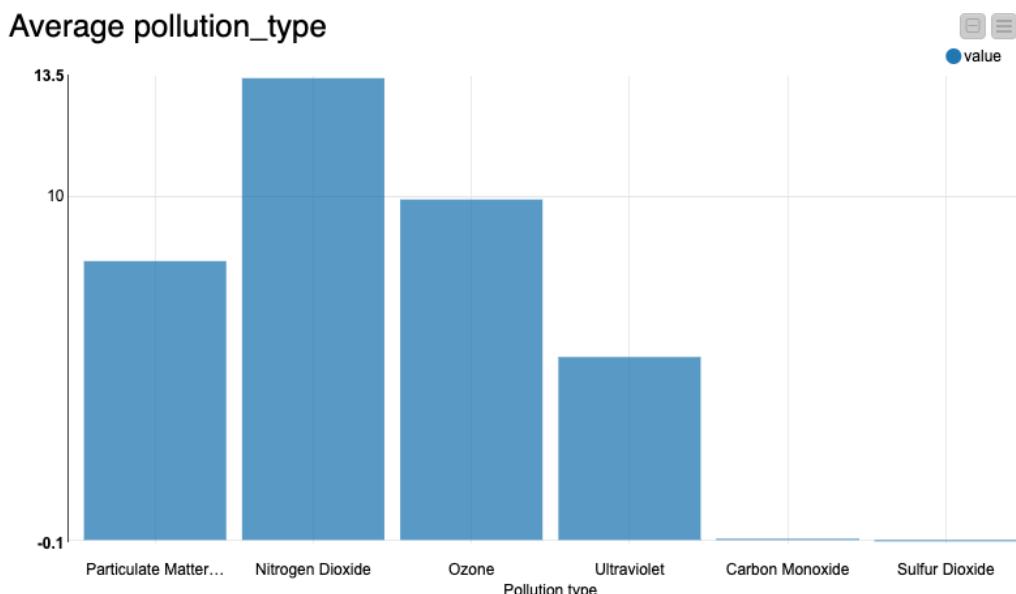


Figura 14: Média dos valores das partículas

De seguida, realizou-se a pesquisa detalhada sobre cada partícula e sobre os valores prejudiciais para o ser humano de cada uma delas, que será apresentada a seguir.

4.1.1 Radiação Ultravioleta

Começando pela radiação Ultravioleta, é necessário ter em consideração que valores excessivos deste tipo de radiação são muito prejudiciais para a saúde de qualquer ser humano.



O excesso de radiação ultravioleta pode provocar várias doenças essencialmente ao nível da pele, tal como o câncer da pele, como é o caso da Melanoma, que é um tipo de câncer que se desenvolve a partir dos melanócitos (são células produtoras de melanina) e ocorre geralmente na pele, mas pode manifestar-se mais raramente na boca, intestinos ou olhos.

Para além das doenças que o excesso deste tipo de radiação pode gerar no ser humano, também pode dar-se o facto de o próprio sistema imunitário do nosso corpo seja impedido de realizar o seu funcionamento adequado bem como o funcionamento das defesas naturais da pele.[10]

Recorrendo ao KNIME e de forma a obter mais informação sobre os valores da radiação Ultravioleta, desenvolveu-se um gráfico onde fosse possível verificar os meses com maior quantidade de radiação, tal como podemos ver na figura 15. Podemos constatar o que já estávamos à espera, ou seja, os valores mais elevados correspondem aos meses de Verão e os valores mais baixos correspondem aos meses de Inverno.

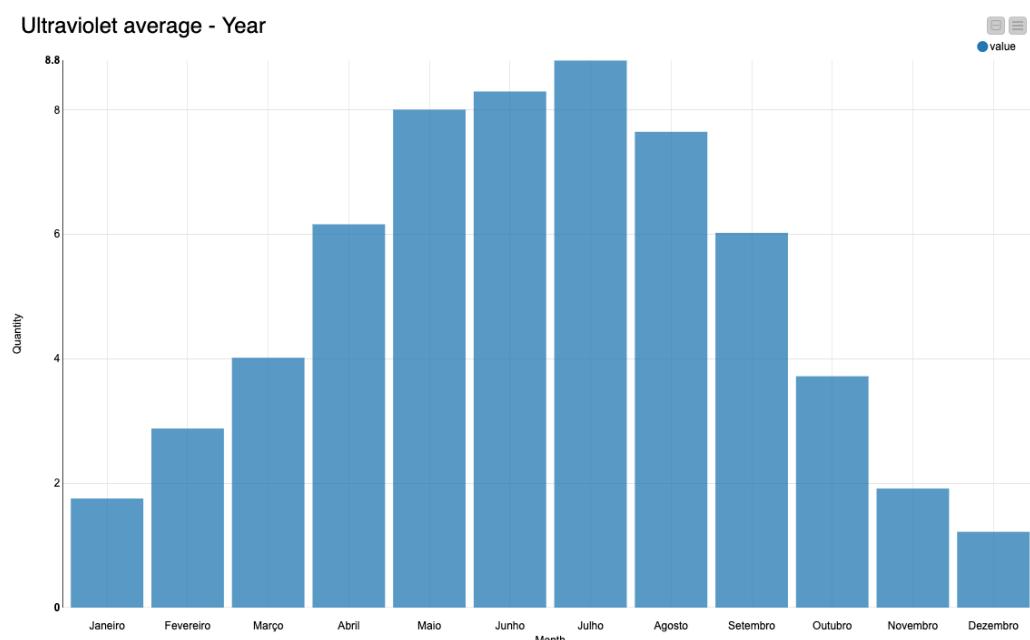


Figura 15: Radiação UV ao logo do ano

Assim sendo, averiguou-se os diferentes índices de radiação UV e o seu risco que cada nível proporciona para um adulto médio para uma posterior comparação e tratamento dos dados deste tipo de poluição.

Índice Ultravioleta:

Índice UV	Color Gráfica	Risco de danos pela exposição desprotegida ao sol, para um adulto médio
0.0–2.9	Verde	Baixo
3.0–5.9	Amarelo	Moderado
6.0–7.9	Laranja	Alto
8.0–10.9	Vermelho	Muito alto
11.0+	Violeta	Extremo

Tabela 1: Tabela radiação UV

4.1.2 Monóxido de Carbono

Quanto ao Monóxido de Carbono sabe-se que é um gás incolor e inodoro que, quando inalado, impede que o sangue transporte oxigénio e não permite que os tecidos o utilizem com eficácia. A realidade é que uma quantidade reduzida deste gás não costuma ser prejudicial à saúde, no entanto, resulta numa intoxicação quando as concentrações de monóxido de carbono no sangue se tornam muito elevadas. O CO desaparece do sangue ao fim de várias horas.



A intoxicação por monóxido de carbono é geralmente o resultado da inalação de uma quantidade excessiva deste gás. **Os sintomas mais comuns são:**

- dores de cabeça;
- tonturas;
- fraqueza;
- vômitos
- dor torácica;
- confusão.

A inalação prolongada pode resultar em:

- perda de consciência;
- arritmias cardíacas
- crise epiléptica;
- morte.

Entre as sequelas a longo prazo estão o cansaço, problemas de memória e problemas ao nível motor. A maior parte dos casos de intoxicação tem origem na inalação do monóxido de carbono produzido na combustão aquecedores ou equipamento de cozinha alimentados e de veículos a motor também a combustíveis fósseis, ou seja, fator de contacto direto com os corredores e ciclistas.[11]

Na tabela 2 está uma gama de valores de concentração prejudicial ao ser humano que se averiguou.

Efeitos das diferentes quantidades de CO no corpo de um ser humano adulto médio:

Concentração	Sintomas
0.0035%	Dor de cabeça e tontura dentro de 6 a 8 horas de exposição constante.
0.01%	Leve dor de cabeça em 2 a 3 horas.
0.02%	Dor de cabeça leve dentro de 2 a 3 horas. Perda de julgamento.
0.04%	Dor de cabeça frontal dentro de 1 a 2 horas.
0.08%	1, náuseas e convulsões em 45 minutos. Insensível dentro de 2 horas.
0.16%	Dor de cabeça, aumento da frequência cardíaca, tontura e náusea em 20 minutos. Morte em menos de 2 horas.
0.32%	Dor de cabeça, tontura e náusea em 5 a 10 minutos. Morte em 30 minutos.
0.64%	Dor de cabeça e tontura em 1 a 2 minutos. Convulsões, parada respiratória e morte em menos de 20 minutos.
1.28%	Inconsciência após 2-3 respirações. Morte em menos de 3 minutos.

Tabela 2: Tabela monóxido de carbono

4.1.3 Dióxido de nitrogénio

A presença de dióxido de nitrogénio no ar deve-se essencialmente à queima constante de combustíveis fósseis. O NO₂ forma-se a partir de emissões de carros, camiões e autocarros, usinas de energia e vários equipamentos que consumam este tipo de combustíveis.

Respirar uma porção de ar com uma alta concentração de NO₂ pode irritar as vias respiratórias. As exposições a este gás em períodos curtos podem agravar doenças respiratórias, particularmente asma, levando ao aparecimento de sintomas de problemas respiratórios (como tosse ou dificuldades em respirar). Exposições mais longas a concentrações elevadas de NO₂ podem contribuir para o desenvolvimento da asma e potencialmente aumentar a susceptibilidade a infecções/doenças respiratórias. É necessário que pessoas com asma, assim como crianças e idosos, que têm um organismo normalmente mais frágil e apresentam maior risco para os efeitos do NO₂ na sua saúde, tenham cuidado redobrado para a inalação de quantidades excessivas deste gás.[13]

Na tabela 3, é possível verificar a gama de valores de NO₂ face às diferentes qualidades do ar e o que fazer em cada caso.

Efeitos no ser humano do dióxido de nitrogénio no ar:[12]



Concentração (ppb)	Qualidade do ar	Considerações
0-50	Boa	Não são esperados impactos na saúde quando a qualidade do ar estiver nesta faixa.
51-100	Moderado	Indivíduos extraordinariamente sensíveis ao dióxido de nitrogénio devem considerar limitar o esforço prolongado ao ar livre.
101-150	Não é saudável para grupos sensíveis	Os seguintes grupos devem limitar o esforço prolongado ao ar livre: <ul style="list-style-type: none">• Pessoas com doenças pulmonares, como asma.• Crianças e adultos mais velhos.
151-200	Pouco saudável	Os seguintes grupos devem evitar esforços prolongados ao ar livre: <ul style="list-style-type: none">• Pessoas com doenças pulmonares, como asma.• Crianças e adultos mais velhos. Todos os outros devem limitar o esforço prolongado ao ar livre.
201-300	Muito pouco saudável	Os seguintes grupos devem evitar todo esforço ao ar livre: <ul style="list-style-type: none">• Pessoas com doenças pulmonares, como asma.• Crianças e adultos mais velhos. Todos os outros devem limitar o esforço ao ar livre.

Tabela 3: Tabela dióxido de nitrogénio

4.1.4 Particulate Matter 10

Analizando agora o *Particulate Matter 10*, sabe-se que o tamanho das partículas está diretamente ligado ao seu potencial de causar problemas de saúde. As partículas pequenas com menos de 10 micrómetros de diâmetro apresentam os maiores problemas para o ser humano, porque podem penetrar profundamente os nossos pulmões e algumas vezes até podem entrar na corrente sanguínea.[16]

A exposição a essas partículas pode afetar essencialmente os pulmões e o coração. Vários estudos científicos vincularam a exposição à poluição de partículas a uma variedade de problemas, incluindo:

- doenças cardiovasculares;
- asma agravada;
- cancro do pulmão;
- doenças respiratórias.

Os valores limite referenciados de PM10 estabelecidos são:

Período de Referência	Valor Limite	Margem de Tolerância
1 dia	50 microgramas/m ³	50%
1 ano civil	40 microgramas/m ³	20%

Tabela 4: Tabela PM10

4.1.5 Dióxido de enxofre

O dióxido de enxofre é um gás incolor com um odor picante e proporciona ardor. Ele ocorre como uma impureza nos combustíveis fósseis e resulta da queima desses mesmos combustíveis. Acredita-se que cerca de 80% do dióxido de enxofre tenha como fonte a queima incompleta de combustíveis fósseis nos transportes e na indústria. Na natureza, o gás pode ser liberado para o ar a partir de erupções vulcânicas, atividade rara nos dias de hoje.[14]

Este gás trás consigo consequências negativas para o ser humano, uma vez que é um gás irritante para as mucosas dos olhos e vias respiratórias em concentrações elevadas e pode provocar efeitos agudos e crónicos na saúde humana. A presença simultânea do SO₂ e de outras partículas na atmosfera pode agravar problemas cardiovasculares e respiratórios.



Qualidade do ar consoante o nível de dióxido de enxofre:

Concentração (microgramas/m ³)	Qualidade do ar
0-19	Boa
20-39	Moderado
40-364	Má
365-800	Muito má
800+	Péssima

Tabela 5: Tabela dióxido de enxofre

4.1.6 Ozono

Por fim, o ozono é a última partícula deste *dataset* e é um gás incolor, que possui um odor característico picante, geralmente associado a faíscas elétricas. O odor é geralmente detectável pelo nariz humano a concentrações de 0,02 e 0,05 ppm.[15]

A partir da observação do gráfico da figura 16 é possível constatar duas situações: o mês de Junho não dispõe de dados conclusivos e os valores terão de sofrer tratamento. Relativamente à primeira situação podemos concluir que o sensor que recolheu as informações sobre este tipo de dados deve ter sofrido um qualquer problema e não recolheu os dados para esse mês. Quanto ao segundo ponto, os dados recolhidos de uma pesquisa indicam os valores para a avaliação dos níveis de Ozono em PPM, ou seja, partículas por milhão, gama de valores diferente dos dados recolhidos.

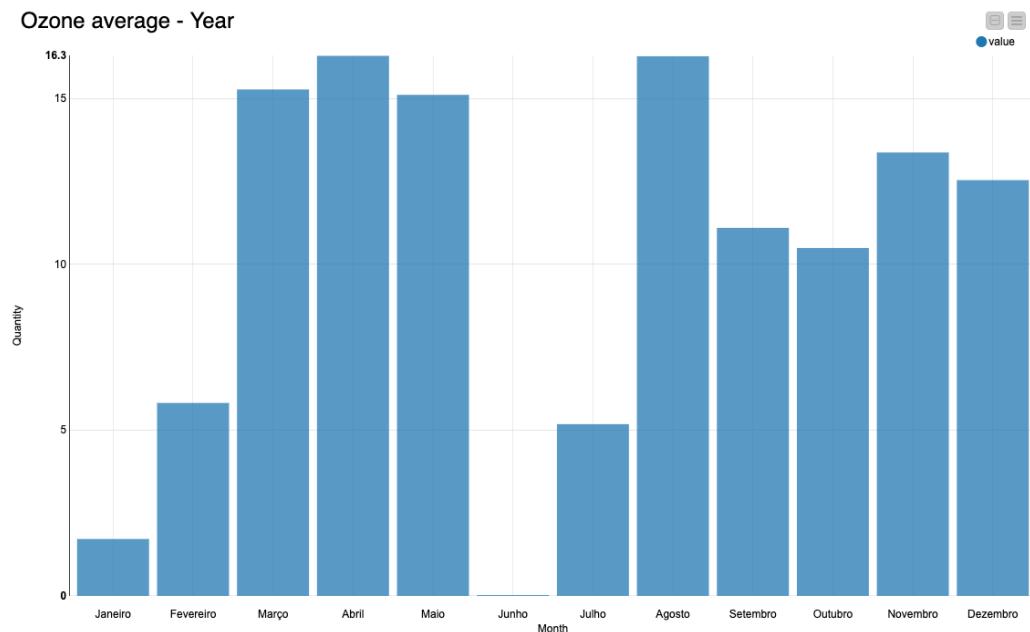


Figura 16: Ozono ao logo do ano

Sabe-se que respirar este gás pode prejudicar a nossa saúde, especialmente em dias quentes e ensolarados, quando o ozono pode atingir níveis prejudiciais, ou seja, níveis mais elevados. Mesmo níveis relativamente baixos podem causar efeitos negativos à saúde. Os valores que se averiguou são limites amplamente aceites na comunidade:

- 8 horas por dia / 5 dias por semana (limite de exposição ocupacional) - 0,1 ppm
- 15 minutos (limite de exposição de curto prazo) - 0,3 ppm

4.2 Dataset Traffic Flow

Relativamente a este segundo *dataset*, pode-se afirmar que é o mais simples de analisar, uma vez que apenas é composto por três colunas, *city_name*, *speed_diff* e *creation_date*. Embora o nome da cidade e a data de criação sejam auto-explicativos, o mais complexo de analisar é cada medição e a data da referida medição, respectivamente.



