NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

RUI RITA NASCIMENTO

Licenciado em Engenharia Eletrotécnica e de Computadores

COLLECTING DATA WITH ANDROID AND ANALYSING IT WITH DECISION TREES

Dissertation Plan

MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa

Draft: *2 de abril de 2024*

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

COLLECTING DATA WITH ANDROID AND ANALYSING IT WITH DECISION TREES

RUI RITA NASCIMENTO

Licenciado em Engenharia Eletrotécnica e de Computadores

Orientadores: Pedro Alexandre Sousa
Associate Professor, NOVA University Lisbon
João Paulo Pimentão
Assistant Professor, NOVA University Lisbon

Dissertation Plan
MESTRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES

Universidade NOVA de Lisboa
Draft: *2 de abril de 2024*

Resumo

O setor das seguradoras tradicionalmente baseia a avaliação de riscos em características qualitativas dos condutores, o que pode levar à discriminação com base em idade, histórico de sinistros e educação. No entanto, iniciativas recentes procuram complementar essa avaliação com dados quantitativos de condução, como o Usage-Based Insurance (UBI) e o Pay-As-You-Drive (PAYD). Estes métodos utilizam dados de condução em tempo real, oferecendo uma abordagem personalizada e ajustando os custos com base no comportamento real dos condutores. Esta dissertação propõe o desenvolvimento de uma aplicação que recolhe dados de condução e os envia para análise na nuvem, utilizando técnicas de modelação e Machine learning (ML), nomeadamente árvores de decisão, para enriquecer a definição do perfil do condutor. O resultado será um modelo capaz de distinguir entre condutores com comportamento agressivos (alterações bruscas de velocidade) e comportamento não agressivos, com potenciais aplicações no setor das seguradoras.

Palavras-chave: Perfil de condutores, *Machine Learning*, Árvores de Decisão, *Android*, *Cloud*

Abstract

The insurance industry has traditionally based risk assessment on qualitative characteristics of drivers, which can lead to discrimination based on age, claims history and education. However, recent initiatives seek to complement this assessment with quantitative driving data, such as UBI and PAYD. These methods use real-time driving data, offering a personalized approach and adjusting costs based on drivers' actual behavior. This dissertation proposes the development of an application that collects driving data and sends it for analysis in the cloud, using modeling and ML techniques to improve driver profiling. The result will be a model capable of distinguishing between drivers with aggressive behavior (sudden speed changes) and non-aggressive behavior, with potential applications in the insurance sector.

Keywords: Driver profile, *Machine Learning*, Decision Trees, *Android*, *Cloud*

Índice

Índice de Figuras

Índice de Tabelas

Siglas

AI Artificial Intelligence *(p. 17)*

CAN Controller Area Network *(p. 5)*

DTW Dynamic Time Warping *(pp. 6, 8, 13)*
DTW-SOM Dynamic Time Warping Self-Organizing Map *(pp. vi, 13, 14)*

GPS Global Positioning System *(pp. 5, 11, 15, 20, 21)*
GRU Gated Recurrent unit *(p. 16)*

IEEE Institute of Electrical and Electronics Engineers *(p. 4)*

K-NN K-Nearest Neighbor *(p. 8)*

LSTM Long-short term memory *(pp. 15, 16)*

ML Machine learning *(pp. ii, iii, 17–19, 21, 24, 26, 31)*

NLP Natural Language Processing *(p. 18)*

OBD On-Board Devices *(p. 11)*

PAA Piecewise Aggregate Approximation *(p. 12)*
PAYD Pay-As-You-Drive *(pp. ii, iii, 1)*

RNN Recurrent Neural Network *(pp. ix, 15–17)*

SAX Symbolic Aggregate Approximation *(pp. vi, 11–13, 20, 29–31, 37)*
SFFS Sequential forward feature selection *(p. 10)*

SOM Self-Organizing Map *(p. 13)*

UBI Usage-Based Insurance *(pp. ii, iii, 1)*

VSAX Variable Symbolic Aggregate Approximation *(p. 13)*

1

Introdução

O perfil de condutores é uma forma de caracterização que procura recorrer a métricas ou critérios, que possibilitem a discriminação dos mesmos. Essa classificação pode ser direcionada para categorização ou orientada a uma avaliação contínua, para identificação da tendência do perfil do condutor da perspetiva de um espetro que nos extremos defina características opostas. As características podem ser de caráter qualitativo, como dados demográficos e a situação socioeconómica, como caráter quantitativo, como as medidas da velocidade e da aceleração.

Os propósitos podem ser diversos como o estudo de tendências de destino, como a avaliação da segurança do condutor. Dessa forma, a análise do perfil do condutor é indispensável para uma compreensão de padrões, para facilitar a tomada de decisões, e que estas sejam sempre feitas o mais informadas possível, bem como a melhoria de estratégias relacionadas à segurança e eficiência no trânsito.

Em setores como as seguradoras, onde as empresas assumem riscos em troca valores acordados, valoriza-se bastante as características qualitativas para determinar o perfil dos condutores, pois é vista como uma forma de estimar o risco associado a cada condutor. Isso traduz-se em práticas que levam à discriminação dos clientes com base na idade, histórico de sinistros, nível de educação, entre outras. Surge nesse sentido uma oportunidade de se olhar para dados mais relevantes e menos subjetivos no que diz respeito à avaliação do risco.

Existem atualmente iniciativas que visam complementar a avaliação de risco dos clientes com base em dados quantitativos, dados esse baseados na condução, como o UBI ou o PAYD, cuja primeira depende da qualidade da condução em termos de velocidade, aceleração entre outros atributos, e a segunda depende da distância percorrida com o automóvel, oferecendo assim uma abordagem mais personalizada, ajustando os custos ao comportamento real de condução dos condutores e não às suas características demográficas.

Para obtenção dos dados existem diversos métodos de se monitorização, como a instalação de *blackboxes*, responsáveis por recolher estes dados. A instalação destes dispositivos envolvem custos não tão atrativos na questão da instalação e da manutenção. Além disso,

podem levantar questões relacionadas à privacidade dos condutores [4].

Para superar as desvantagens associada às *blackboxes*, os *smartphones* tornam-se populares como substitutos. Estes dispositivos possuem sensores embutidos, a capacidade de processamento, acesso à *internet*, são um bem comum e são de relativamente de baixo custo no que toca à manutenção, convenientes serem integrados no desenvolvimento de sistemas.

Baseado neste potencial dos *smartphones*, e nas necessidades de empresas que dependem de motoristas, surge a oportunidade de se desenvolver um sistema capaz de modelar a condução de forma que seja útil para essas mesmas empresas. No setor de seguros automóveis já existem soluções como, a *AvivaRateMyDrive* [24], *StateFarm DriverFeedback* [6], *AXA drive* [1].

Existe também a *Greenroad* [11] que é uma aplicação para as gestoras de frotas que coletam a informação dos condutores durante a sua atividade e atribuem uma descrição do indivíduo relativamente ao perigo da condução e do consumo de combustível.

A modelação da condução de veículos permite entender como os condutores se comportam em determinadas circunstâncias e assim abre oportunidade de melhoria na qualidade rodoviária, por exemplo, melhorando as infraestruturas, melhorando as tecnologias de auxílio à condução no próprio veículo ou até fornecendo uma compreensão sólida para que *software* seja desenvolvido para melhorar a prática da condução em si.

No contexto de seguradoras e outras organizações parecidas, a modelação da condução torna-se uma abordagem mais moderna, simples e eficaz para definir o perfil dos condutores, facilitando assim o processo de tomada de decisão. Nessa linha de pensamento, com o avanço da tecnologia, novas soluções modernas adequam-se a essa função, usando *smartphones*, *cloud computing*, *machine learning*.