O *speed_diff* é um parâmetro que se deverá ter em consideração no tratamento de dados devido à informação que fornece face ao trânsito no momento em que foi captado o valor. Este parâmetro é a diferença de velocidades entre a velocidade máxima que os carros podem atingir num cenário sem trânsito e a velocidade que realmente se verifica. Quanto maior for o valor, maior será a diferença entre a velocidade a que os carros circulam no momento e a que velocidade deveriam circular sem trânsito, ou seja, valores altos deste atributo implicam que os veículos estão a circular mais devagar.

Analisando a média da diferença de velocidade ao longo da semana (figura 17), constatámos que os valores mais elevados corresponde aos dias da semana, ou seja, de segunda-feira a sexta-feira em que o pico situa-se na sexta-feira. Ao fim-de-semana os valores são bastante mais baixos que os restantes.

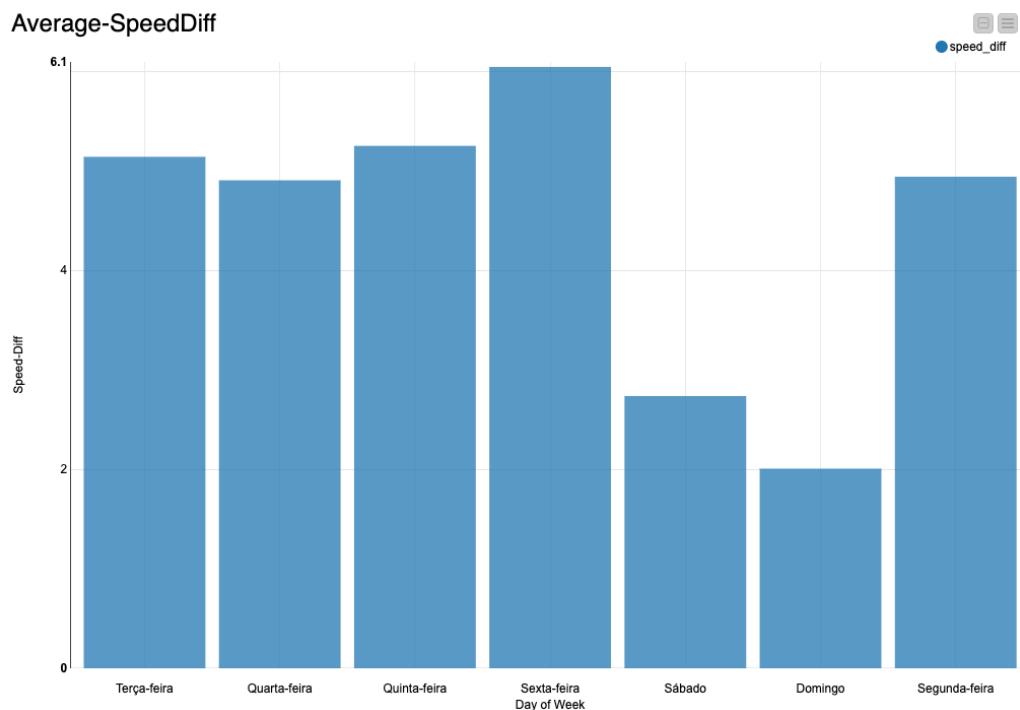


Figura 17: *speed_diff* ao longo da semana

Obtivemos ainda um gráfico com a média do *speed diff* por hora ao longo do ano (figura 18), onde se verificou os valores que estávamos à espera, ou seja, os valores mais elevados corresponde nomeadamente às horas de ponta, isto é, horas de maior afluência de trânsito, mais concretamente entre as 16h e as 19h.

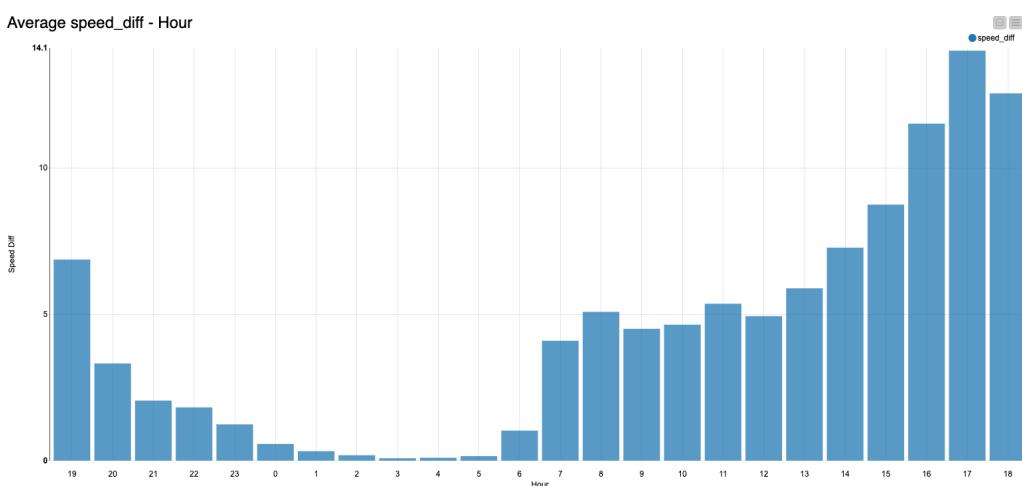


Figura 18: *speed_diff* ao longo da semana



4.3 Dataset Traffic Incidents

No que se refere ao conjunto de dados dos incidentes rodoviários, podemos afirmar que dos quatro *datasets* será o mais complicado de tratar, devido à quantidade de informação e ao seu tipo de informação.

Mais uma vez (tal como os dois *datasets* anteriores), este conjunto de dados contém informações sobre o nome da cidade, número de sequência e a data de criação, tudo dados de fácil tratamento e que possivelmente poderão ser removidos.

Posto isto, será necessário realizar uma análise de cada tipo de dado disponível para se definir as ações a tomar.

description:

Começando pela descrição do incidente, este tipo de dados permite-nos obter uma informação sobre a perigosidade do incidente para os VRUs.

É evidente que as filas de trânsito, as estradas cortadas e os trabalhos nas mesmas influenciam a prática dois dos desportos que se estão a tratar.

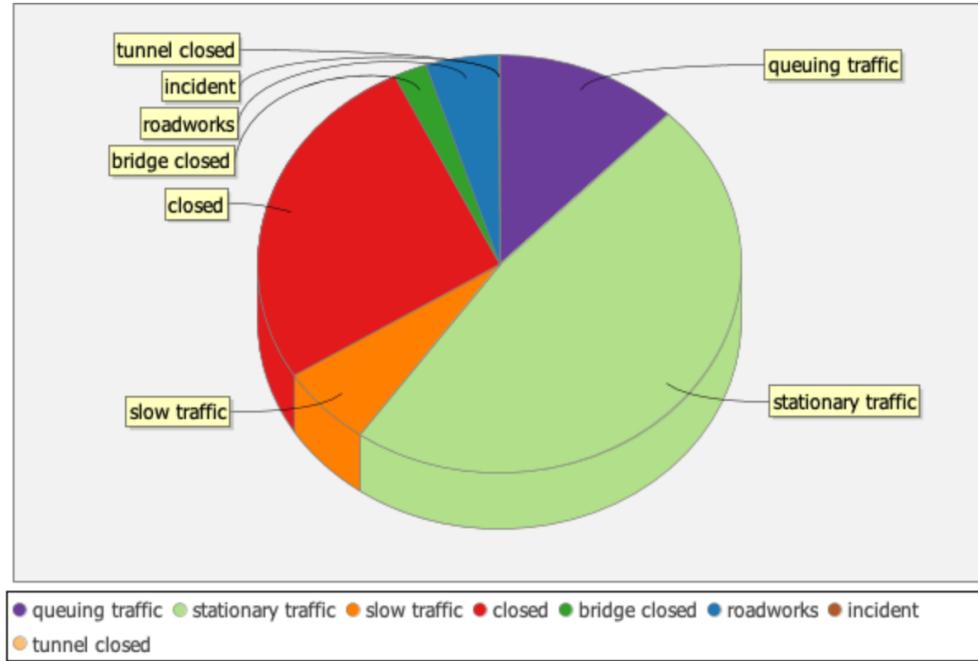


Figura 19: Descrição dos incidentes

from_road:

A informação relativa ao local onde se inicializou o incidente também não será relevante para o objetivo do projecto, tendo em conta que as recomendações serão realizadas de um modo mais geral, isto é, para toda a cidade de Braga.

Desta feita, será um dado a não ter em consideração.

to_road:

Do mesmo modo que não se tomará em consideração o local onde foi inicializado o incidente, também não se tomará em consideração o local onde termina o incidente devido às mesmas razões.

affected_roads:

Devido aos mesmos motivos que nos levam a desconsiderar os dois tipos de dados anteriores, os dados relativos às ruas afetadas pelos incidentes também deverão ser desconsiderados no tratamento de dados.

incident_category_desc:



Quanto à categoria do incidente, podemos afirmar que será um dado a ter em apreciação, visto que nos poderá ajudar a prever o risco que cada incidente tem para os VRUs. É necessário compreender que filas são diferentes de estradas cortas, por exemplo, e que um tipo de categoria pode prejudicar o VRU e o outro não.

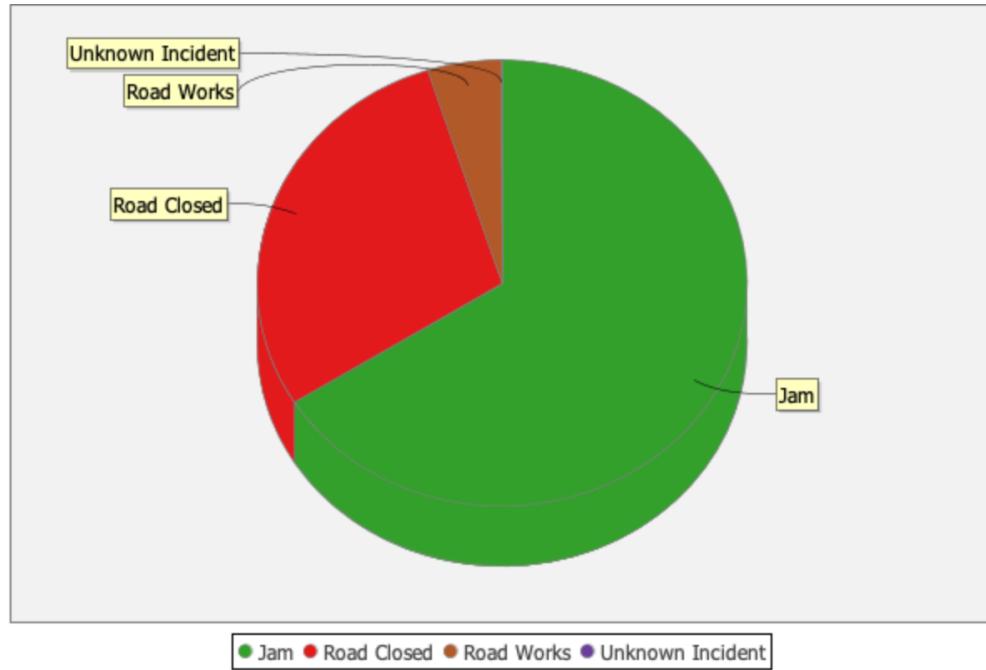


Figura 20: Categoria dos incidentes

magnitude_of_delay_desc:

Em relação à magnitude do incidente é possível retirar informação que nos permitirá perceber a gravidade do incidente. Compreendendo os diferentes níveis de perigosidade, será possível prever o risco que o incidente terá para o VRU, logo é um dado deste *dataset* a utilizar.

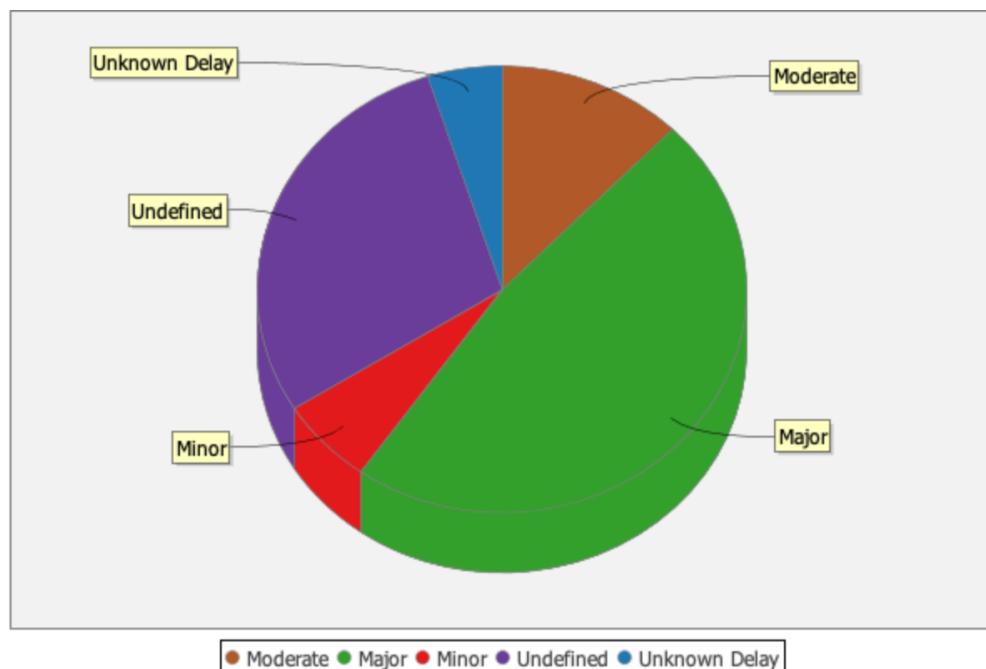


Figura 21: Magnitude dos incidentes



cause_of_incident:

Já as causas do incidente não é um tipo de dados que influencie uma decisão de recomendação para os VRUs, uma vez que apenas explica o porquê de o incidente ter ocorrido, não tendo à partida qualquer peso para a tomada de decisão.

É ainda importante referir que é um fator que não contém grande informação, visto que uma elevada quantidade de ausência de dados.

length_in_m:

Da mesma maneira que as filas são um incidente que influencia os VRUs e os pode prejudicar, o comprimento das mesmas também tem importância e pode agravar o risco que os VRUs correm. Grandes distâncias de trânsito afetam os percursos dos corredores e dos ciclistas, logo quanto maior for a distância pior será para estes praticantes de desporto.

delay_in_s:

Tal como o tipo de dados anterior, o atraso do trânsito também prejudica os VRUs. Quanto maior for o atraso do trânsito, maior será a quantidade de veículos a circular, logo será pior para os corredores e ciclistas.

Assim sendo, através deste dado é possível determinar quando será ou não favorável a prática destas duas atividades.

4.4 Weather

Relativamente a este último *dataset*, encontram-se mais uma vez colunas semelhantes às que aparecem nos *datasets* anteriores, como é o caso da *city_name* e *creation_date*. Mais uma vez o nome da cidade e a data de criação são auto-explicativos, logo a nossa análise será mais focada nas outras variáveis.

Assim sendo, de maneira a começar a análise deste conjunto de dados averiguou-se quais os dados numéricos e quais os dados nominais, descobrindo quais dos dados terão um tratamento mais fácil e precisaram de menor análise, uma vez que os dados numéricos são mais fáceis de tratar face aos nominais como iremos verificar no capítulo 5.

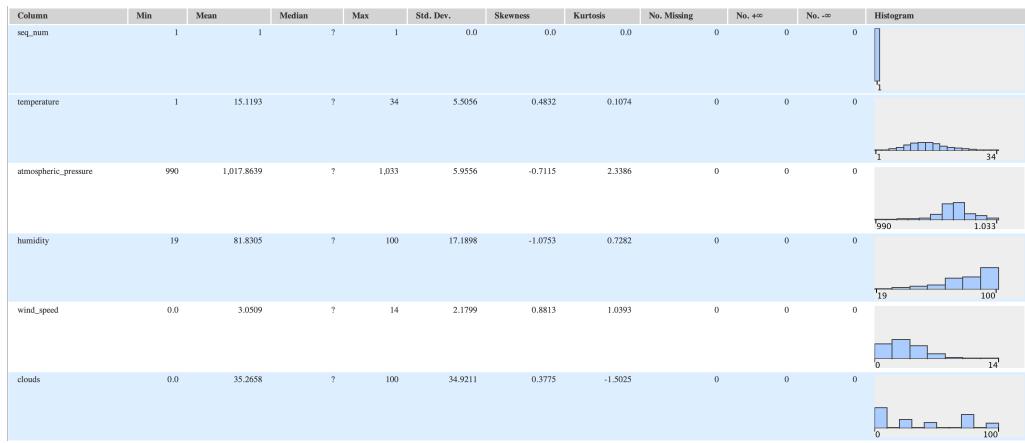


Figura 22: Dados numéricos *dataset Weather*

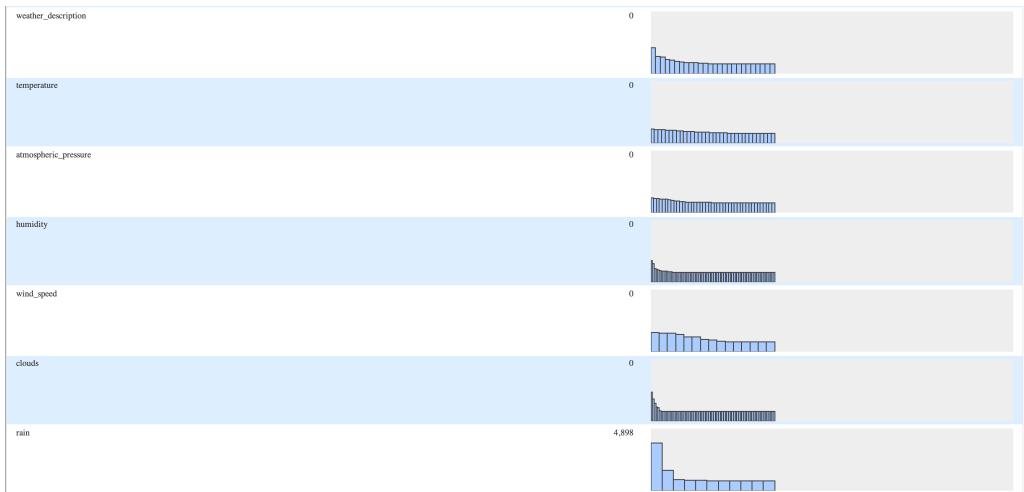


Figura 23: Dados nominais *dataset Weather*

Através da apreciação feita às figuras 22 e 23, podemos retirar algumas conclusões:

- A maioria dos dados numéricos são fatores importantes na avaliação a fornecer ao VRU;
- Os dados nominais irão necessitar de tratamento;
- Alguns dados nominais serão removidos, não têm grande interferência numa futura avaliação a fornecer ao VRU.

Desta feita, iremos analisar os dados que existem no conjunto um a um para que seja possível perceber quais são possíveis remover, alterar ou manter.

weather_description:

Quanto a este dado é importante referir que várias das suas variáveis (descrições do tempo) são semelhantes como são os casos de "névoa" e "neblina", "céu pouco nublado" e "algumas nuvens" e o caso "chuvisco e chuva fraca" e "chuvisco fraco".

Logicamente que o estado do tempo influencia os VRUs na prática de exercício de exercício físico, visto que a existência de nevoeiro e chuva, por exemplo, é pior que um dia que esteja limpo e com sol.

temperature:

Em relação à temperatura, sabemos quanto maior for a temperatura maior será o desgaste corporal e maior será o cansaço e a desidratação. Da mesma forma que as temperaturas altas afetam, temperaturas muito baixas podem provocar mais facilmente lesões e existe, na mesma, um grande desgaste por parte do VRU.

atmospheric_pressure:

A pressão atmosférica está proporcionalmente relacionada com a altitude, uma vez que à medida que a altitude aumenta, a pressão atmosférica diminui.

Com uma diminuição da pressão atmosférica (ou valores mais baixos que o normal), menos moléculas de oxigénio estão disponíveis no ar.[17] Consequentemente diminuição do oxigénio disponível afeta o corpo de muitas maneiras:

- a frequência e a profundidade da respiração aumentam;
- alteração do equilíbrio entre gases nos pulmões e no sangue;
- aumento da alcalinidade do sangue;
- alteração da distribuição de sais nas células.

humidity:

A humidade ao longo do ano varia bastante e Braga é uma zona de Portugal onde os seus valores sofrem bastantes alterações. Segundo a Organização Mundial de Saúde (OMS), o nível ideal de humidade do ar para o organismo do ser humano varia entre 40%



e 70%. Quando a taxa diminui para 30% é considerada uma situação de alerta e prejuízos para a saúde tornam-se mais evidentes.

Alguns dos sintomas que os valores muito mais de humidade provocam são a garganta irritada, sangramento pelo nariz e o aparecimento de doenças respiratórias (exemplos, rinite e asma), além de dor de cabeça, sensação de areia nos olhos e pele seca.[18]

wind_speed:

Quando é velocidade do vento, é de fácil compreensão o facto de que este fator terá mais influência nos ciclistas. Ventos fortes pode levar o ciclista a perder o equilíbrio e consequentemente ter uma queda que lhe pode provocar lesões. À partida, será um fator importante a ter em consideração no tratamento de dados.

clouds:

As nuvens não serão um fator importante para os VRUs, uma vez que não têm influência direta na prática do exercício físico, ou seja, não condiciona nenhuma das duas práticas.

rain:

No que diz respeito à chuva, é de senso comum que influencia a prática de desporto. Avaliando a situação de um corredor num dia de chuva, só existem fatores negativos, pois este pode sofrer uma queda pelo piso estar molhado e pode ainda ficar doente depois de realizar a sua corrida.

Já os ciclistas pode também sofrer dos mesmos problemas dos condutores, sendo até pior, pois muitas vezes estes utilizam vias onde circulam veículos de combustão, como carros e motas, e pode dar-se a situação de um acidente grave.

current_luminosity:

Quanto à luminosidade, não terá influência direta nos VRUs, visto que caso não existir iluminação solar, existirá iluminação do local onde o VRU se encontrar, não prejudicando assim a prática de desporto.

sunrise:

O nascer do sol assemelha-se ao fator da luminosidade, não tendo um contributo benéfico nem prejudicial para os VRUs.

sunset:

Por último, o pôr do sol é um fator igual ao nascer do sol.



5 Tratamento dos dados

Este capítulo tem como objetivo explicar o processo detalhado do tratamento de dados realizado com base no que foi mencionado no capítulo anterior, isto é, com base na análise dos dados elaborada.

Para tal, mais uma vez foi utilizado a ferramenta KNIME para realizar esse tratamento e nas secções seguintes será detalhado todo o tratamento efetuado para cada conjunto de dados.

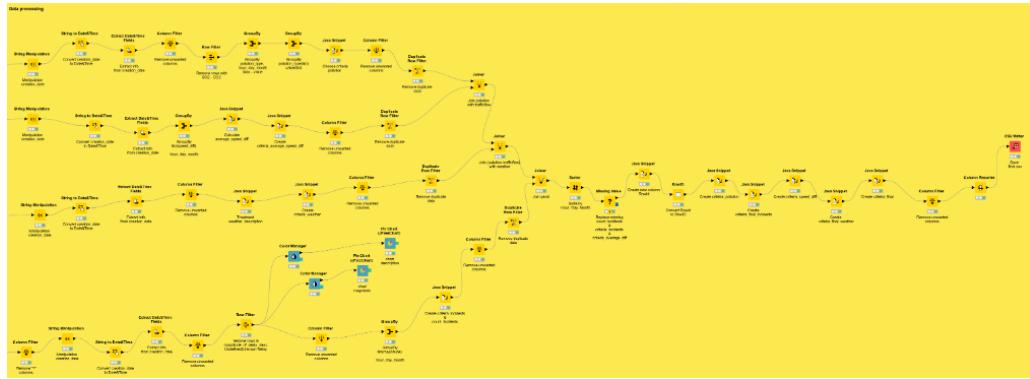


Figura 24: Workflow tratamento de dados

5.1 Dataset Pollution

Começando pelo conjunto de dados da Poluição, numa primeira fase foi realizado o tratamento da *creation_date*. Iniciou-se o processo através de um *String Manipulation* que nos permitiu remover o milissegundos da hora com a expressão regular:

```
regexReplace($creation_date$, ".000000", "")
```

Retirados os milissegundos, aplicou-se um *String to Date&Time* para converter a *string* no tipo de *Date&Time*, onde através de um *Extract DateTime Fields* foi possível extrair as informações que achamos necessárias, sendo elas:

- Mês;
- Dia da semana;
- Hora do dia;

Posto isto, decidiu-se remover todas as colunas que não seriam utilizadas e com um *Column Filter* removeu-se as seguintes colunas:

- *city_name*;
- *seq_num*;
- *last_updated*;
- *creation_date*;

Estas colunas foram removidas, visto que a cidade é sempre a mesma assim como o *seq_num*, a data da última atualização é informação que não tem interesse e a data de criação já foi tratada na fase anterior, de maneira que não é necessária novamente.

Assim sendo, com base no capítulo anterior, com a aplicação de um *Row Filter* optou-se por remover tanto o SO2 como o CO2, uma vez que os valores que existem destes dois tipos de poluição não têm qualquer tipo de influencia para os seres humanos, pois são valores negativos.

Removidos todos os dados desnecessários foi fundamental agrupar os dados que restavam, para tal colocou-se um *GroupBy* para que fosse possível agrupar o tipo de poluição, o mês, o dia e a hora pelo máximo do valor de cada poluição, uma vez que para cada hora de cada dia do mês cada tipo de poluição continha mais do que um valor, ou seja, foi necessário ficar com o máximo desses valores.



De seguida, através de um *GroupBy* foi necessário criar uma lista com os tipos de poluição e uma lista com os valores máximos dos mesmos para cada hora de maneira a que fosse possível elaborar critérios de risco. Utilizou-se um *Java Snippet* para que, consoante os valores de perigo de cada poluição analisados no capítulo anterior, fosse colocada uma *label* segundo a perigosidade. As *labels* utilizadas foram:

- risco_muito_baixo;
- risco_baixo;
- risco_moderado;
- risco_alto;
- risco_muito_alto;

Apenas no tipo de poluição *Particulate Matter 10* é que as *labels* apenas foram risco_baixo e risco_alto, devido aos valores que foram investigados no capítulo anterior.

Criadas novas colunas para cada tipo de poluição com os diferentes riscos, foi preciso remover as duas colunas que continham as listas de tipos de poluição e valores por hora por intermédio de um *Column Filter* (figura 25).

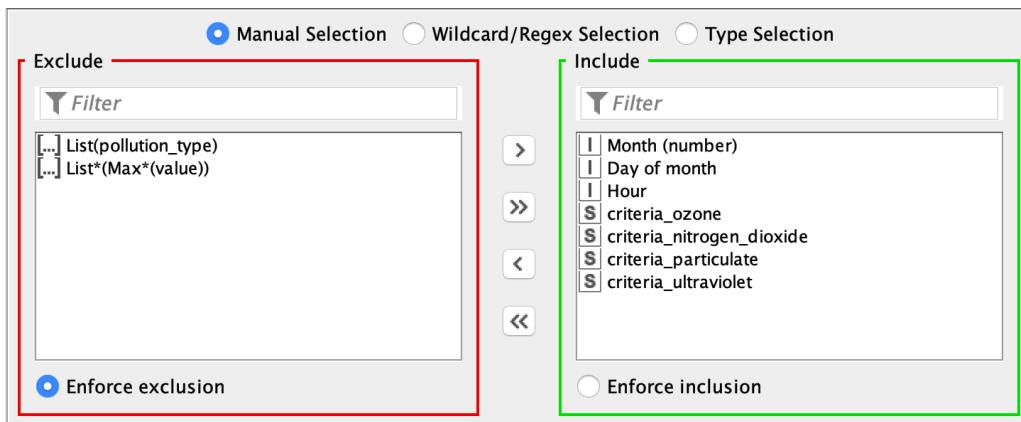


Figura 25: *Column Filter Pollution*

Para terminar foi utilizado um *Duplicate Row Filter* para remover a informação duplicada que o conjunto de dados contenha. Na figura 26 é possível visualizar a sequência retratada no KNIME.

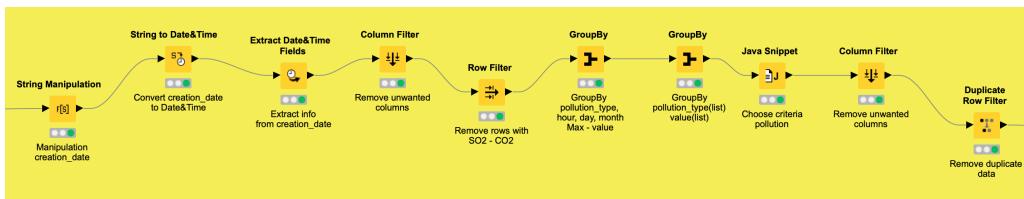


Figura 26: Tratamento do *dataset Pollution*

5.2 Dataset Traffic Flow

Passando para o conjunto de dados relativos ao fluxo de trânsito, foi realizado o mesmo processo para o tratamento da data de criação, utilizando um *String Manipulation*, seguido de um *String to Date&Time* e acabando com um *Extract Date&Time Fields*.

Por conseguinte, aplicou-se um *GroupBy* para que fosse possível agrupar uma lista com os valores do *speed_diff* por mês, dia do mês e hora e remover as colunas com o nome da cidade (mais uma vez é sempre a mesma cidade, Braga) e a data de criação, uma vez que já foi tratada.



Dado que existiam vários valores de *speed_diff* para cada hora em cada lista, optou-se por calcular a média dos valores presentes em cada lista e ficar com esse valor, como tal foi aplicado um *Java Snippet* que indo a cada hora, somava os valores da lista e dividi-a pela quantidade de valores, obtendo assim as diferentes médias.

Tendo agora um valor de *speed_diff* para cada hora, decidiu-se definir valores de risco para diferentes intervalos deste dado com o auxílio de um *Java Snippet*, isto é, definiram-se as seguintes *labels* para os seguintes intervalos:

- risco_muito_baixo: para valores menores que 5;
- risco_baixo: para valores entre 5 e 10;
- risco_moderado: para valores entre 10 e 15;
- risco_alto: para valores entre 15 e 20;
- risco_muito_alto: para valores maiores que 20;

Criada uma coluna com os diferentes riscos para o *speed_diff*, é possível remover-se tanto a coluna *average_speed_diff* como a coluna com a lista dos diferentes valores desta propriedade que foi criada anteriormente. Empregou-se um *Column Filter* para esse efeito (figura 27).

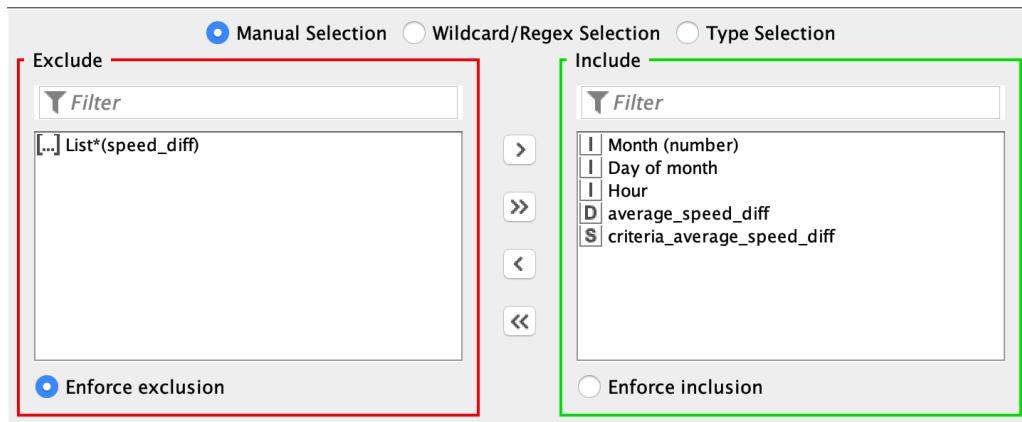


Figura 27: Column Filter Traffic Flow

Por fim, aplicou-se um *Duplicate Row Filter* para remover a informação duplicada que exista neste conjunto de dados. Na figura seguinte é possível visualizar a sequência retratada no KNIME.

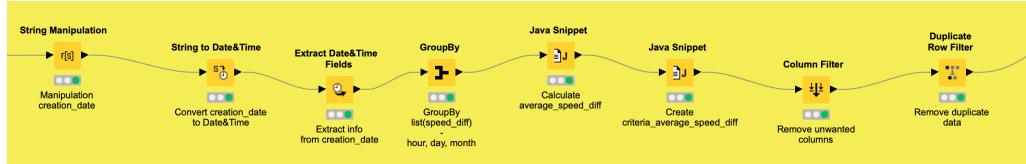


Figura 28: Tratamento do dataset Traffic Flow

5.3 Dataset Traffic Incidents

Relativamente ao *dataset* dos incidentes de trânsito, colocou-se um *Column Filter* e começou-se por remover as colunas do *dataset* que apenas continham "?" como é o caso da coluna *cause_of_incident* e também foram removidas as colunas que não se iriam utilizar, ou seja, removeram-se as colunas *city_name*, *seq_num*, *from_road*, *to_road* e *affected_roads*.