Os *smartphones* permitem a ligação à *internet* e contam com uma coleção de sensores, úteis a estudos que dependem de dados monitorizados e flexibilidade a baixo custo. As *cloud computing* são uma forma de armazenamento e processamento através da *internet*, ou seja, nem os dados, nem os processos são armazenados e executados localmente, trazendo vantagens como escalabilidade e diminuição de custos em infraestruturas. O *machine learning*, por sua vez, são algoritmos e modelos que desenvolvidos para aprenderem padrões processando um conjunto de dados, sem que sejam programados explicitamente para desempenhar algo específico[21].

Soluções que integrem as três tecnologias mencionadas seguem a tendência do desenvolvimento tecnológico.

Nesta dissertação pretende-se desenvolver uma aplicação que recolha dados durante a condução de indivíduos, para posteriormente que esses dados sejam enviadados para uma infraestrutura *cloud* para serem analisados. Nessa análise procuramos validar a hipótese: Será possivel identificar o perfil de condução de condutores com o recurso a telefones moveis e algoritmos de aprendizagem automática para utilização no sector segurador?

A estrutura da dissertação é composta por cinco secções além da presente.

A primeira secção é a exploração dos estudos e pesquisas realizadas neste âmbito, com o propósito de identificar os expor a diversidade de abordagem. A segunda secção é a exposição da metodologia proposta, onde será explicado os recursos e os algoritmos utilizados no desenvolvimento desta dissertação. A terceira secção é apresentação explícita do implemetação bem como a justificação da tomada de certas decisões. A quarta secção conta com a apresentação dos resultados obtidos. A quinta secção é a conclusão onde se fará um apanhado sobre as conclusões retiradas desta dissertação bem com a exposição de futuros possíveis trabalhos que se podem realizar para melhor o projeto.

2

Estado da arte

Os estudos variam desde explorar a relação do estado psicológico do condutor, o seu desempenho em diferentes condições de estrada e diferentes condições climáticas [12], como uma análise mais orientada à classificação recorrendo a exemplos de referência pré-classificados[9, 14], outros optam por arranjar métricas estatísticas capazes de discriminar classes [34], também existindo outras técnicas que possibilitam a extração de características não estatísticas que permitem distinguir entre manobras e condutores [5, 31, 32].

Apesar da variedade de técnicas, todas levam ao mesmo objetivo: compreender em algum grau, o comportamento dos condutores e separá-los por classes ou colocá-los por ordem.

Este capítulo serve o objetivo de aprofundar, conhecer melhor as técnicas, recursos e algoritmos utilizados nas pesquisas mais relevantes deste domínio, entrando em detalhes para apresentar e compreender o uso de certos algoritmos.

2.1 Aquisição de dados

Nos últimos anos, algumas pesquisas publicadas em páginas de organizações populares como o Institute of Electrical and Electronics Engineers (IEEE) abordam várias técnicas, recursos e combinações de algoritmos para desenvolver modelos que contribuam para este tópico, mostrando diversidade de soluções possíveis.

Como estas pesquisas dependem de dados, é necessário saber-se que categoria de dados se está a analisar e a sua origem, consequentemente que opções de sensores existem para esse efeito.

2.1.1 Tipos de dados e de sensores

Os dados que normalmente são analisados encontram-se separadas em dois tipos: os dados orientados aos veículos e os orientados ao condutor.

Quando se analisa dados orientados ao veículo, existe uma distinção entre tipos de sensores que se pode usar, sendo essas categorias, sensores internos e sensores externos [28].

2.1.1.1 Dados orientados ao veículo

As abordagens à analise dos dados são diversas, desde o uso de critérios definidos manualmente, como definição de intervalos de valores para as diferentes classes, ao uso de *machine learning* que poderá deduzir esses mesmos intervalos sozinha. O tipo de dados envolvidos pode englobar Global Positioning System (GPS), velocidade, aceleração, ângulo do volante, pressão dos pedais, consumo do combustível, entre outras. Sobre os dados podem-se calcular propriedades estatísticas [34], aplicar técnicas de transformação dos dados, por exemplo, diminuição da dimensão, para simplificar a sua análise, podendo trazer mais benefícios [5, 18] ou inserir diretamente numa rede neuronal utilizada como classificador [28] entre outros métodos tão ou mais diversificados que estes.

Este tipo de abordagem, está dividido em duas categorias, dependendo do tipo de sensores que se esteja a utilizar para se fazer a análise: sensores internos e sensores externos.

Sensores internos Quando se fala de sensores internos, refere-se a equipamentos sensoriais embutidos no veículo, por exemplo, Controller Area Network (CAN) ou *black boxes*, com acesso a várias propriedades dos carros como: velocidade, ângulo de rotação do volante, pressão nos pedais, consumo de combustível, etc. A vantagem deste tipo de sensores é terem sido projetados para desempenhar essa função no contexto de veículos, trazendo confiabilidade em troca de apresentar custos altos de instalação, manutenção e inflexibilidade.

Sensores externos Quando se fala de sensores externos refere-se a dispositivos conectados ao veículo, que trazem a vantagem da flexibilidade em troca de menor confiabilidade do sinal recolhido.

Presentemente, com a popularização dos *smartphones*, que basicamente são telemóveis e computadores pessoais integrados no mesmo dispositivo, esta tecnologia chama a atenção como matéria de estudo neste domínio, pois é possível reduzir os custos associados à instalação e à manutenção sem perder significativamente a qualidade dos dados monitorizados, face aos custos [34], porque grande parte das pessoas usam-na no seu quotidiano o que significa que as empresas apenas têm que se preocupar com a parte relacionada com o *software*.

Os *smartphones* têm sido melhorados, o que leva ao uso dos seus sensores externos.

2.1.1.2 Dados orientados ao condutor

Os estudos que utilizam dados orientados ao condutor implicam métodos mais invasivos que os estudos que utilizam dados orientados aos veículos porque podem depender de sistemas de leitura de dados biométricos ou psicológicos pouco simples de serem obtidos [28]. Os dados captados podem ser ondas cerebrais através de sensores de eletroencefalografia [35], medidas cardíacas com eletrocardiograma, monitorização da posição da

cabeça e dos olhos para se compreender a relação que estes têm com a condução [12, 22, 35].

2.2 Estudos recorrendo a templates

Algumas das abordagens, como já mencionadas, recorrem a técnicas que dependem de um *template*, que serve de referência para se distinguir condutores entre classes ou distinguir manobras, entre outras características que possam ser separáveis. Esses estudos baseiam-se várias vezes em técnicas utilizadas para sistemas de reconhecimento de voz [27].

2.2.1 Classificação de comportamentos recorrendo ao *DTW* e ao *Bayes classifier*

O primeiro estudo a apresentar [9], aspira distinguir o comportamento na condução em duas classes, agressivo e não agressivo. Os dados foram recolhidos com o auxílio dos sensores de um *smartphone*, dos quais foram escolhidos o acelerómetro, o giroscópio e o magnetómetro, que são relevantes porque oferecem uma relação direta com propriedades do veículo como, velocidade, variação do ângulo relativamente à trajetória do veículo, etc. Para lidar com o ruído do sinal a técnica de *Moving Average* foi a que apresentou melhores resultados na obtenção de um sinal limpo mais idêntico ao original. O *Moving Average* é uma técnica que transforma um sinal noutro, mais limpo e suave, ou seja, com redução do ruído existente. Este resultado é atingido aplicando a média de uma quantidade finita de pontos consecutivos, normalmente centrada em volta do ponto que se pretende calcular a média, esta sequência de pontos finita comporta-se como uma janela deslizante, para se calcular todos os valores.

Após se ter obtido o sinal limpo, sem ruído, dois algoritmos são aplicados em seguida, o primeiro foi *end-point detection* e o segundo foi o Dynamic Time Warping (DTW). O *end-point detection* consiste em identificar o início e o fim de um potencial evento, possibilitando o seu isolamento do resto do sinal, que poderá trazer alguns benefícios dependendo do objetivo. O DTW permite comparar dois sinais de tamanhos diferentes de forma a minimizar a diferença entre os dois, é útil porque permite que dois eventos que aconteceram em espaços temporais diferentes possam ser comparados de uma forma relativamente justa. A porção isolada pelo *end-point detection* será comparada com todos os sinais previamente gravados e classificados denominada de *template*. O *template* que se assemelhar mais com o sinal isolado será o escolhido como referência para classificar o evento isolado.

Os eventos detetados neste sistema foram a direção do volante, aceleração, desaceleração e mudança de faixa.

Após todos os eventos terem sido classificado aplica-se a fase final que consiste em classificar os hábitos de condução, que neste caso foi escolhido para desempenhar essa função o *Bayes classificator*. O *Bayes classificator* é um algoritmo de aprendizagem

supervisionada, o que significa que este precisa de ser treinado com um conjunto de dados previamente classificado. Este conjunto de dados surge após a experiência prática ser realizada.

A experiência foi realizada reunindo um grupo de quinze pessoas, das quais cinco eram inesperientes, cinco eram experientes e os outros cinco foram completamente aleatórios. Dos cinco experientes, dois foram nomeados como avaliadores, o que significa que o papel destes, era avaliar a condução dos outros todos, inclusivamente um do outro, por um preenchimento de um questionário de modo a classificar como agressivo e não agressivo. Todos os condutores foram submetidos a *test drives*, duas vezes cada um, sob condições de estrada e climáticas diferentes para haver robustez na classificação. Do resultado das avaliações, os cinco melhores foram utilizados para obter os *templates*, que foram manualmente separadas e classificadas. Os dez restantes foram utilizados como conjunto de dados de treino para o *Bayes classificator*, dos quais dois foram considerados condutores agressivos, e oito como não agressivo.

Com o sistema completo, a precisão de acerto foi de 93,3%. Apesar do aparente sucesso deste sistema, não ter sido deixado claro quais foram os critérios utilizados para avaliar os condutores nem detalhes sobre as condições climáticas e de estrada que os condutores foram submetidos abre espaço a questões. Também devia ter ficado explicito se o conjunto de elementos de teste foram os mesmos que o conjunto de elementos de treino, para se esclarecer qualquer tipo de predisposição do sistema. É de notar também que a composição é relativamente pequena, o que poderá indicar que há uma baixa variedade de exemplos para representar o universo que se pretende modelar. Um lado positivo desta análise foi a remoção dos melhores condutores, assim diminuiu consideravelmente a diferença relativa entre as instâncias classificadas como agressivo e não agressivo.

2.2.2 Classificação das manobras recorrendo ao DTW

Outro estudo feito nesta linha de pensamento, resultou numa aplicação chamada *MIROAD*, que teve um objetivo final diferente. O objetivo foi a exploração da ideia de desenvolver um sistema capaz de distinguir entre eventos e classificá-los entre oito manobras distintas [14], sendo estas curva à direita, curva à esquerda, inversão de marcha em "U", estas manobras mas agressivas e não agressivas e mudança de faixa agressiva à esquerda e à direita. As mudanças de faixa não agressivas neste estudo não foram consideradas porque, relativamente aos sinais que geravam, eram muito parecidas com as curvas não agressivas.

O esquema do sistema utilizado foi parecido com o anterior descrito, o que mudou foi a técnica escolhida para se obter os dados, chamada *sensor fusion*, que permite explorar um conjunto de dados resultante da junção de dados de vários sensores. A vantagem deste método é que os dados combinados tornam-se mais ricos em informação, dando assim uma perceção mais completas, seguras e confiáveis dos dados.

Os conjunto resultante do *sensor fusion* foi $T = \{g_x, a_y, ex\}$, uma mistura de componentes dos sensores escolhidos para este estudo que são, $A = \{a_x, a_y, a_z\}$, $E = \{e_x, e_y, e_z\}$,

$G = \{g_x, g_y, g_z\}$, que representam o acelerómetro, o sensor de ângulo de rotação e o giroscópio, respetivamente.

Seguidamente foi aplicado o *end-point detection* e o DTW, que como no exemplo anterior, foram usados para isolar o potencial evento, compará-los com *templates* gravados e classificados previamente.

Como referência do que se espera de cada manobra, foram gravados cinco *templates* por manobra, e por cada sensor, que totaliza 120 *templates*. Cada manobra foi gravada utilizando o mesmo condutor e o mesmo veículo. Foi usado o K-Nearest Neighbor (K-NN) para com $k = 3$ para classificar o tipo de evento do conjunto T de *sensor fusion*.

Durante a experiência prática foram coletados 201 eventos, dos quais aproximadamente 50 foram considerados potencialmente agressivos, gerados intencionalmente para testar o sistema.

Esta experiência teve um resultado de 97% de precisão, além de conseguir distinguir entre inversão de marcha em "U"e curva à esquerda, com grande precisão, resultado este atingido devido à técnica de *sensor fusion* que não foi possível ser atingido com os outros sensores individualmente.

A Figura 2.1 mostra a diferença que o conjunto de sinais *T* faz a distinguir a manobra *U-turn* com a curva à esquerda, sendo este conjunto o único capaz de o fazer com clareza. A Figura 2.1 mostra a taxa de deteção dos padrões e mais uma vez o conjunto *T* é consideravelmente superior nesta função.

Taxa de deteção									
	R	L	U	HR	HL	HU	SR	SL	Total
A	72	68	23	71	100	100	100	83	77
G	76	63	46	71	100	100	100	83	79
T	92	83	77	100	100	100	100	83	91

Tabela 2.1: Tabela a comparar os resultados das taxas de deteção das manobras para os diferentes conjuntos de sinais em percentagem (%) [14]

Este resultado prova que o *sensor fusion* é superior em relação aos outros sensores individualmente no que diz respeito à distinção entre curva à esquerda e inversão de marcha em "U".

2.3 Estudos sem recurso a templates

Neste estudo [34] a contribuição para o assunto ofereceu uma forma de classificar os dados em agressivo e não agressivo através da exploração dos dados estatísticos, evitando assim a utilização de *templates*. Este método elimina a subjetividade da pré-classificação humana, que poderá ser uma mais-valia dependendo do contexto em que é aplicada.

A experiência foi realizada a partir de dados de uma empresa de transportes públicos, recolhidos a partir de 110 viagens sob a mesma rota e mesma condição climática.