Realizou-se de seguida o tratamento da data tal como foi executado nos conjuntos de dados anteriores, obtendo-se três novas colunas: mês, dia do mês e hora.



Posteriormente, o conjunto de dados foi revisto e achou-se que continuavam a existir dados que não seriam necessário, e através de um *Column Filter* removeu-se as colunas *creation_date*, *incident_category_desc*, *length_in_m* e *delay_in_s*.

Nesta fase, iniciou-se o processo de tratamento da *magnitude_of_delay_desc* por intermédio de um *Row Filter* onde se removeu todas as linhas que continham o valor *Unknown Delay* e *Undefined*, uma vez que ambos referem-se ao mesmo e ambos não têm interferência para os dados, pois não têm informação sobre a magnitude.

Posto isto, colocou-se dois *Pie Charts* para verificar os novos valores das colunas *description* e *magnitude_of_delay_desc* tal como se pode verificar nas figuras seguintes.

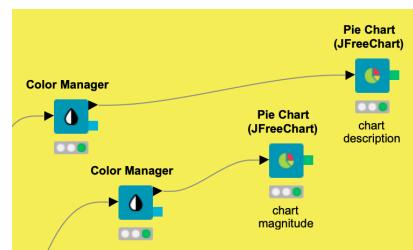


Figura 29: Análise das colunas *description* e *magnitude_of_delay_desc*

Como se pode verificar ficou-se apenas com três valores distintos na coluna: *slow traffic*, *queuing traffic* e *stationary traffic*.

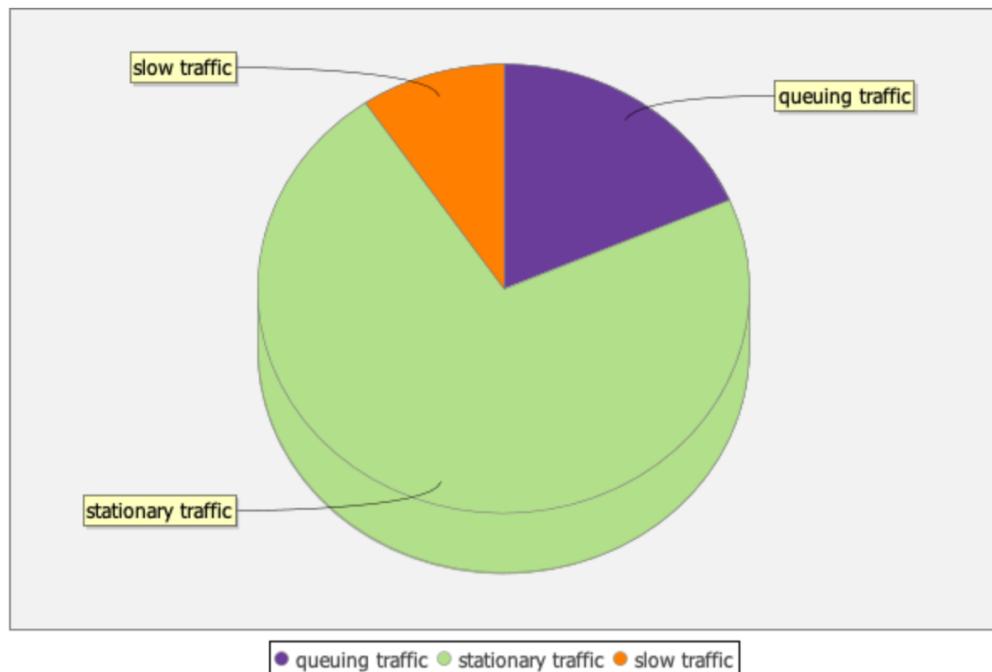


Figura 30: Valores de *description*

Da mesma forma, a coluna relativa à magnitude também ficou apenas com três dados diferentes: *minor*, *moderate* e *major*.

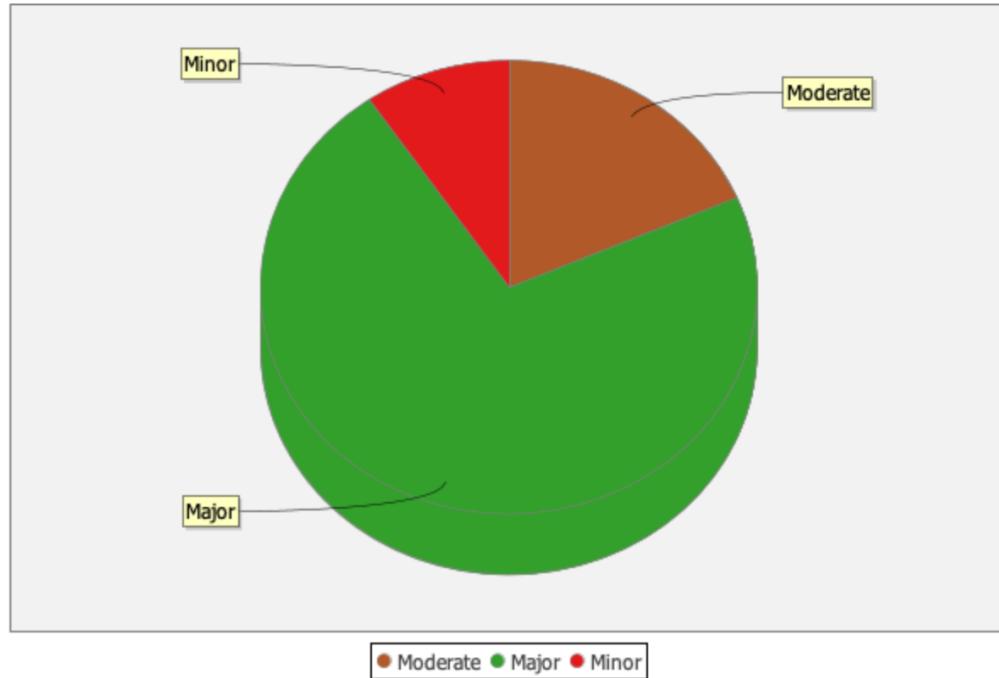


Figura 31: Valores de *magnitude_of_delay_desc*

Efetuada esta análise, optou por ficar apenas com a coluna *magnitude_of_delay_desc*, uma vez que indicava o mesmo que a *description*, logo removeu-se esta última com o auxílio de um *Column Filter* (figura 32).

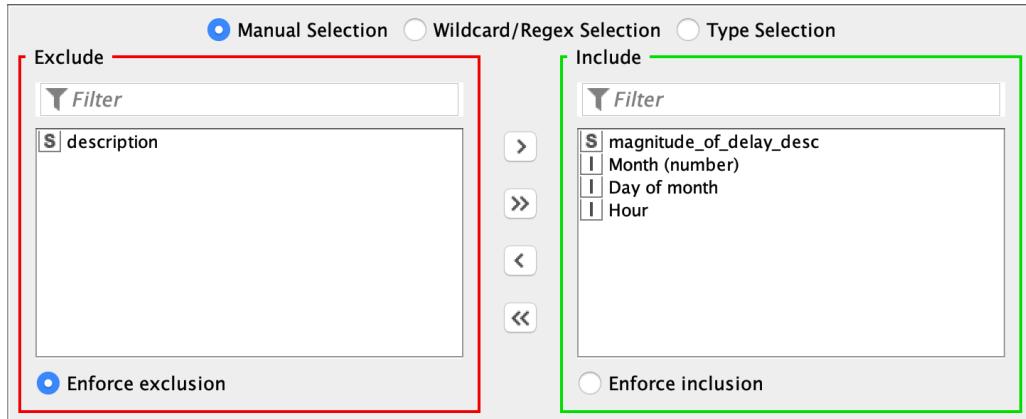


Figura 32: *Column Filter Traffic Incidents*

Neste momento, continuou-se o tratamento destes dados e com um *GroupBy* agrupou-se todos os valores de magnitude por cada hora em listas para que através de um *Java Snippet* atribuiu-se pesos a cada tipo de magnitude, mais concretamente um peso de 5 unidades para *minor*, 10 unidades para *moderate* e 15 unidades para *major*.

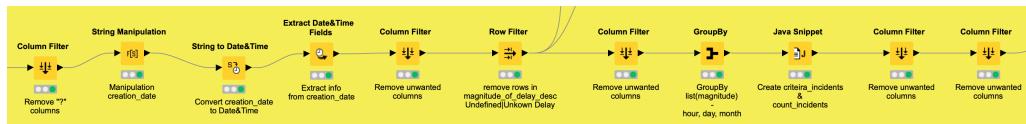


Figura 33: Tratamento do dataset *Traffic Incidents*



5.4 Dataset Weather

Por fim, temos o tratamento do conjunto de dados do Tempo. Mais uma vez foi necessário tratar da data e o processo é igual ao realizado nos *datasets* anteriores.

Realizado o tratamento da data, foi necessário remover, através de um *Column Filter*, as colunas que eram prescindíveis como é o caso da *creation_date*, *city_name*, *seq_num*, *clouds*, *rain*, *current_luminosity*, *sunrise* e *sunset*. A escolha de remover estas colunas foi baseada na análise de dados realizada anteriormente e pela percepção de que existiam colunas como informação repetida, como é o caso da *weather_description* e da *rain*, visto que em caso de chuva tinham as duas o mesmo.

Em seguida, empregou-se um *Java Snippet* com o intuito de realizar um tratamento à coluna *weather_description*, uma vez que esta disponha de 26 valores diferentes com vários deles a conterem imensas semelhanças entre si.

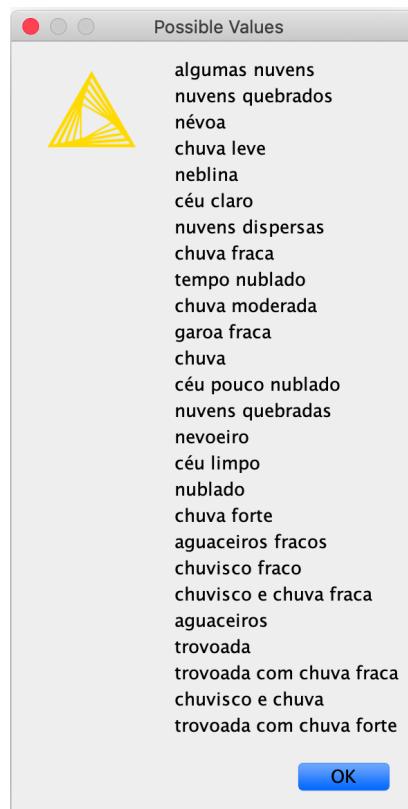


Figura 34: 26 valores do *weather_description*

Assim sendo, ao invés de existirem esses 26 valores diferentes, optou-se por agrupar estes valores em 8 valores diferentes:

- limpo;
- nublado;
- nevoeiro;
- chuva fraca;
- chuva;
- chuva forte;
- trovoada;
- trovoada com chuva;

Feita esta distribuição, através de um *Java Snippet* é possível realizar mais facilmente uma distribuição dos riscos de perigosidade como se fez com outras variáveis. Mais uma vez utilizou-se os 5 riscos diferentes consoantes os 8 diferentes valores:

- risco_muito_baixo: para os valores "limpo" e "nublado";



- risco_baixo: para os valores "nevoeiro" e "chuva fraca";
- risco_moderado: para o valor "chuva";
- risco_alto: para os valores "chuva forte" e "trovoada";
- risco_muito_alto: para o valor "trovoada com chuva";

Por último, foi necessário remover as colunas desnecessárias como a *weather_description*, a *atmospheric_pressure* e a *wind_speed*, por intermédio de um *Column Filter* (figura 35).

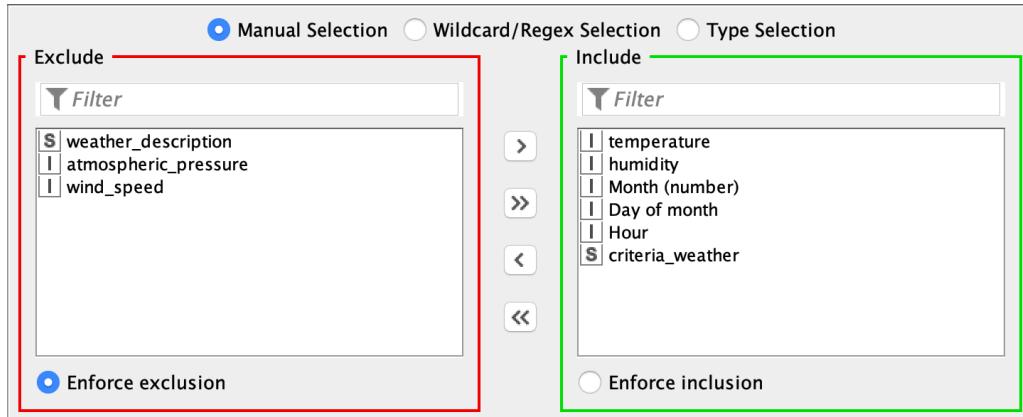


Figura 35: *Column Filter Weather*

Também foi necessário remover todos os dados duplicados através de um *Duplicate Row Filter*. Na figura 36 é possível visualizar a sequência retratada no KNIME.

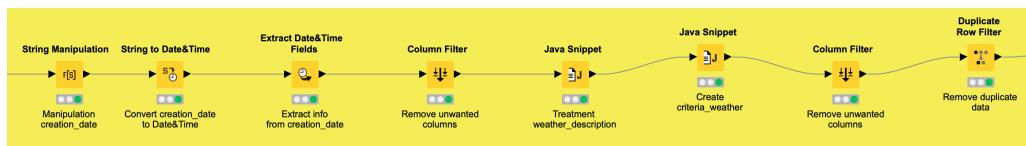


Figura 36: Tratamento do *dataset Weather*

5.5 Agregação dos dados

Após realizado o tratamento anterior relativo aos quatro conjuntos de dados, foi necessário agregá-los num *dataset* só. Para tal esta agregação foi realizar através de três *Joiners*, onde o primeiro agrupa o *dataset* da poluição com o *traffic flow*, já o segundo *Joiner* agrupa este novo *dataset* com o conjunto de dados do tempo e, por último, este novo *dataset* agrupa-se ao conjunto de dados dos incidentes através do último *Joiner*.

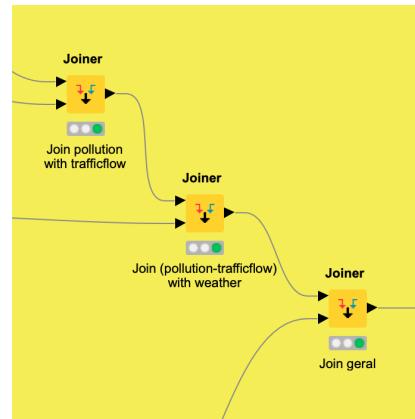


Figura 37: Agregação dos dados



5.6 Tratamento dados em falta

De seguida, feita a agregação de todos os dados num *dataset*, foi possível constatar que existiam alguns dados sem informar, ou seja, existiam alguns *missing values*.

Para resolver este problema, optou-se por inicialmente ordenar todos os dados por pela hora, dia e mês através de um *Sorter*. Realizada esta ordenação, foi possível, com o auxílio de um *Missing Value*, substituir todos estes valores em falta pelo valor anterior, isto é, todos estes valores foram substituídos pelo valor da hora anterior. Esta decisão foi tomada, uma vez que nos pareceu sensato utilizar os valores da hora anterior, pois na maioria dos casos a variação não é tão acentuada.

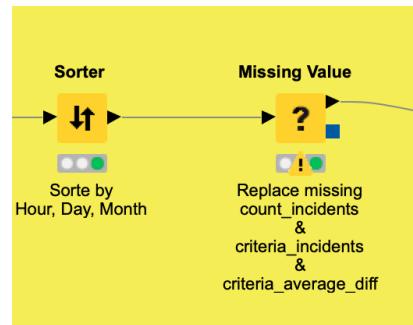


Figura 38: Tratamento dados em falta

5.7 Criação do critério final

Por último, foi necessário definir um critério final para que seja usado como termo de comparação para os resultados que se irão obter da rede neuronal a construir a seguir.