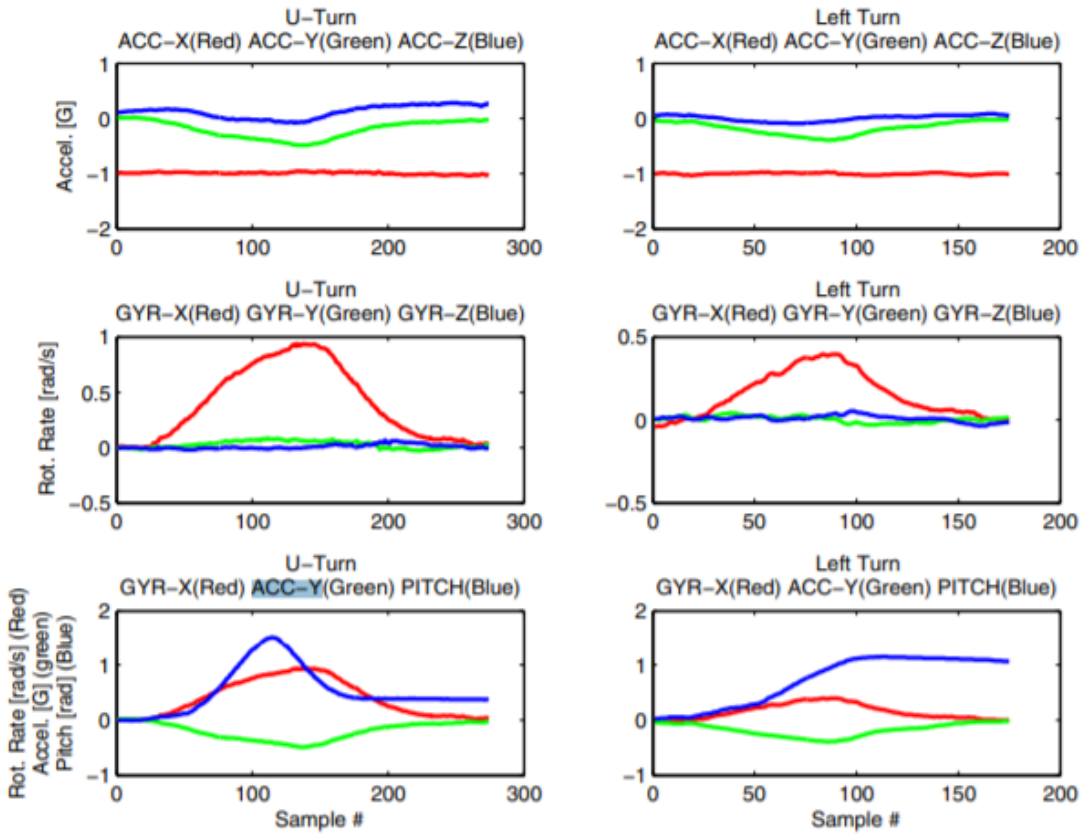

Figura 2.1: Exemplo a comparar os sinais coletados durante a manobra *U-turn* e curva à esquerda entre os conjunto de sinais *A*, *G* e *T* [14]

Isto oferece maior semelhança entre dados e por isso espera-se que haja uma melhor discriminação entre tipos de comportamento.

Os dados analisados foram os três eixos da aceleração, onde destes foram extraídas as seguintes propriedades estatísticas:

- Tendências centrais (média aritmétca, mediana, moda)
- Tendências de dispersão (variância, amplitude – interquartil, desvio da média, desvio absoluto da mediana, desvio padrão, amplitude total)
- máximo e mínimo
- características do histograma (assimetria, *kurtosis*);
- coeficiente de correlação tau de Kendall
- covariância
- critério de violação de limite de dados
- diferença entre os histogramas de direção de referência e comuns
- parâmetros polinomiais identificados após a aproximação do histograma de densidade cumulativa inversa

O critério de violação de limite de dados ou *DTV* do inglês *criteria of data threshold violation* (2.1) é calculado através da razão entre a quantidade de vezes que o sinal de referência passa um limite máximo e mínimo (N_1) sobre o tamanho do sinal inteiro (N).

$$DTV = \frac{N_1}{N} \tag{2.1}$$

Foi utilizado também como propriedade estatística a diferença entre a soma das diferenças absolutas das barras dos histogramas porque como na Fig. 2.2, nota-se que os histogramas têm diferentes distribuições, logo espera-se que as suas propriedades estatísticas tenham algum grau de discriminação.

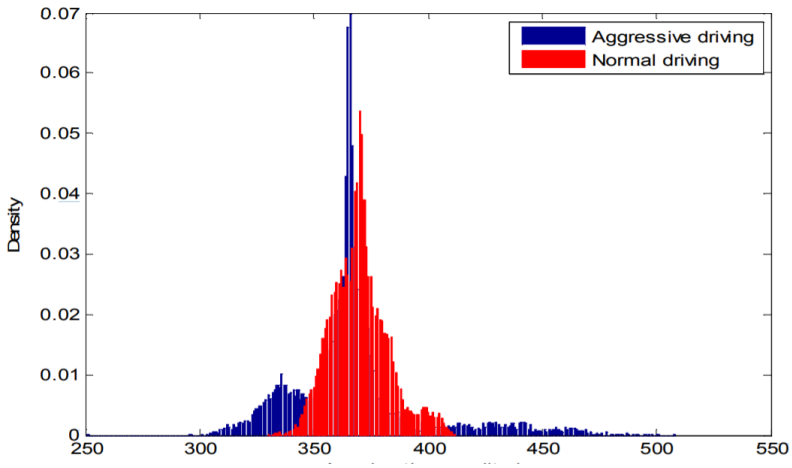

Figura 2.2: Diferença entre histograma das conduções normais e agressiva [34]

Também se utilizou uma aproximação polinomial da densidade cumulativa inversa do histograma, que os autores acreditam que os coeficientes da equação resultante são úteis também.

$$y = \sum_{n=0}^{n+1} p_{n+1-i} x^{n+1-i} \tag{2.2}$$

Contando com 39 propriedades por eixo, somando um total de 117 propriedades. Destas características, todas, as mais discriminatórias foram escolhidas recorrendo a dois algoritmos, o *Student-t test*, o Sequential forward feature selection (SFFS). O *Student-t test* é um algoritmo que consegue organizar as propriedades por ordem de discriminação. O SFFS consegue selecionar do melhor conjunto de propriedades mais discriminantes.

O resultado foi uma capacidade de acerto de 100% com apenas 7 propriedades:

- média da aceleração longitudinal;
- média da aceleração vertical;
- mediana da aceleração vertical;
- covariância entrea aceleração longitudinal e lateral;

- Três coeficientes polinomiais de 5ª ordem após aproximação de histograma de densidade cumulativa inversa

É importante referir que utilizando apenas média e a mediana é possível obter uma precisão 98%.

Estes resultados são interessantes porque traz confiança neste método, que discriminam com base em propriedades estatísticas caso sejam aplicados em sistemas onde se analisa dados recolhidos sob a mesma rota, condição de estrada e condição climática, apropriado para distinguir condutores que precisam de efetuar a mesma rota recorrentemente.

Um exemplo onde esta técnica poderia ser aplicada, seria em sistemas que pretendem avaliar a condução de condutores geral, mas o cálculo da condução é efetuada através da soma das avaliações por segmento de estrada, ou por rotas comuns.

2.4 Extração de padrões da condução

Outra abordagem, bastante criativa, recorre a métodos de extração símbolos de um sinal monitorizado, possíveis de serem interpretados como padrões que traduzem comportamentos diferentes, ou manobras diferentes, dependendo do contexto [5, 32, 31].

2.4.1 Ordenar padrões de comportamento por ocorrência

Um dos artigos que aplicou este conceito para a análise da condução[5], apresentou uma forma de comparar os condutores por ordem de qualidade de condução através da procura, no sinal de aceleração, informação relevante que evidencie hábitos diferentes de condução, que possivelmente representam comportamento agressivo ou não agressivo. Estes dados foram obtidos através do sinal de GPS recolhidos a partir de On-Board Devices (OBD) instalados nos táxis de uma empresa de táxis chinesa, de onde se calculou a velocidade e por sua vez a aceleração. O número de participantes foi de 124, dos quais cada um registou aproximadamente 480 horas.

O primeiro passo nesta abordagem é normalização os dados. O método escolhido foi *Z-standardization* ou também chamado *Z-score*, que obedece à expressão 2.3 que permite comparar valores numa escala padrão. Esta escala padrão é obtida transformando todos os valores originais (x) de forma a que estes novos valores tenham um média (μ) igual a 0 e um desvio padrão (σ) igual a 1. A razão da sua escolha reside principalmente por ser robusto a dados atípicos.

$$x^* = \frac{x - \mu}{\sigma} \tag{2.3}$$

O segundo passo é tratar o sinal normalizado num sinal adequado a ser trabalhado para este propósito, para isso utilizou-se o algoritmo de SAX [33] que transforma um sinal discreto numa sequência de caracteres, diminuído a dimensão dos dados recolhidos o que poderá trazer vantagens ao nível de processamento por parte dos algoritmos mantendo

os resultados ou sem haver perdas significativas, como mostra a Figura 2.4. Este objetivo é atingido representando cada segmento num único valor a que denominada de Piecewise Aggregate Approximation (PAA), normalmente a média, mas poderia ser qualquer outra propriedade estatística, e a seguir converter o PAA numa sequência de caracteres.