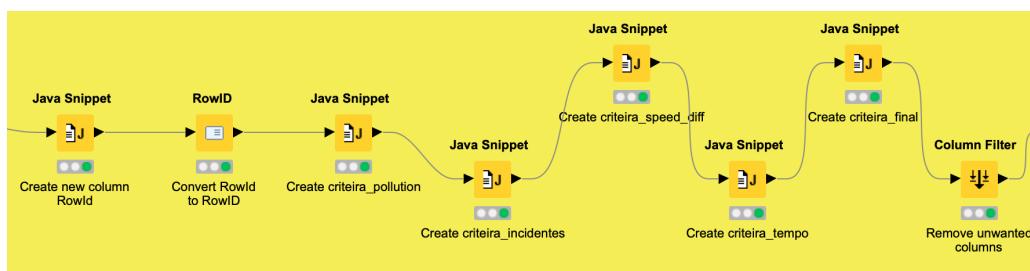


Figura 39: Criação do critério final

Inicialmente foi necessário definir uma coluna com o *Row ID* para cada linha através de um *Java Snippet* e de um *RowID*.

Feito isto, foi necessário tratar de quatro grupos de informação:

- **Poluição;**
- **Incidentes;**
- **Speed_diff;**
- **Tempo;**

Cada grupo de informação foi transformado num critério só através de um *Java Snippet* e através destes novos quatro critérios e com a utilização de mais um *Java Snippet* foi definido um critério final. Criado este critério final todos os outros quatro critérios foram removidos com um *Column Filter*.



5.7.1 Critério da Poluição

De maneira a tratar da Poluição, há que ter noção que existem quatro tipos de informação: a PM10, o dióxido de nitrogénio, a ultravioleta e o ozono. Para cada um destes tipos de informação foi necessário inserir alguns valores numéricos para cada tipo de risco dos diferentes dados, aos quais denominamos de "pesos", para que fosse possível realizar o cálculo da métrica deste critério.

Tipo de dados	Tipo de risco ou intervalo	Peso
criteria_particulate	risco_baixo	2
	risco_alto	4
criteria_ozone	risco_muito_baixo	1
	risco_baixo	2
	risco_moderado	3
	risco_alto	4
	risco_muito_alto	5
criteria_nitrogen_dioxide	risco_muito_baixo	1
	risco_baixo	2
	risco_moderado	3
	risco_alto	4
criteria_ultraviolet	risco_muito_baixo	1
	risco_baixo	2
	risco_moderado	3
	risco_alto	4

Tabela 6: Tabela com pesos da Poluição

Neste caso a métrica definida para o cálculo do critério da Poluição foi atribuir uma percentagem de importância do valor total a cada tipo de dados, o criteria_particulate, o criteria_nitrogen_dioxide, criteria_ultraviolet, o criteria_ozone onde a importância de cada um é 35%, 30%, 25% e 10%, respetivamente. De seguida, soma-se todos os valores das diferentes multiplicações e verifica-se entre que valor estava esta soma, atribuindo um valor final de 1 a 5 consoante o intervalo em que se encontre a soma. Na tabela 7 teremos os diferentes intervalos bem como as diferentes atribuições.

Intervalo	Atribuição (valor)
0 a 1.5	1
1.5 a 2.5	2
2.5 a 3.5	3
3.5 a 4.5	4
>= 4.5	5

Tabela 7: Tabela com atribuições da Poluição

5.7.2 Critério da Incidentes

Relativamente aos Incidentes o processo é semelhante, apesar de que neste caso apenas foi necessário tratar de um tipo de dados, mais concretamente o *criteria_incidents* e atribuir-lhe os pesos consoante o tipo de risco.

Tipo de dados	Tipo de risco ou intervalo	Peso
criteria_incidents	risco_muito_baixo	2
	risco_baixo	4
	risco_moderado	7
	risco_alto	12

Tabela 8: Tabela com pesos do Incidentes

Neste caso, o segundo tipo de dados que é a quantidade de incidentes foi utilizado para ajudar a calcular os valores a atribuir no cálculo da métrica. Isto é, para efetuar o cálculo da métrica neste critério, optou-se por multiplicar o peso de cada linha relativo ao *criteria_incidents* pelo número de incidentes dessa linha e, consoante a gama de valores em que este resultado estivesse, seria atribuído um valor de 1 a 5.



Intervalo	Atribuição (valor)
0 a 10	1
10 a 30	2
30 a 50	3
50 a 200	4
≥ 200	5

Tabela 9: Tabela com atribuições dos Incidentes

5.7.3 Critério da Speed_diff

No caso do critério da *Speed_diff*, como apenas existia um tipo de dados, decidiu-se simplesmente converter as *strings* dos riscos, que já estavam definidos nesse tipo de dados, para valores inteiros.

- risco_muito_baixo = 1;
- risco_baixo = 2;
- risco_moderado = 3;
- risco_alto = 4;
- risco_muito_alto = 5;

5.7.4 Critério do Tempo

Para o critério do tempo, o processo foi semelhante aos anteriores, embora neste caso para além de existirem dados em que têm definidos o tipo de risco, também existem dados apenas com diferentes valores numéricos. Nesses casos, foi necessário tratar estes valores com base no que foi investigado no capítulo 4.

Tipo de dados	Tipo de risco ou intervalo	Peso
criteria_weather	risco_muito_baixo	1
	risco_baixo	2
	risco_moderado	3
	risco_alto	4
	risco_muito_alto	5
temperature	<5	5
	5 a 10	3
	10 a 15	1
	15 a 25	3
	≥ 25	5
humidity	<20	5
	20 a 30	4
	30 a 40	3
	40 a 45	2
	45 a 65	1
	65 a 70	2
	70 a 80	3
	80 a 90	4
	≥ 90	5

Tabela 10: Tabela com pesos do Tempo

Para o cálculo da métrica para o Tempo, optou-se por começar por atribuir uma percentagem do valor total consoante a importância de cada tipo de dado, ou seja, existem três tipos de dados, o *criteria_weather*, a temperatura e a humidade, e a importância de cada um para este critério é de 50%, 25% e 25%, respetivamente.

$$\text{criteria_tempo} = \text{criteria_weather} * 0.5 + \text{temperatura} * 0.25 + \text{humidade} * 0.25$$

Isto é, como os três tipos de dados contém valores de 1 a 5, a soma dos valores de cada tipo multiplicado pela percentagem respetiva irá resultar num valor no intervalo de 1 a 5.



Intervalo	Atribuição (valor)
0 a 1.5	1
1.5 a 2.5	2
2.5 a 3.5	3
3.5 a 4.5	4
4.5 a 5	5

Tabela 11: Tabela com atribuições do Tempo

5.7.5 Critério Final

Cálculos estes últimos quatro critérios, é possível efetuar o cálculo do critério final a utilizar como termo de comparação para os resultados que se irão obter da rede neuronal a construir a seguir.

A métrica matemática a utilizar é simples e semelhante a métricas anteriores, tanto o critério da poluição como dos incidentes valerão 35% e tanto o critério para o *speed-diff* como o critério para o tempo valerão 15%, e como todos os valores de cada critério variam entre 1 e 5, o resultado da soma do valor de cada critério varia entre 1 e 5.

$$\text{criteria_final} = \text{pollution} * 0.35 + \text{incidentes} * 0.35 + \text{speed_diff} * 0.15 + \text{tempo} * 0.15$$

Intervalo	Atribuição (valor)
0 a 1.5	1
1.5 a 2.5	2
2.5 a 3.5	3
3.5 a 4.5	4
4.5 a 5	5

Tabela 12: Tabela com atribuições do Critério Final

5.8 Dataset final

Finalizado o tratamento de dados, foi gerado um *dataset* final com todos as *features* que iremos utilizar na fase seguinte e com um total de 6820 linhas, ou seja, 6820 dados com todas as *features*. Na seguinte lista iremos apresentar o conteúdo que compõe este *dataset*.

- **Row ID** : composto pela data;
- **Hour** : composto pela hora;
- **Day of month** : composto pelo dia do mês;
- **Month (number)** : composto pelo número do mês;
- **criteria_ozone** : composto pelo critério definido para o nível do ozono na hora correspondente;
- **criteria_nitrogen_dioxide** : composto pelo critério definido para o nível do dióxido de nitrogénio na hora correspondente;
- **criteria_particulate** : composto pelo critério definido para o nível do PM10 na hora correspondente;
- **criteria_ultraviolet** : composto pelo critério definido para o nível de ultravioleta na hora correspondente;
- **temperature** : composto pelo critério definido para a temperatura na hora correspondente;
- **criteria_weather** : composto pelo critério definido para tempo na hora correspondente;
- **count_incidents** : composto pelo número de incidentes na hora correspondente;
- **criteria_incidents** : composto pelo critério definido para o nível dos incidentes na hora correspondente;



- ***criteria_pollution*** : composto pelo critério definido para o nível final da poluição na hora correspondente;
- ***criteria_final_incidents*** : composto pelo critério definido para o nível final dos incidentes na hora correspondente;
- ***criteria_speed_diff*** : composto pelo critério definido para o nível final da *speed diff* na hora correspondente;
- ***criteria_final_weather*** : composto pelo critério definido para o nível final do tempo na hora correspondente;

Neste *dataset* também está guardado o *target* a utilizar no capítulo seguinte, que é o ***criteria_final*** composto pelo critério final geral na hora correspondente. É importante relembrar que este *target* é o resultado de uma análise complexa de um conjunto de dados que deram origem a este critério final tal como foi detalhado anteriormente.



6 Desenvolvimento e treino

Concluído o tratamento de dados, é necessário construir o modelo da rede neuronal. O grupo definiu duas alternativas de abordar o problema: a utilização de uma *Random Forest* ou a criação de uma *Multi Layer Perceptron* (MLP). No entanto, após testar tanto a *Random Forest* como a MLP, o grupo optou por utilizar a MLP, tal como se vai explicar neste capítulo.

6.1 Random Forest

Começando pela *Random Forest*, o seu processo foi realizado no KNIME e é constituído pelo *workflow* da figura 40.

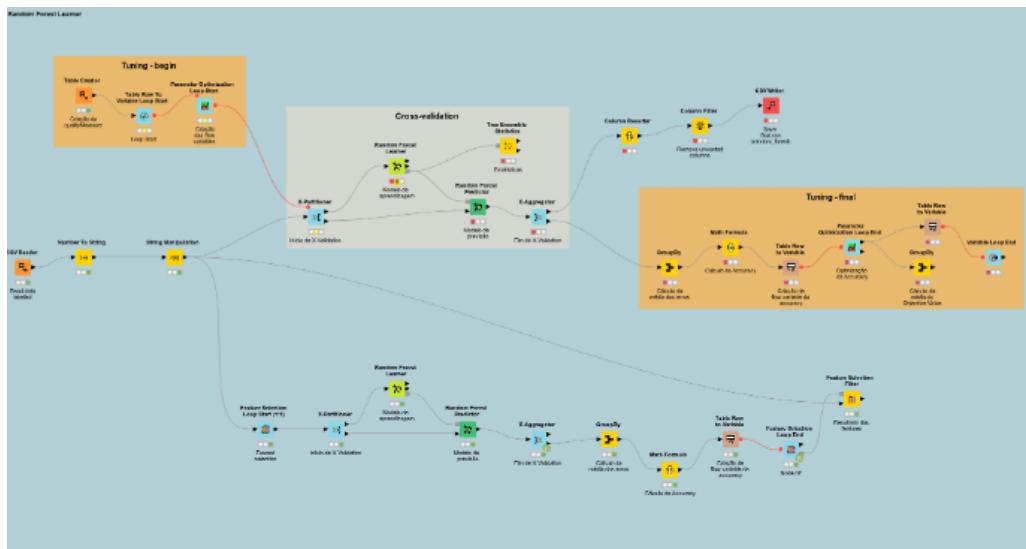


Figura 40: Workflow Random Forest

A *Random Forest* elaborada foi aplicada com *Cross-Validation*. Nesta técnica foi aplicado um *tunning* inicial de modo a fornecer os parâmetros otimizados para a *Cross-Validation* e, posteriormente, o conteúdo do processamento obtido na *Cross-Validation* será fornecido a um *tunning* final, de modo a que seja possível calcular a *accuracy* do modelo construído.

6.1.1 Tuning inicial

Relativamente ao *Tunning* inicial, foi realizado o *tunning* de parâmetros nominais e de parâmetros numéricos, respetivamente.

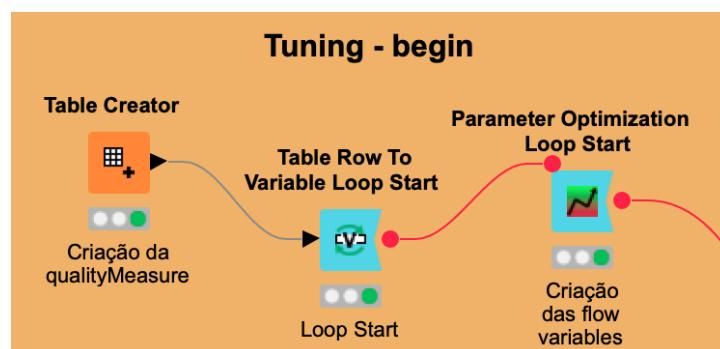


Figura 41: Parte inicial do tuning

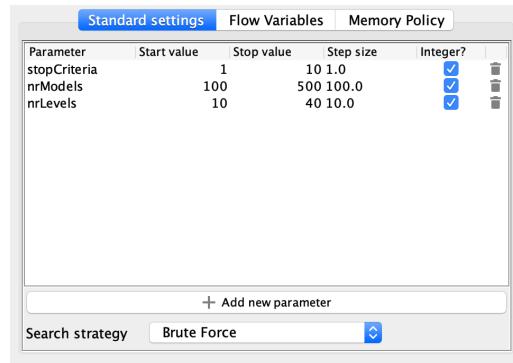
Inicialmente definiu-se os parâmetros nominais da tabela a ser criada, através do *Table Creator*, com o intuito de otimizar a variável *Quality Measure*. Para tal definimos os parâmetros *Information Gain*, *Information Gain Ratio* e, por último, o *Gini Index*.



	<input type="checkbox"/>	qualityMeasure
Row0		InformationGain
Row1		InformationGainRatio
Row2		Gini

 Figura 42: Definições *Table Creator*

Após a introdução dos parâmetros nominais, aplicou-se os parâmetros numéricos. Através do *Parameter Optimization Loop Start* criou-se três variáveis diferentes: *nrModels* que corresponde ao número máximo de modelos a usar, *nrLevels* que corresponde ao número máximos de níveis de cada árvore de decisão e o *stopCriteria* que corresponde a uma variável que serve de critério de paragem. A estratégia utilizada foi a *Brute Force*.


 Figura 43: Definições *Parameter Optimization Loop Start*

Feito o *Tunning* inicial, os parâmetros definidos são fornecidos um nodo chamado de *X-Partitioner* no qual se inicia a *Cross-Validation*.

6.1.2 Cross-Validation

Mas o que é realmente a *Cross-Validation*? *Cross-validation* é uma técnica de validação de modelos de *Machine Learning* que tem como objetivo ter uma métrica precisa do desempenho do modelo na prática.

Essencialmente, consiste em dividir o conjunto de dados em *k-folds*. Em cada execução do modelo, *k-1* dobras são usadas para treino e 1 dobra (a restante) é usado como teste. Continua-se a repetir o processo até que todas as dobras tenham sido usadas para teste. A métrica de erro final é baseada no valor médio de todas as métricas de erro.

Foi decidido colocar em prática o *Cross-validation* devido a uma das suas vantagens: diminuição do *overfitting*. O *overfitting* consiste na produção de uma análise que corresponde aproximadamente a um determinado conjunto de dados e, portanto, pode falhar ao se ajustar aos dados adicionais ou prever observações futuras sem fiabilidade.

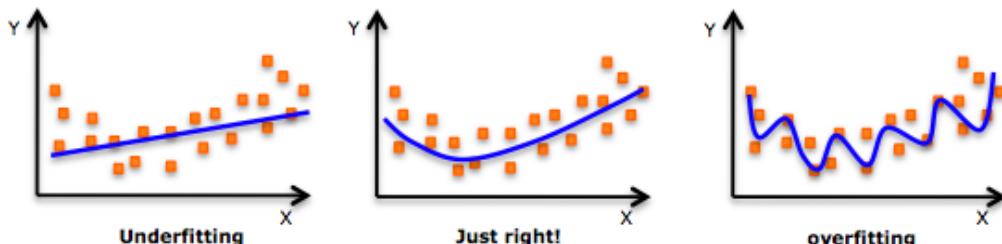


Figura 44: Tipos de modelos

Queremos com isto dizer que o *overfitting* reflete que o modelo memoriza e modela demasiado o *training set*, originando uma previsão errada caso se utilize outro *dataset* no modelo.