Para se representar PAA é preciso decidir em quantos níveis se representa o sinal original. O número de níveis possíveis escolhido para representar uma porção do sinal foi de sete, além de ser ímpar, o que permite uma representação de neutralidade do sinal, ou seja, quando uma porção do sinal pode ser resumida ao valor que está à mesma distância que o valor máximo e o mínimo, também pelo facto de entre outros números ímpares, foi o que obteve melhores resultados. No algoritmo original, o resultado é apresentado em caracteres alfabéticos, o que significa que a representação do sinal em caracteres inclui, caracteres de "a"a "g". Para simplificar a interpretação e a computação, foram escolhidos caracteres numéricos, ou seja, a representação abrange números de 0 a 6. O número de padrões diferentes que podem aparecer são de $7^3 = 343$, para padrões que indicam uma intenção de aceleração e de desaceleração são $4^3 - 1 = 63$ para cada.

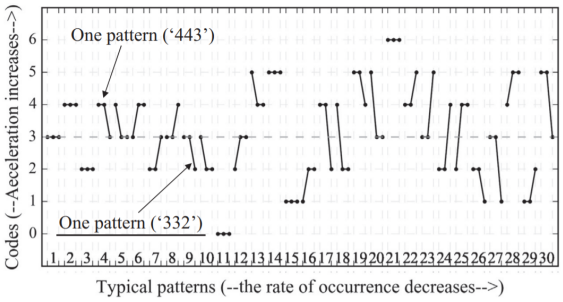

Figura 2.3: Exemplo ilustrativo de *ranking* de padrões[5]

O terceiro passo é detetar o padrão. Como três segundos são considerados o necessário para estabilizar o veículo relativamente ao contexto, três símbolos são considerados um padrão de condução. Defini-se como intenção de aceleração um padrão que seja a combinação de três símbolos que representam aceleração ou neutro, a mesma lógica se aplica à desaceleração.

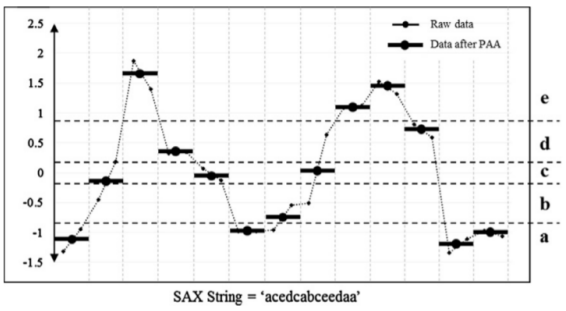

Fig. 1. Detail of SAX.

Figura 2.4: Exemplo ilustrativa do SAX [5]

Como os dados são lidos continuamente, os padrões são detetados numa janela

deslizante de tamanho três, ou seja, os três últimos símbolos detetados são registados como um novo padrão lido. Dos padrões extraídos, os mais comuns são utilizados para definir os hábitos de cada condutor. É feita uma contagem de padrões, dos quais os trinta melhores são ordenados num gráfico que pode ser traduzido como a representação dos hábitos de condução de cada condutor, como consta na Figura 2.3. É também feita uma avaliação quantitativa, ou seja, é calculada a pontuação de cada condutor com base nas trinta manobras mais frequentes 2.4.

$$SCORE = \sum_{i=1}^{30} freq_i * \{abs[mean(CODE) - 3] + 0.5 * std(CODE)\} \tag{2.4}$$

Se se obtiver dados suficientes para que se tenha informações sobre a condução sob diferentes condições climáticas, tipos de estrada diferentes e fatores externos, torna a conclusão mais robusta à subjetividade, tornado esta análise valiosa a estudos mais complexos.

Este estudo apresenta uma maneira de comparar condutores através da comparação dos gráficos dos padrões mais frequentes de cada um, e apresenta também valor que representa a pontuação dada a cada condutor em termos de agressividade.

Algumas observações interessantes neste estudo indicam que as desacelerações acentuadas são mais representativas de um comportamento inadequado do que as acelerações agressivas, a evidência encontra-se no facto de que quanto mais um condutor é considerado inadequado, ou seja, um condutor com frequências altas para padrões que são considerados inadequados, pior colocada fica o padrão que representa a aceleração mais acentuada, já em relação ao padrão que representa a pior desaceleração, este fica melhor colocado com a diminuição da qualidade da condução.

2.4.2 Organizar padrões de comportamento por semelhança

O estudo [32] procura desenvolver uma forma de organizar os padrões recorrendo à aprendizagem não supervisionada.

O objetivo é atingido primeiramente aplicando um algoritmo parecido ao SAX, mas com a diferença de que este consegue segmentar o sinal original em pedaços de tamanho variável dependendo do comportamento do sinal, chamado Variable Symbolic Aggregate Approximation (VSAX). Seguidamente propõem o agrupamento de padrões parecidos, por um algoritmo chamado DTW-SOM [31], que como o nome indica é uma combinação do DTW [2] com o Self-Organizing Map (SOM) [16]. O SOM é um algoritmo de aprendizagem não supervisionada com objetivo de agrupar itens que sejam semelhantes bidimensionalmente, orientada à análise visual. Também são exploradas outras técnicas de visualização de resultados como as *U-matrix* e as *Winner matrix*, cuja primeira evidência as semelhanças entre *clusters* adjacentes e a segunda contabiliza quantas das instâncias presentes no *dataset* pertencem a cada *cluster*.

Os dados analisados foram obtidos através do *dataset* chamado *UAH-Driverset* [26]. Os dados escolhidos para trabalhar este estudo foram a aceleração lateral e longitudinal.

A partir de um dos condutores do *dataset*[26], do qual foram detetato 281 padrões, inicializou-se o modelo DTW-SOM que organizou os padrões em *clusters* numa matriz de cinco por cinco, como mostra a Figura 2.5. É relevante notar que esses *clustes* estabelecem uma relação de semelhança com os *clusters* adjacentes.

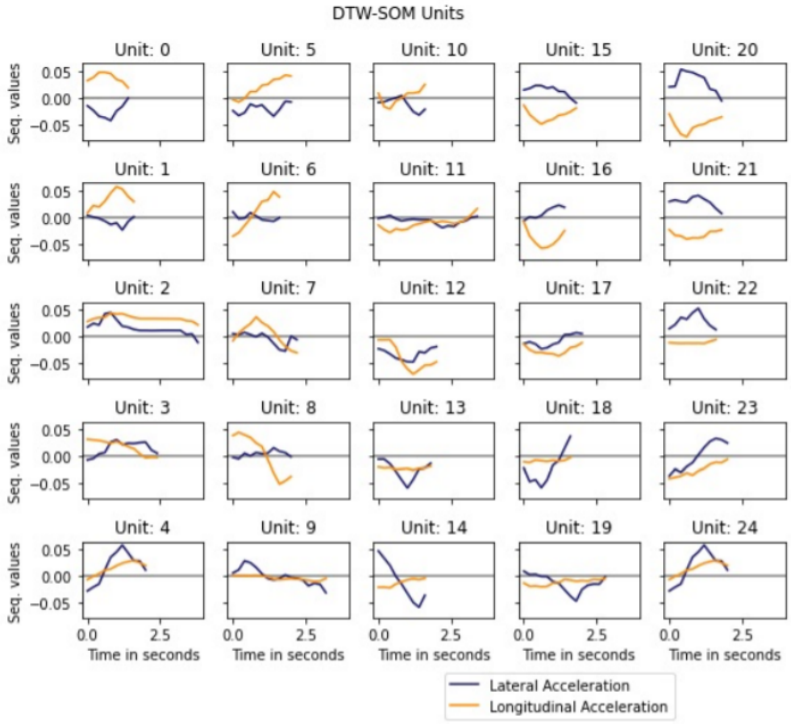

Figura 2.5: Matrix resultante do DTW-SOM [32]

O resultado mostra haver uma tendência para alguns *clusters* acumularem mais padrões que dizem respeito a um tipo de comportamento específico que outros, como consta na Figura 2.6, onde cada matriz exibe a distribuição dos padrões identificados em viagens com diferentes classificações, representando os padrões mais frequentes com uma cor mais intensa. Esta conclusão pode ser usada para se desenvolver um classificador para novos padrões nunca vistos.

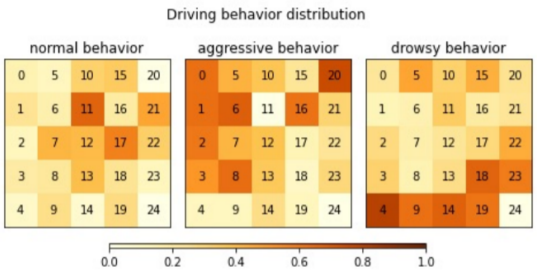

Figura 2.6: Distribuição das instâncias pelos clusters[32]

2.5 Redes Neuronais

A próxima abordagem, recorre ao uso de diferentes redes neuronais. Estas são estruturas compostas por diferentes nós, organizados de uma forma específica denomidada de modelo, com diferentes benefícios dependendo do contexto a ser aplicado.

Além de evitar o uso de *templates*, também evita o uso de limites entre classes definidos manualmente, por exemplo, recorrendo a especialistas.

2.5.1 Long-short term memory

O primeiro estudo que se descreve [28], recorrendo um tipo de rede neuronal que é uma variação da arquitetura chamada RNN, essa variante é designada por Long-short term memory (LSTM).

As RNN são redes cuja estrutura permite que a informação flua para os próprios estados ou estados anteriores, esta propriedade concede em algum grau a capacidade de memória, levando a resultados bastante positivos quando se lida com dados de caráter temporal, pois é possível analisar dados tendo em conta algum contexto relativamente aos dados que foram passados.

O LSTM são um tipo de RNN que acrescenta muito valor a áreas em problemas que necessitam de processar informação que dependem de uma longa contextualização, por exemplo, reconhecimento de fala [13].

Recorrendo às técnicas de *sensor fusion*, este artigo usa os seguintes dados:

- Acelerómetro de três eixos
- giroscópio
- sensor GPS (velocidade)
- câmara (número de carros à frente, distância do carro da frente)

A ideia é utilizar dados que estão disponíveis em vários veículos atuais, visto que este muitos vêm equipados com sensor GPS, sensores de inércia e câmaras, e a tendência é que mais venham. Como já mencionado, em vez de processar os dados para extrair alguma propriedade, ou diminuir a sua dimensão, os dados serão diretamente injetados na rede neuronal. O resultado esperado é obter um sistema capaz de classificar o comportamento do condutor em tempo real, em três categorias diferentes, normal, agressivo ou sonolento.

Para treinar, validar e testar a rede, utilizou-se o *dataset UAH-Driverset* [26], que contém os dados necessários para a realização desta experiência. É de notar que foram registados mais atributos do que a experiência precisa, além de que a frequência de amostragem dos sensores não serem todas a mesma, o que significa que foi necessária uma preparação de dados para garantir os atributos necessários e que estes estivessem sincronizados, esta preparação consiste primeiramente em fazer *resampling* da base dados eliminando os atributos a mais, deixando apenas aqueles necessários, bem como a seleção das instâncias que contenham a mesma *timestamp* para a sincronização de dados.

Após o último passo aplica-se uma normalização dos dados (2.3) para que estes possam ser medidos na mesma escala. Após o *resampling* e a normalização foi aplicado no que resultou destas a técnica de *rolling window* de tamanho 64 com uma sobreposição de 50%. Esta técnica consiste em sequencialmente introduzir na rede neuronal os últimos 64 dados da base dados, sendo que a cada conjunto de dados introduzido, 50% foi introduzido no ciclo anterior. Esta técnica traz vantagens, como o aumento relativamente à quantidade de dados utilizado, como também permite a classificação em tempo real, porque cada conjunto de dados será classificado individualmente.

O tamanho do *dataset* foi de 9499 instâncias, que foram separadas em 70% para conjunto de treino e 30% para conjunto teste, feito com *3-fold cross validation*.

Os resultados desta experiência foram comparados utilizando a métrica *F1 measure*. Esta métrica combina a precisão e com o *recall*, fornecendo uma perspetiva geral do desempenho do classificador. A precisão é uma métrica que estuda a relação das instâncias positivas corretamente acertados relativamente ao número de instâncias classificadas como positivas, independentemente de terem sido corretamente classificadas. O *recall* é a métrica que estuda a relação das instâncias positivas corretamente acertados relativamente ao número de instâncias que foram corretamente classificadas.

2.5.2 Gated Recurrent unit

O estudo [29], combinou a classificação recorrendo a neural networks e também com o recurso à definição de regras. Ambos os métodos contribuem para o mesmo objetivo, a classificação dos dados em duas categorias, normal e agressivo. Os dados utilizados foram a aceleração de três eixos e a velocidade e a saída era na forma de histograma que representava a frequência por ocorrência. Os dois histogramas foram combinados num só.

Foram consideradas duas RNN diferentes, a LSTM e a Gated Recurrent unit (GRU). A escolhida foi GRU, porque em comparação com LSTM teve menos tempo de treino apesar de terem atingido praticamente a mesma precisão de acertos. Foram utilizadas, uma camada de 128 neurónios e uma *hidden layer* de dois neurónios com a função de ativação *softmax*. Para o treino utilizaram-se janelas temporais, tal que cada janela tinha como classe aquela que correspondia à classe das instâncias que as compunham, por exemplo, uma janela composta por instâncias de comportamento agressivo seria classificada como uma janela agressiva. O objetivo é ensinar a rede a classificar uma sequência de dados nunca vistos. O histograma resultante seria um composto com a frequência de janelas que correspondem a comportamentos normais e agressivos.

Para a definição para a deteção de eventos foi utilizado descrito em [3], que distingue os eventos em aceleração, travagem e curva em três níveis diferentes, baixo, médio e alto. O resultado desta análise será um histograma com a frequência por tipo de evento.

Os dois histogramas serão fundidos e normalizados através da divisão pela duração da viagem, para a seguir serem utilizados como entrada de um perceptrão padrão que

irá decidir se o comportamento do condutor é agressivo ou normal. Um perceptrão é um algoritmo utilizado para classificação supervisionada binária.

Os resultados deste estudo indicam que o método híbrido supera o método por regras e o método da rede neuronal individualmente.

	RNN	Regras	Híbrido
Precisão nos comportamentos normais	11/12	**12/12**	**12/12**
Precisão nos comportamentos agressivos	6/11	8/11	**10/12**
Precisão geral	17/23	20/23	**22/23**

Tabela 2.2: Tabela com os resultados a comparar os métodos híbrido, regras e RNN [29]

2.6 Machine learning

ML é uma técnica considerada um subconjunto de técnicas chamada de Artificial Intelligence (AI) onde esta última é definida como a incorporação da inteligência humana nas máquinas. A ML pode ser visto como as técnicas que pertimem que se obtenha esse fenómeno que se pareça com a inteligência humana. Existem várias defições disponíveis na literatura [7], [21] sendo assim um conceito difícil de definir e portanto difícil de se achar um consenso.