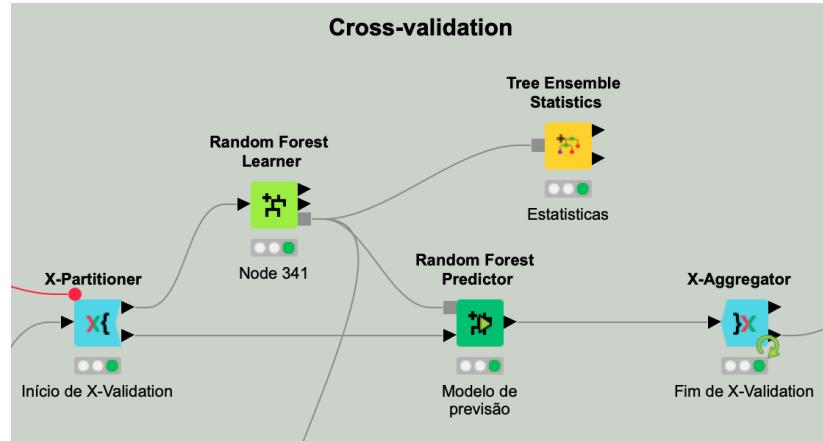


Figura 45: *Cross-validation* no *Workflow*

Assim sendo para a construção deste modelo optou-se por atribuir uma valor de 5 ao *k-fold* ($k=5$). No entanto na figura seguinte, é ilustrado um exemplo de *Cross-validation* com $k-fold=10$.

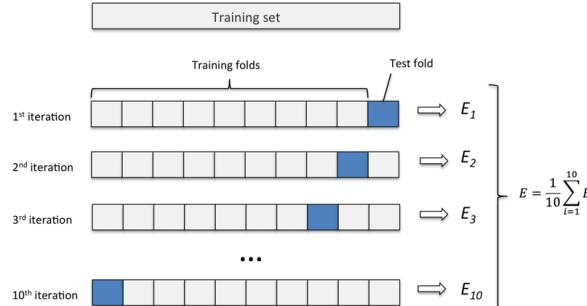


Figura 46: *Cross-validation* com $k\text{-}fold}=10$

Utilizou-se um *X-Partitioner* e um *X-Aggregator* para que fosse possível aplicar esta técnica. O primeiro faz a repartição do *dataset* pelo *k-fold* definido, ou seja, por $k=5$ (5 partes iguais) com *sampling* aleatório. Já o segundo serve para definir a coluna "alvo" e a coluna da previsão, retornando a tabela das previsões já com a percentagem de erro de cada *fold*.

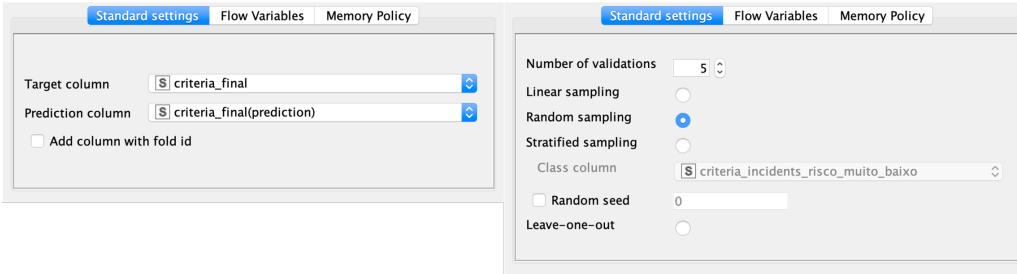


Figura 47: Definições dos nodos *X-Partitionere X-Aggregator*

6.1.3 Tuning final

Por fim, com o modelo construído na fase anterior, defini-se o *Tunning* final de maneira a que seja possível calcular o *accuracy* do modelo. Optou-se por se utilizar um *GroupBy*, que através das percentagens fornecidas pelo *X-Aggregator*, calcula a média dos erros. Esta média é usada posteriormente no *Math Formula* que efectua o cálculo da *accuracy* segundo a fórmula $accuracy = 100 - mean_error$.

Seguidamente no *Parameter Optimization Loop End* selecionou-se a função objetivo que se queria maximizar, isto é, neste caso selecionou-se a opção *maximized* para a *accuracy*. Estes dados são, por fim, passados ao *Variable Loop End* onde foram definidos os parâmetros a otimizar. Colocou-se ainda um *GroupBy* com o intuito de calcular a média final do valor da função objetivo, ou seja, calcular o valor médio da *accuracy* do *output*.

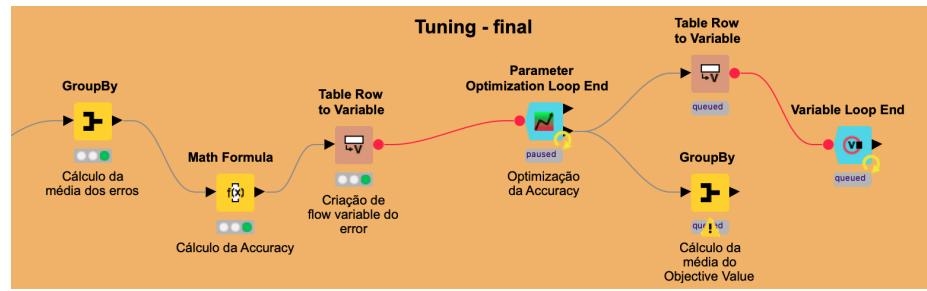


Figura 48: Parte final do *tuning*

gerado. Este *GroupBy* foi apenas utilizado para uma avaliação do *output* por parte do grupo de trabalho.

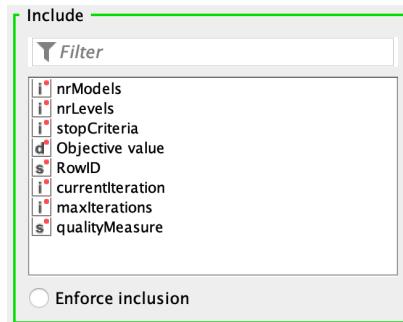


Figura 49: Definições *Variable Loop End*

No entanto, achamos que no contexto do problema não seria a melhor opção, por isso, e como já foi referido, escolheu-se outra abordagem para a criação deste modelo.

6.2 Artificial neural network em Python

A nossa segunda opção foi a construção de uma rede neuronal, mais concretamente uma *Multi Layer Perceptron*, que fosse capaz de suprimir as nossas necessidades. No entanto ao longo do desenvolvimento do projeto, foram criadas duas abordagem onde podemos considerar que a 2^a abordagem é construída com base na 1^a abordagem. Explicaremos agora os dois casos.

6.2.1 1^a Abordagem

Numa primeira abordagem, começou-se por transformar o *dataset* gerado no KNIME na fase do tratamento de dados através da criação do script "encoding_data.py". Neste script efetuou-se uma leitura do *dataset* gerado e realizou-se um *label encoding* para todas as colunas do tipo objeto, originando um novo ficheiro com o conjunto de dados que iremos utilizar. Este conjunto de dados após o *labelling* encontra-se bem balanceado, uma vez que não existem dados muito dispersos e, devido ao facto de se utilizar um critério que varia entre 1 e 5, permitiu-nos balancear todos os tipos de dados tanto as *features* como o *target*. Posto isto, desenvolveu-se um script para criar o modelo da *Multi Layer Perceptron*.

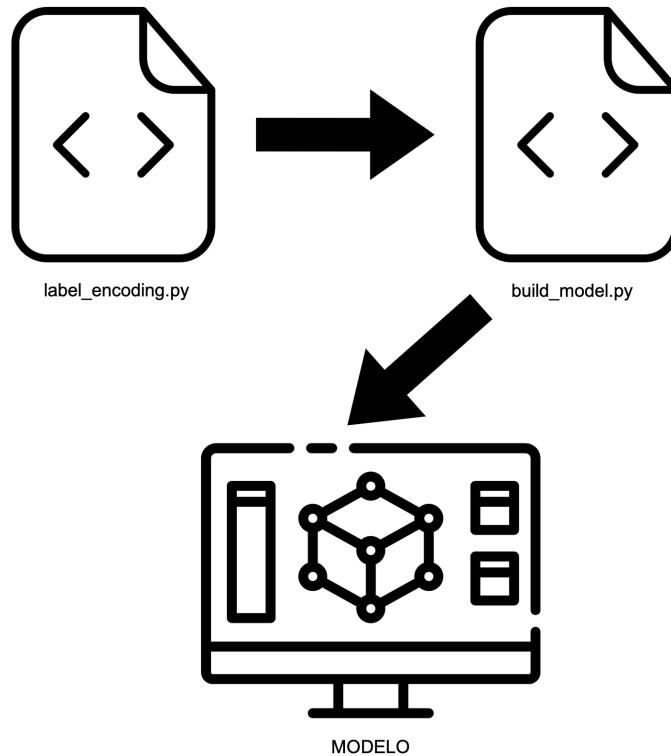


Figura 50: Processo de obtenção do modelo

Como se sabe, um dos problemas iniciais na utilização de dados foi a obtenção de dados de interesse com os quais a rede iria ser alimentada. Logo foi necessário carregar o ficheiro com o *dataset* final o qual tínhamos denominado de "dataset_labelled.csv" e guardar num *dataframe*. De seguida, foi necessário preparar toda a informação disponível definindo todas as *features* e o nosso alvo (*target*). Neste caso, todas as colunas são *features* com excepção da coluna "criteria_final" que é nosso *target*, ou seja, o que pretendemos prever.

Definidos as nossas *features*, é indispensável realizar uma normalização das mesmas. Assim sendo, todos os dados das mesmas foram normalizados para valores entre -1 e 1 e guardados num novo *dataframe*. Para além da criação deste novo *dataframe*, também foi gerado e guardado um *scaler* por intermédio da função "*MinMaxScaler*", função que onde o *scaler* dimensiona e traduz cada *feature* individualmente, de modo que esteja no intervalo especificado no conjunto de treino.

Processada a normalização das *features*, realizou-se um *one hot encoding*, isto é, aplicamos um processo pelo qual as variáveis categóricas são convertidas num formato que pode ser fornecido aos algoritmos de *Machine Learning* para fazer uma melhor previsão. Neste caso, fornecemos o *target* para o nosso *one hot encoder* aplicando-lhe também um *reshape* entre -1 e 1 e guardando a informação num novo *dataframe*.

Finalmente, estes dois novos *dataframes* gerados anteriormente são utilizados para realizar a divisão entre os dados de teste e os dados de treino. Para tal utilizou-se a função "*train_test_split*" que divide *arrays* ou matrizes em subconjuntos aleatórios de dados de treino e de teste, e definimos o tamanho do teste para 33% do *dataframe*.

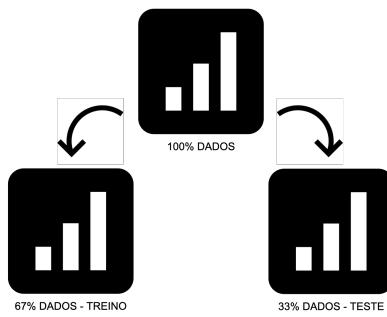


Figura 51: Divisão dos dados

Nesta fase, está tudo pronto para construir o modelo da MLP e realizar o posterior treino do modelo. O treino da rede neuronal escolhida foi maioritariamente todo ele feito no GPU da plataforma *Google Colab*. Além disso, os computadores pessoais de cada elemento também foram usados para treino, apesar de o poder computacional ser menor, quando comparado com o da máquina disponibilizada pelo *Google Colab*, o que tornava mais lenta a obtenção de resultados. Para além disso, de modo a poder medir os tempos de treino de forma uniforme, era importante usar sempre o mesmo GPU. De feita, de modo a que fosse possível realizar este processo, foi necessário definir alguns parâmetros.

- Número de neurónios: $nr_neurons = 128$;
- Número de épocas: $epochs = 500$;
- *Dropout*: $droup_out = 0.4$;
- *Batch size*: $batch_size = 128$;
- *Verbose*: $verbose = 1$;
- Paciência: $patience = 30$;

Para além destes parâmetros, foi necessário a criação de duas *callbacks* para utilizar no treino do modelo. Com o auxílio destas *callbacks* será possível evitar o *overfitting*, uma vez que o treino do modelo será encerrado atempadamente, ou seja, quando este não precisar de percorrer as épocas todas, irá parar o treino.

```
lr = tf.keras.callbacks.ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.3,
    patience=patience,
    mode='auto',
    min_lr=0.00005)
early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    min_delta=0,
    patience=patience,
    verbose=verbose,
    mode='auto')
```

Falando mais concretamente da construção do modelo, este é composto por 6 *layers* com a ativação *relu* onde foram aplicadas várias variações do número de neurónios:

- 1^a *Layer*: duas vezes o número de neurónios;
- 2^a *Layer*: quatro vezes o número de neurónios;
- 3^a *Layer*: duas vezes o número de neurónios;
- 4^a *Layer*: o número de neurónios;
- 5^a *Layer*: metade do número de neurónio;
- 6^a *Layer*: um quarto do número de neurónios;

Para a *loss* do modelo foi utilizado *categorical_crossentropy*, para o *optimizer* foi utilizado o *adam* e como métrica utilizou-se a *accuracy*. Na figura seguinte, está apresentado o modelo completo.

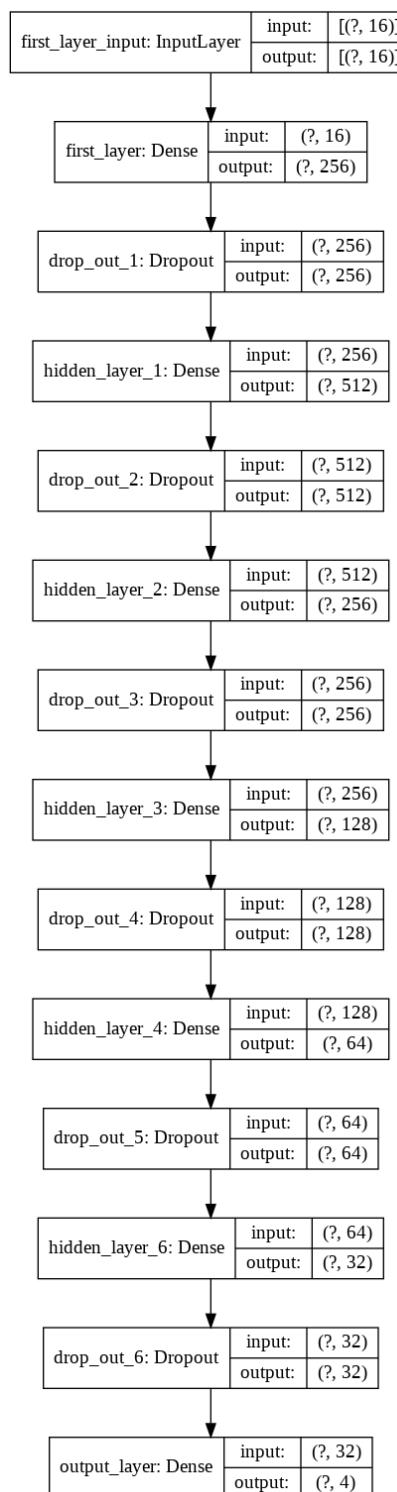


Figura 52: Modelo final da MLP

Construído o modelo, foi realizado o treino do mesmo e foram utilizadas as *callbacks* definidas acima, permitindo um maior controlo sobre o treino. Em jeito de sumário, apresentamos um esquema de todo o processo de implementação seguido ao longo da execução de abordagem.



Figura 53: Processo de implementação

6.2.2 2^a Abordagem

Nesta segunda abordagem, foi utilizado o modelo da primeira abordagem com várias alterações. O objetivo principal foi a introdução do *Cross-validation* tal como foi utilizado na *Random Forest*. Assim sendo, efetuou-se mais uma vez a leitura do *dataset* gerado e necessitou-se de preparar os dados lidos.

Ao invés de se realizar esta preparação através de diferentes funções, concentrou-se tudo na função *data_prepare*, onde foi se define mais uma vez o *target* do problema e as suas *features*. Efetua-se a normalização das *features* da mesma forma que se tinha realizado na abordagem anterior tal como o *one hot encoding* do *target*.

Ainda na função *data_prepare* realiza-se a divisão dos dados para treino e para teste. Utiliza-se mais uma vez a função *train_test_split* em que a percentagem para teste desta vez fixou-se nos 25% do total dos dados. Posto isto, removeu-se no *X_train* e no *X_train* as novas colunas com os *targets* e nos *Y_train* e *Y_train* removeu-se as *features*.

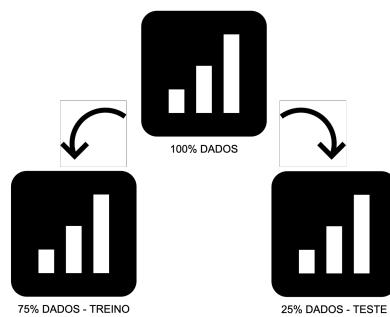


Figura 54: Divisão dos dados

Posto isto, aplicou-se o *Cross-validation* que nos permite gerar vários modelos mediante o *k-fold* definido e permite-nos escolher o melhor modelo, uma vez que os guarda cada modelo por cada validação. Neste processo utiliza-se a mesma a função para a construção do modelo, a *build_model* e utilizam-se as mesmas *callbacks*, a *early_stop* e a *lr*, ou seja, a rede criada é exatamente a mesma da primeira abordagem. Já a escolha do modelo a utilizar irá tender para o modelo modelo do *k* gerados.

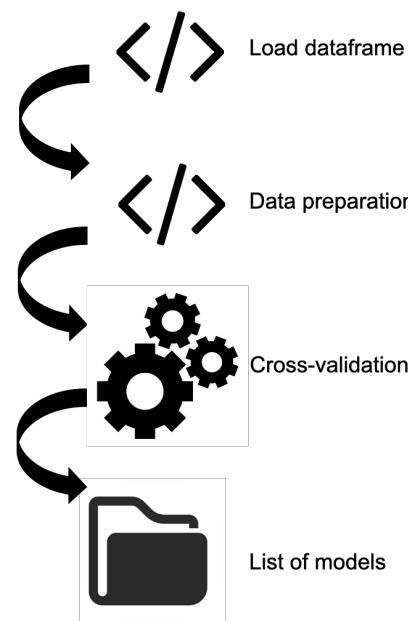


Figura 55: Processo completo da 2^a abordagem

A grande diferença desta abordagem para a primeira, foi o *Cross-validation* tal como foi referido. A aplicação deste conceito, permitiu que fosse possível gerar vários modelos com uma boa *accuracy* e com pouco *loss* e dentro destes modelos gerados pode-se escolher o melhor para se utilizar como modelo final, como foi referido anteriormente.

Tal como foi realizado anteriormente, em jeito de sumário, apresentamos um esquema de todo o processo de implementação seguido ao longo da execução de abordagem.



Figura 56: Processo de implementação

Os resultados obtidos com esta rede neuronal foram bastante satisfatórios, assim como os resultados da primeira abordagem, daí a decisão do grupo cair sobre esta metodologia em detrimento da *Random Forest*. No entanto, no capítulo seguinte iremos apresentar os resultados dos três modelos e realizar uma avaliação detalhada dos mesmos.



7 Análise dos resultados

Neste capítulo iremos efetuar uma análise aos resultados obtidos na execução dos três modelos explicados e desenvolvidos no capítulo anterior. Como se vai verificar, os resultados da *Random Forest* foram bastante bons tal como os resultados obtidos nas MLPs desenvolvidas, mas, tal como foi dito anteriormente, as *Multi Layers Perceptrons* adequam-se mais ao nosso problema e gera um modelo mais fidedigno e viável.

Iremos primeiro apresentar os diferentes resultados obtidos e posteriormente uma análise final com a comparação dos resultados das 3 abordagens numa tabela comparativa.

7.1 Random Forest

Posto isto, obteram-se resultados bastante satisfatórios na execução deste modelo, tal como se pode verificar na figura seguinte. As três *quality measures* geraram valores na ordem dos 95%, valores altos e bastante bons para a quantidade de dados que estavam disponíveis para treino e teste.

Row ID	stopC...	nrModels	nrLevels	Objective value	RowID	curren...	maxit...	qualityMeasure
Row0	1	500	40	95.125	Best parameters	0	3	InformationGain
Row1	1	500	20	95.234	Best parameters	1	3	InformationGainRatio
Row2	1	500	20	94.941	Best parameters	2	3	Gini

Figura 57: Resultados do modelo *Random Forest*

7.2 Artificial neural network em Python

No caso do desenvolvimento da rede neuronal, ambas as duas abordagens arrecadaram resultados muito satisfatórios. No entanto a abordagem utilizada como modelo final foi a segunda abordagem, uma vez que é um modelo mais completo devidos aos fatores que já foram mencionados anteriormente.

7.2.1 1ª Abordagem

```
71/71 [=====] - 0s 2ms/step - loss: 8.6623e-04 - accuracy: 0.9996
Evaluation ['loss', 'accuracy']: [0.0008662297623232007, 0.999555766582489]
```

Figura 58: Métricas da MLP

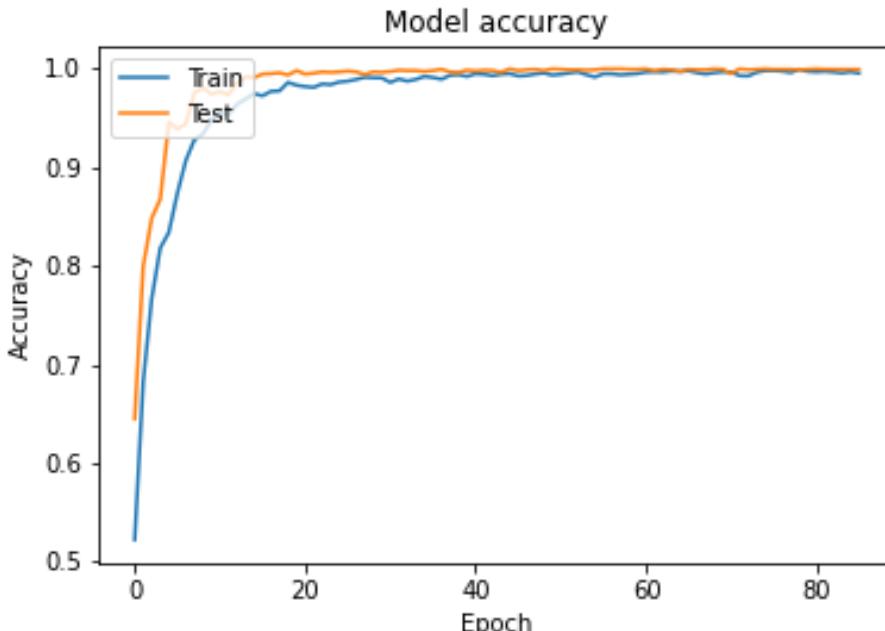


Figura 59: Gráfico da *accuracy* da MLP

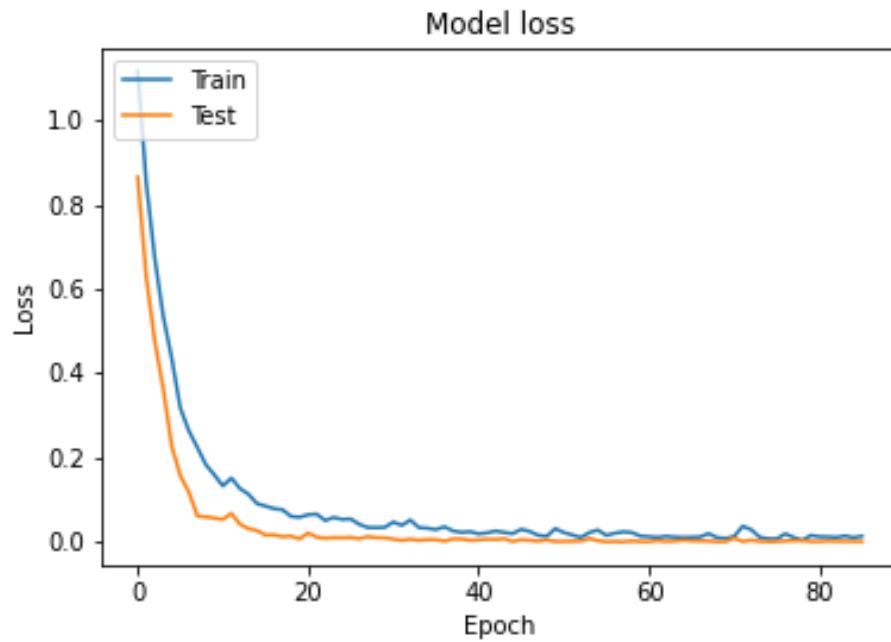


Figura 60: Gráfico da *loss* da MLP

7.2.2 2^a Abordagem

Como nesta abordagem foram gerados mais que um modelo, vamos apresentar dois gráficos: o primeiro com a evolução da *accuracy* à medida que foram gerados os modelos e o segundo com a evolução da *loss* à medida que foram gerados os modelos.

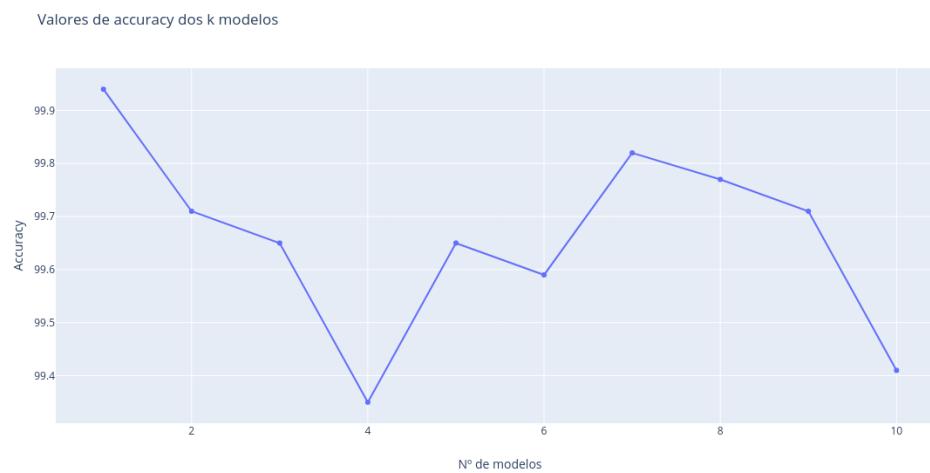


Figura 61: *Accuracy* dos *k* modelos

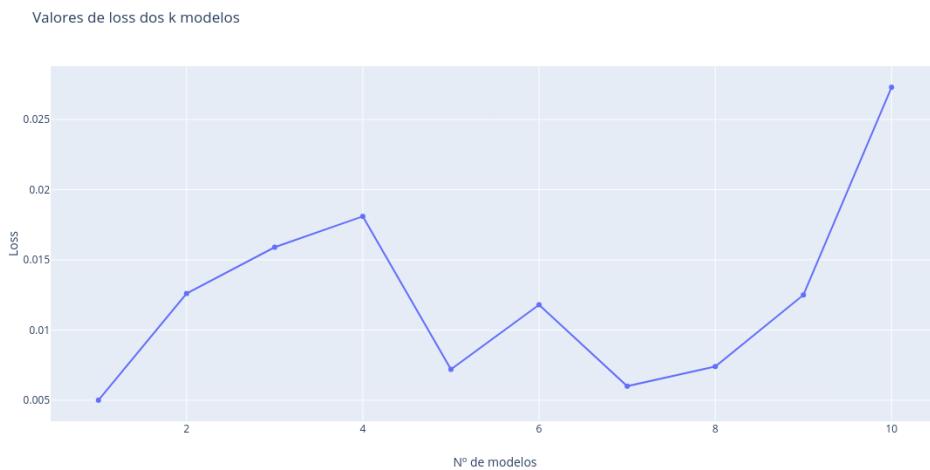


Figura 62: *Loss* dos k modelos

Analizados os k modelos, é possível escolher o melhor modelo. A escolha diz respeito ao primeiro modelo gerado com uma *accuracy* de 99.94% e uma *loss* de 0.005%, tal como se pode ver nos dados seguintes.

```
54/54 [=====] - 0s 2ms/step - loss: 0.0097 - accuracy: 0.9994
accuracy: 99.94%
```

Figura 63: Métricas da MLP

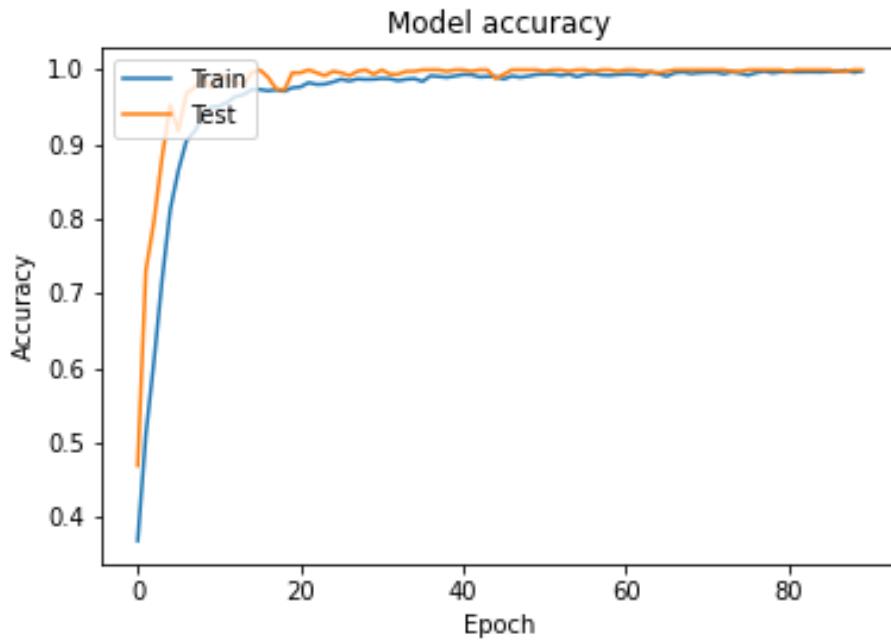


Figura 64: Gráfico da *accuracy* da MLP

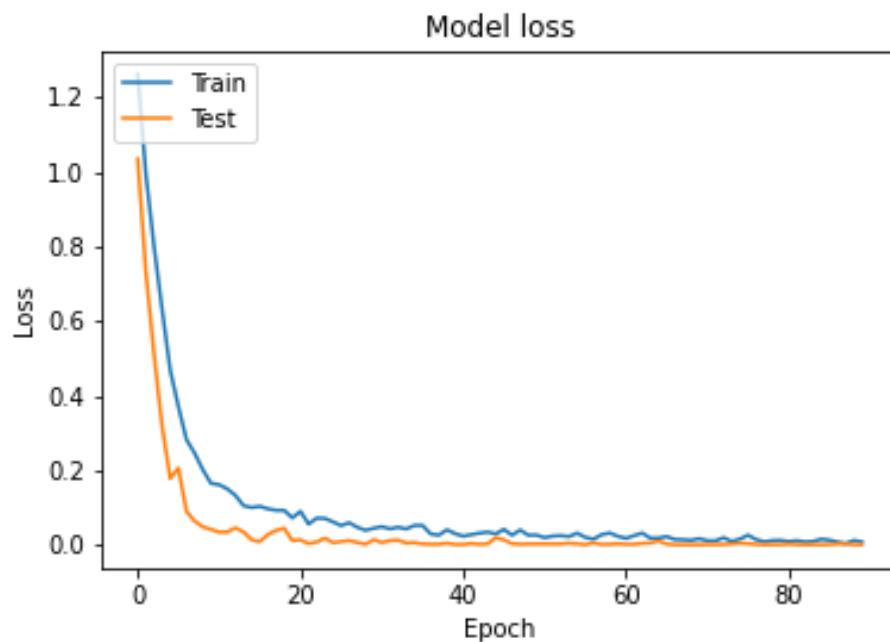


Figura 65: Gráfico da *loss* da MLP

7.3 Comparação final

Como podemos visualizar na tabela seguinte e tendo em consideração o que foi dito anteriormente, a escolha final irá recorrer pela rede neuronal com *Cross-validation*. A *accuracy* do modelo obtido é bastante alta, consequentemente a *loss* é bastante baixa, e a viabilidade do modelo é a melhor entre as 3 abordagens. Em relação ao tempo de execução é bastante satisfatório.

Método	Accuracy	Loss	Tempo Execução	Viabilidade
Random Forest	95.234%	4.766%	Lento	Pouco viável
MLP	99.96%	0.04%	Muito rápido	Viável
MLP (Cross-validation)	99.94%	0.06%	Rápido	Muito viável

Tabela 13: Tabela com resultados 3 modelos



8 Criação da plataforma

Neste capítulo, vamos apresentar duas fases da criação da aplicação. Numa primeira fase foi necessário introduzir o modelo construído no capítulo anterior num servidor *online*. Já numa segunda fase, foi a criação da aplicação em *React Native* e a sua ligação com o servidor.