A ideia por detrás da ML é de permitir que as máquinas aprendam de forma autónoma, isto é, sem que estas necessitem de que seja expressa as especificações da tarefa. A aprendizagem é então baseada na observação de exemplos que se deseja modelar.

Normalmente, se os problemas forem abordados através de algortimos clássicos, é possível criar um algoritmo baseado em regras para determinar se um condutor é agressivo ou não, aproveitando o contexto desta dissertação, com base em critérios como velocidade média, frequência de aceleração e desaceleração bruscas, e histórico de infrações de trânsito, por exemplo. É uma tarefa importante para companhias de seguros automóveis para avaliar o risco dos clientes e assim determinar os valores dos seguros.

Por outro lado, a ML oferece uma abordagem mais sofisticada para essa tarefa. Ao fornecer ao modelo dados de condução históricos de uma grande amostra de condutores, juntamente com informações sobre quais desses condutores foram envolvidos em acidentes ou receberam infrações de trânsito, o modelo pode aprender padrões sutis que distinguem condutores agressivos de não agressivos.

Por exemplo, o modelo pode descobrir que condutores com uma alta frequência de acelerações bruscas e desacelerações estão mais propensos a se envolver em acidentes. Ele pode também identificar padrões de comportamento de condução que indicam cautela, como manter uma velocidade constante e respeitar os limites de velocidade.

Ao analisar esses padrões, o modelo pode classificar novos condutores como agressivos ou não agressivos com base em seu comportamento de condução, ajudando assim a

companhia de seguros a avaliar o risco associado a cada condutor de forma mais precisa e justa.

Com a ajuda de ML é possível construir modelos que conseguem captar padrões que os olhos do ser humano não são capazes de captar, mas não garante a perfeição da tafera a desempenhar, existindo normalmente sempre métricas de avaliação como a precisão, por exemplo [7].

O desempenho do modelo de ML está relacionada com a qualidade e a quantidade de dados [21]. Logo para que altos desempenhos sejam atingidos é necessário que o *dataset* esteja bem classificado e com exemplos suficientes para que o modelo seja capaz de generalizar.

É uma técnica que depende de dados e pode ser aplicado a qualquer contexto por exemplo, física, biologia, economia, entre outros. As suas aplicações são diversas desde Natural Language Processing (NLP), *speech recognition*, *language modeling*, *computer vision application* entre outros [21]. As aplicações aumentam à medida que a ML vai se desenvolvendo.

A questão que deve ser feita quando se desenvolve estes modelos é como se desempenham frente a dados nunca antes vistos, ou seja, dados não presentes no conjunto de treino. É importante reconhecer que o desempenho do modelo no conjunto de treino não reflete o desempenho em dados nunca antes visto [17]. Em casos onde o modelo tem altos desempenhos no conjunto de treino e baixo desempenho em lidar com novos dados, este fenómeno se chama de *overfitting*. Estes fenómenos acontecem quando o modelo é incapaz de generalizar. Para isso algumas soluções podem ser exploradas, como divisão do *dataset* em conjunto de treino, conjunto de teste e possívelmente em conjunto de validação. Também é possível aplicar *crossvalidation* que itertivamente utiliza uma parte do *dataset* como conjunto de teste e o resto como conjunto de teste, sendo que a cada iteração o conjunto de treino vai sendo trocado até que todas as partes sejas usadas como conjunto de treino [7].

Normalmente o ML está dividida em duas categorias, a *supervised learning* que consiste numa análise de dados que estão previamente classificados e a *unsupervised learning* que consiste numa análise de dados onde não existe classificação prévia. Exemplos de *supervised learning* são os problemas de classificação onde o modelo tenta prever a classe de novas instâncias, e problemas de regressão onde o modelo tenta prever valores numéricos. Já exemplos de *unsupervised learning* são problemas de *clustering* onde o modelo aprende a agrupar dados em função da sua similaridade, e de redução de dimensionalidade, onde o objetivo é reduzir a complexidade dos dados [21].

2.6.1 Supervised learning

No *supervised learning* o *dataset* é composto por um par de informação *input-output*, cujo o *input* corresponde aos valores que caracterizam uma instâncias e o *output* o valores de definem a classe a que este pertence. Estes par de valores que permite que a máquina avalie o seu desempenho ao longo do processo de aprendizagem, comparando o resultado

das suas previsões com os valores reais do *dataset* [7]. O modelo será treinado feito sobre os exemplos disponíveis, onde este irá sempre procurar melhorar a sua precisão das suas previsões comparado aos mesmos exemplos.

Como já foi referido são exemplos de problemas do tipo de *supervised learning*, problemas de classificação e de regressão. No primeiro caso como o próprio nome indica o objetivo é classificar corretamente dados, principalmente dados não antes vistos, dentro de um conjunto finito de valores utilizados para classificar esses mesmos dados. Para ilustrar o segundo caso pode-se utilizar o exemplo de um modelo que tente prever a temperatura baseadas em condições climáticas, neste caso os valores a considerar deixam de ser categóricos e passam a ser número real.

Existem vários exemplos na literatura, como por exemplo, onde *Supervised ML* foi usado para classificar pacientes com linfomas [30]. Outro exemplo foi um modelo desenvolvido para classificar a viabilidade de um projeto de construção [15] baseado em dados utilizado para análise de risco de projetos de construção. Para ilustar os problemas de regressão foi feito um modelo que efetua a previsão do hidrogênio limpo produzido pela gaseificação de biomassa [23].

2.6.2 Unsupervised learning

O objetivo da abordagem através de *unsupervised learning* é de aprender sobre a estrutura dos dados nos quais o modelo de ML irá trabalhar, sem que esses mesmos dados tenham sido classificados *labels*, ou seja, só há dados de entrada [7]. O modelo deverá ser capaz de encontrar padrões. Está sujeito a interpretações visto que não há forma objetiva de avaliar desempenho.

A maioria algoritmos estão separados por *clustering* e redução de dimensionalidade. Para ilustrar o primeiro, suponha-se que se quer categorizar consumidores de uma economia em função dos seus hábitos de compra, como não é possivel simplesmente atribuir classes a cada individuo, seria necessário ser aplicado um algoritmo *unsupervised clustering*.

O objetivo da abordagem através de redução de dimensionalidade é de diminuir o tamanho a complexidade dos dados do *dataset*, eliminando dados menos relevantes, unindo vários num só ou apenas transformando a forma como os dados são interpretados ou processados pelo modelo. Esta abordagem está associada a duas técnicas, a *feature selection* e a *feature extration* [7]. A primeira consiste na eliminação das características menos relevantes mantendo as mais relevantes, já a segunda consiste na criação de características novas a partir das características originais [25]. A redução da dimensionalidade é benéfica para eliminar informação redundante, positivo para o desempenho do modelo. É pertinente realçar que a redução da dimensionalidade pode ser utilizada nos dados de entrada em problemas de *supervised learning*.

Para ilustrar a aplicação de *unsupervised learning* existem aplicações onde se pretende agrupar pacientes baseado nas caracteristicas genéticas [20].

3

Metodologia

Neste capítulo, são apresentados e explicados os métodos e recursos escolhidos na abordagem do problema proposto nesta dissertação.

Na primeira secção será arbordado de onde surgiu a inspiração da metodologia utilizada, ou seja, como se pretende abordar o problema de forma geral.

Na segunda secção será apresentado o método de coleta de dados escolhido, destacando as suas vantagens e limitações.

Na terceira secção serão explorados os processos de processamento, que incluem técnicas como a remoção de ruído, normalização e a aplicação do um algoritmo chamado SAX, que será utilizado na transformação das séries temporais numa sequência de símbolos que representam a mesma série temporal numa dimensionalidade reduzida, que será o objeto de análise.