8.1 Desenvolvimento do servidor

Para que o modelo fosse utilizado *online* a cada iteração com o utilizador, foi necessário colocá-lo num servidor *online*. A nossa escolha recaiu pela *Amazon Web Services* devido ao serviço gratuito que eles disponibilizam bem como a segurança que eles oferecem ao utilizador.



Figura 66: *Amazon Web Services*

Amazon Web Services, também conhecido como AWS, é uma plataforma de serviços de computação em *cloud* formando uma plataforma de computação na *cloud* oferecida pela *Amazon.com*. Esses serviços *web* de computação em *cloud* fornecem um conjunto de infra-estruturas técnica abstratas primitivas, componentes e ferramentas de computação distribuída aos seus utilizadores, ou seja, servidores.

Um desses serviços é o *Amazon Elastic Compute Cloud*, que permite que os utilizadores tenham à sua disposição um *cluster* virtual de computadores, disponível em tempo real, pela Internet. A versão dos computadores virtuais da AWS emula a maioria dos atributos de um computador real, incluindo unidades centrais de processamento de *hardware* (CPUs) e unidades de processamento gráfico (GPUs) para processamento. Memória local e RAM, armazenamento em disco rígido ou SSD, uma escolha de sistemas operacionais, rede e programas pré-carregados.

Os seus serviços são disponibilizados em várias zonas geográficas distribuídas pelo mundo e os seus serviços mais conhecidos são o *Amazon Elastic Compute Cloud*, mencionado anteriormente, e o *Amazon S3*.

Dito isto, a escolha deste serviço pareceu-nos bastante sensata e eficaz para o problema que tínhamos em mãos. O processo tem duas partes, numa primeira fase foi necessário definir o tipo de máquina que queríamos utilizar no AWS e colocar na mesma o nosso modelo elaborado anteriormente. Numa segunda fase, foi o desenvolvimento do servidor.

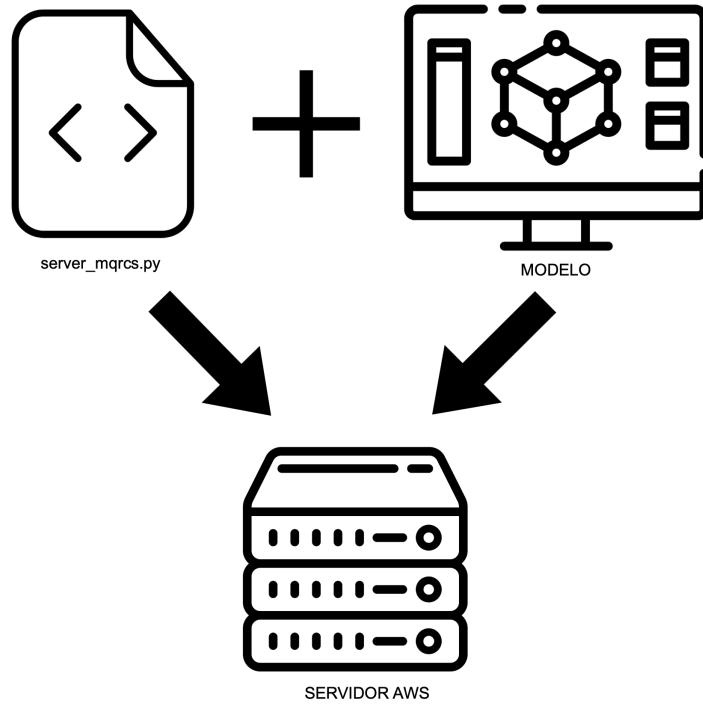


Figura 67: Processo de desenvolvimento do servidor

O desenvolvimento do servidor é iniciado pela leitura do modelo construído, a leitura do *scaler* definido na geração do modelo e, por último, a leitura do conjunto de dados utilizado.

Assim sendo, retirou-se o *target*, ou seja, retirou-se a coluna *criteria_final* e apenas se manteve as *features* no *dataset*. Retirado o *target*, realizou-se a normalização dos dados com o *scaler* anterior.

Com os dados preparados, definiu-se os dois tipos de pedidos que podem ser feitos ao servidor, um GET ou um POST. Em ambos os tipos de pedido, são recolhidos ou enviados os dados da hora, dia e mês para efetuar a previsão, com estes dados geramos um novo valor do género "hora/dia/mês" tal como o tipo de dados disponível na coluna *index*. Este novo valor é utilizado para bloquear a célula com aquele mesmo valor no *dataframe* normalizado.

Neste momento, irá ser realizado a previsão, onde primeiro verificamos se o *dataframe* está vazio e caso isso acontecer, retorna-se o valor -1, caso contrário realiza-se o *predict*, converte-se o resultado numa *label* e retorna-se o mesmo.

É importante referir que a informação que é utilizada para a previsão, mais concretamente a hora, dia e mês é fornecida pela aplicação bem como o resultado da previsão que é enviado para a aplicação para a posterior recomendação para o utilizador. Assim sendo, vamos explicar o processo da criação e desenvolvimento da aplicação, a qual denominamos de GoSafe.

8.2 GoSafe

Mas o que é a GoSafe? A GoSafe é o resultado final do projeto, é a aplicação que foi desenvolvida para os VRUs. Para a criação desta APP utilizou-se a *framework React Native*.

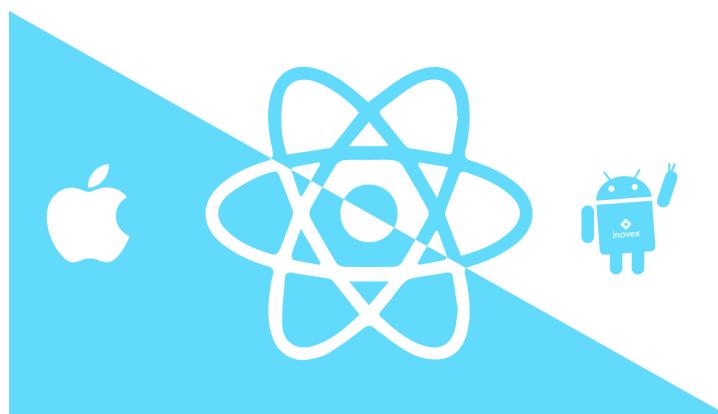


Figura 68: *React Native*

O *React Native* é uma *framework* baseado no *React*, desenvolvido pelo *Facebook*, que permite a criação de *mobile apps* tanto para *Android* como para *IOS* através de *Javascript* e *NodeJS*. A opção por esta *framework* advém da experiência que ela num outro projeto e, como tal, sabemos das suas vantagens, tais como:

- Permitir o desenvolvimento de aplicações para *Android* e *IOS* (já mencionado);
- A interface desenvolvida é bastante fluída;
- Os carregamentos da aplicação em geral são rápidos;
- Melhor integração com as funções do *smartphone* como camera, giroscópio, etc.;
- Dispõe de *Hot Reloading*, que permite que o programa fique em execução e a cada atualização no código a aplicação é compilada novamente;
- Maior segurança;
- Melhor *performance*;

Posto isto, iniciou-se o desenvolvimento da aplicação. Para tal, começou-se por definir um logótipo para a GoSafe, como se pode verificar na figura seguinte, e optou-se por uma abordagem simples definindo-se um menu simples e intuitivo.



Figura 69: Logótipo da aplicação

Em termos visuais, o utilizar ao iniciar a aplicação, o utilizador visualiza a plataforma que é apresentada na figura seguinte e, para este receber a recomendação, basta clicar no botão "*Check Predictions*" que imediatamente faz com que seja lançado um *alert* com a referente recomendação para aquela hora naquele dia do mês específico.

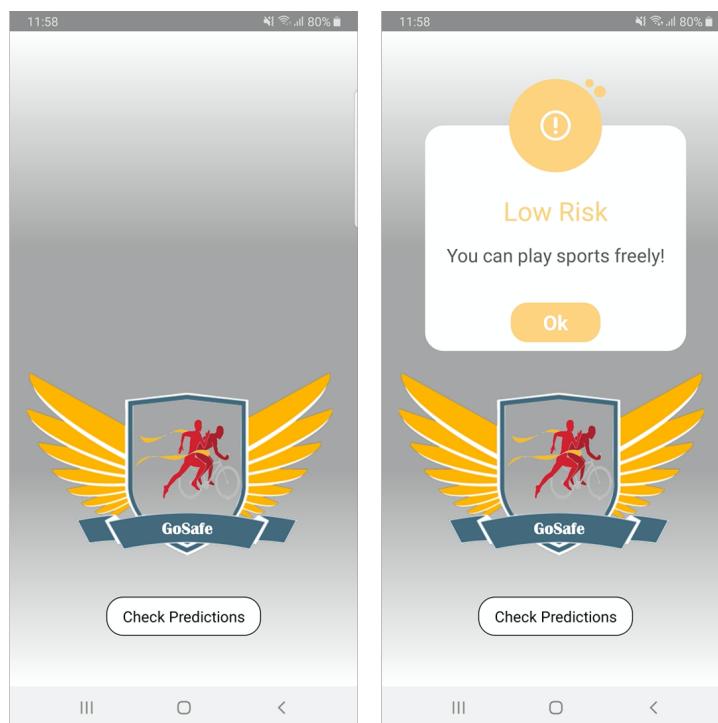


Figura 70: Plataforma *mobile*

Este *alert* para a recomendação advém da função definida `_showAlert` que consoante o *criteria* recebido do servidor. Cada tipo de *criteria* tem um *alert* definido, inclusive nos casos em que o servidor não conseguiu gerar uma previsão ou simplesmente existiu um erro no sistema.

```
_showAlert = (criteria) => {
    if (criteria === '-2') {
        this.setState({
            title: 'Error',
            message: 'Something went wrong..',
            status: 'error-info',
        })
    } else if (criteria === '-1') {
        this.setState({
            title: 'Info',
            message: 'There is no forecast for this time!',
            status: 'info',
        })
    } else if (criteria === '0') {
        this.setState({
            title: 'Minimal Risk',
            message: 'Highly recommended to practice sports!',
            status: 'success',
        })
    } else if (criteria === '1') {
        this.setState({
            title: 'Low Risk',
            message: 'You can play sports freely!',
            status: 'warning-light',
        })
    } else if (criteria === '2') {
        this.setState({
            title: 'Moderate Risk',
            message: 'Practice sport in moderation!',
            status: 'warning',
        })
    } else if (criteria === '3') {
        this.setState({
            title: 'High Risk',
            message: 'Sport is not recommended! Stay at home!',
        })
    }
}
```



```
        status: 'error',
    })
}
this.setState({
    visible: true
})
}
```

Para que fosse possível obter as previsões do servidor, foi necessária desenvolver a função assíncrona que se denominou de `_checkPredictions`. Foram recolhidas as informações do dispositivo relativas ao dia, hora e mês que foram posteriormente fixados no url do servidor para se realizar o pedido ao mesmo. Este pedido foi efetuado através do `axios` no qual se faz um `get` com o url definido.

Caso exista resposta da parte do servidor, a função `_showAlert` é invocada com o `criteria` fornecido pelo servidor, caso não exista resposta, a função `_showAlert` é invocada na mesma, embora lhe seja fornecido o valor de `criteria -2` aparecendo ao utilizar o `alert` de erro.

```
_checkPredictions = async () => {
    const today = new Date();
    const day = today.getDate();
    const hour = today.getHours();
    let month = today.getMonth();
    month = month + 1;
    const headers = {
        "Content-Type": "application/x-www-form-urlencoded",
        "Accept": "application/json",
    }
    const url = 'http://ec2-3-21-167-77.us-east-2.compute.amazonaws.com:5000/
                    predict?h=${hour}&d=${day}&m=${month}'
    await axios.get(url, { headers })
        .then((response) => {
            const data = response.data;
            this._showAlert(data.criteria);
        })
        .catch((error) => {
            this._showAlert('-2');
        });
};
```



9 Conclusão

Neste capítulo, faz-se uma conclusão do projeto desenvolvido durante todo o semestre e faz-se um lamiré do possível trabalho que poderá vir a ser realizado no futuro.

9.1 Conclusões e trabalho futuro

O grupo conseguiu articular e integrar os diferentes métodos de *Machine Learning* e inteligência artificial aprendidos na universidade, além de habilidades autodidatas em relação à criação da aplicação móvel, principalmente o uso de uma *framework* que atualmente é muito procurada no mercado, única e coesa e bem fundamentada.

Embora tenham surgido algumas dificuldades, como na escolha do melhor modelo de *Machine Learning* a ser implementado, foi necessário realizar um tratamento do conjunto de dados para garantir que não existe algum problema que pudesse ocorrer devido a incompatibilidades com o modelo escolhido e imprecisão em relação aos limites teóricos de cada *features*. Após a preparação da implementação do modelo escolhido, foi apenas uma questão de utilizar o conhecimento adquirido através dos nossos professores na universidade e pela investigação realizada por nós.

Em relação ao aplicativo móvel, ocorreram algumas dificuldades triviais, mas não foi nada que não foi tratado rapidamente com algumas pesquisas na Internet. Por isso, o grupo concorda por unanimidade que o ponto mais importante nesse projeto é o modelo de previsão desenvolvido e com altos valores de desempenho. Embora esse desempenho ainda seja calculado com os dados disponíveis, acredita-se que, após entrar em circulação com a povoação de Braga, certamente será muito preciso, e ainda seria possível procurar novas abordagens para melhorá-lo. Isso pode ser feito adicionando novos tipos de informações sobre a segurança da VRU, bem como treinando com uma quantidade maior de dados.

Por último, achamos importante referir que devido à situação que o país e o mundo atravessaram nos últimos meses e ainda travessam devido ao *Covid-19*, afirmamos com clareza que a aplicação desenvolvida não irá fornecer as recomendações necessárias para este período de confinamento social.



Referências

- [1] Deep Learning. URL <https://developer.nvidia.com/deep-learning>.
- [2] An open source machine learning framework for everyone. URL <https://www.tensorflow.org>.
- [3] Keras: The python deep learning library. URL <https://keras.io>.
- [4] Keras examples directory. URL <https://github.com/keras-team/keras/tree/master/examples>.
- [5] Caffe - deep learning framework. URL <http://caffe.berkeleyvision.org>.
- [6] Tensors and dynamic neural networks in python with strong gpu acceleration. URL <http://pytorch.org/>.
- [7] Theano. URL <http://deeplearning.net/software/theano/>.
- [8] Machine Learning. URL https://www.sas.com/pt_br/insights/analytics/machine-learning.html.
- [9] CRISP-DM. URL https://pt.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining
- [10] Health problems of Overexposure to the Sun. URL https://www.epa.gov/sites/production/files/documents/healtheffects_1.pdf
- [11] Carbon monoxide poisoning. URL https://en.wikipedia.org/wiki/Carbon_monoxide_poisoning
- [12] Air Quality Guide for Nitrogen Dioxide. URL <https://www3.epa.gov/airnow/no2.pdf>
- [13] Nitrogen Dioxide (NO₂) Pollution. URL <https://www.epa.gov/no2-pollution/basic-information-about-no2>
- [14] Dióxido de enxofre: conheça o SO₂. URL <https://www.ecycle.com.br/2409-dioxido-de-enxofre-so2>
- [15] Ozone Safety. URL <https://www.ozonetech.com/ozone-safety>
- [16] Partículas Inaláveis – PM10 e a Qualidade do Ar Ambiente. URL <https://www.apopartner.pt/particulas-inalaveis-pm10-e-a-qualidade-do-ar-ambiente/>
- [17] Pressão Atmosférica. URL <https://globoesporte.globo.com/eu-atleta/saude/noticia/doenca-da-altitude-causas-sintomas-e-prevencao-do-problema-de-montanhistas.ghtml>
- [18] Humidade. URL <https://www.uol.com.br/vivabem/noticias/redacao/2018/07/07/como-a-baixa-umidade-do-ar-afeta-a-saude-veja-como-se-protoger-dos-efeitos.htm>
- [19] CRISP-DM methodology leader in data mining and big data, <https://towardsdatascience.com/crisp-dm-methodology-leader-in-data-mining-and-big-data-467efd3d3781>
- [20] Cross-Industry Standard Process for Data Mining FEUP, https://paginas.fe.up.pt/~ec/files_0405/slides/02%20CRISP.pdf?fbclid=IwAR26NsvrzD10z_jF3QKwiTUZKFvtGJrr28gW_WNeIFX-U-0fmv0ZKImx1Y
- [21] What is the CRISP-DM methodology?, https://www.sv-europe.com/crisp-dm-methodology/?fbclid=IwAR2y1dQ_JkikLr1eA10D5u4bFjij24p6c8CRAFIWdYVR0SQgHwARCeCy5UI#modeling