Na quarta etapa, será abordado os métodos e algoritmos utilizados para analisar os dados em si, nomeadamente as árvores de decisão numa tentativa de abordar o problema de um ponto de vista supervisionado.

3.1 Inspiração

Toda a metodologia realizada nesta dissertação foi inspirada no trabalho realizado no artigo entitulado de "*A graphical modeling method for individual driving behavior and its application in driving safety analysis using GPS data*"cujo os autores são Chen Chen, Xiaohua Zhao, Yunlong Zhang, Jian Rong e Xiaoming Liu [5], que desenvoveram um processo de que tinha como objetivo a ordenação dos condutores por agressividade. Esse objetivo foi alcaçado recorrendo à equação 2.4, que calcula e atribui a cada viagem uma pontuação que representa a avaliação da agressividade na condução dos condutores. A atribuição dessa pontuação é baseada nos padrões e a sua ocorrência relativamente ao número total de padrões, em percentagem, de cada condutor.

A ideia proposta nesta dissertação é de que, aplicando o método utilizado no artigo mencionado [5], de utiliar o *ranking* dos padrões mais frequêntes condutores, assumindo que em alguma medida expressa o comportamento do condutor, e utilizar essa informação

para analisar as viagens disponíveis. Em vez de simplesmente ordenar cada condutor considerando a equação 2.4, recorrer a esse mesma equação para classificar os dados disponíveis para análise que possa proceder a uma análise supervisionada, que envolvem técnicas de ML nomeadamente modelos do tipo de àrvores de decisão para assim se poder obter conclusões.

A expectativa é de que desta análise seja possível modelar viagens com diferentes classes baseada nos padrões presentes no *ranking*, mostrando que é plausível assumir que para qualquer *dataset* com o *rankings* de viagens ou de condutores e para qualquer critério divisão do *dataset* recorrendo ao *score* é possivel obter um desempenho suficientemente bom, que se traduza na viabilização deste método na integração desta solução em processos mais completos, como por exemplo, a gestão de seguradoras automóveis.

3.2 Origem dos dados

A realização desta dissertação exigiu a coleta de dados, estes dados foram obtidos de fontes diferentes e também são de naturezas diferentes, desde corridas feitas a pé, viagens de carro ou de bicicletas, que podem ser todas consideradas porque nesta dissertação fundamentalmente estamos interessados em analisar a aceleração e não a velocidade. Esta tarefa mostrou-se desafiadora devido à ausência recursos prontamente disponíveis e da necessidade de colaboração de participantes coletassem dados. Diante dessa dificuldade, foi necessário pensar numa estratégia para coletar e armazenar informações que pudessem ser utilizadas em análises futuras. Além disso, diante da importância de dispor de uma quantidade substancial de dados, foram consideradas fontes adicionais, com dados relacionados a viagens de bicicletas.

As fontes de dados utilizadas apresentam origens distintas. A primeira abordagem envolve a criação de um sistema escalável, capaz de lidar com o aumento do número de utilizadores e, consequentemente, com o aumento da quantidade de dados em circulação. Esse sistema é composto por uma aplicação *Android* e serviços *Google Cloud*, com vista a garantia da eficiência na coleta e no armazenamento dos dados GPS, além de assegurar a sua disponibilidade para análises posteriores.

Por outro lado, a segunda abordagem baseia-se na utilização de um conjunto de dados denominado *EcoSense dataset* [8]. Esse conjunto de dados consiste numa coleção de viagens de bicicletas, originalmente coletadas para um projeto académico desenvolvido pelo Departamento de Sistemas de Informação da Universidade de Oldemburgo, na Alemanha, em parceria com a empresa "CoSynth GmbH & Co. KG", uma empresa focada no desenvolimento de ferramentas e serviços para *design* de *hardware* e *software* integrado. A inclusão desse *dataset* amplia a diversidade das informações disponíveis, possibilitando uma análise mais abrangente e enriquecedora.

3.2.1 Coleta de dados *pipeline*

O objetivo primordial do sistema de coleta de dados é de garantir uma utilização simples e uma manutenção fácil, ao mesmo tempo em que possibilitava escalabilidade para futuras melhorias no projeto. Levando em conta que a solução deveria ser de utilização fácil de compreender, de utilizar e eficiente, a intenção é de criar uma solução que não exigisse preparação prévia por parte dos utilizadores, bastando apenas que estes tivessem acesso a um *smartphone* durante as suas viagens. A integração com os serviços *Google Cloud* foi uma escolha estratégica, visando fornecer uma base sólida para a implementação de novas funcionalidades integradas em *Cloud*, facilitando assim trabalhos futuros.

A *pipeline* criada, consiste numa aplicação desenvolvida para *Android*, integrado com serviços de *GoogleCloud* para a transmissão e armazenamento dos dados, como demonstrado na Figura 3.1. Dos serviços *GoogleCloud* disponíveis na plataforma *Google*, os utilizados foram os serviços *PubSub*, uma *Cloud Function* e serviços *BigQuery*.

Neste contexto a aplicação *Android* serve de ponto de entrada do sistema. O serviço *PubSub* permite que comunicação entre os diferentes dispositivos e os serviços *Cloud* sejam realizadas de forma assíncrona, a *Cloud Funtion* serve o propósito de processamento dos dados recebidos pelo *PubSub* e inserção no *BigQuery*, e este último será usado como local de armazenamento.

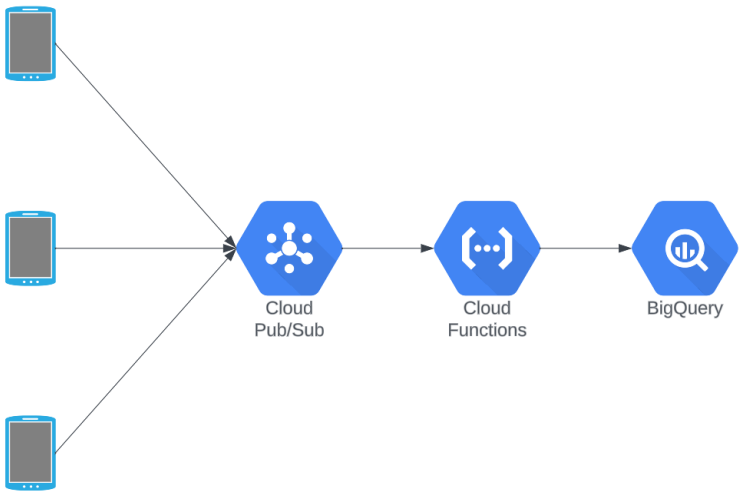

Figura 3.1: Esquema da *pipeline*.

3.2.1.1 Android

A tecnologia escolhida para esta dissertação foi o *Android*, embora também pudesse ter sido utilizado o *iOS* outro qualquer compatível com *smartphones*, pois é fundamental que se aproveite as suas características vantajosas. Essa características são vantagens que os *smartphones* oferecem são por exemplo, a ampla disponibilidade no mercado, acesso à *internet*, e uma variedade de sensores integrados.

Além disso, os *smartphones* possibilitam o desenvolvimento de aplicações sofisticadas

que podem coletar e enviar dados em tempo real. A integração desses recursos é relativamente simples, e a facilidade de uso dos dispositivos é uma vantagem adicional. Vale a pena ressaltar que os *smartphones* são uma tecnologia em constante evolução, com uma ampla gama de recursos e conteúdo online disponível para auxiliar no desenvolvimento de aplicações entre outras vantagens.

Para assegurar a qualidade no desempenho da aplicação *Android* em termos de precisão, frequência de amostragem, gestão de energia e fluidez, optou-se por adaptar uma aplicação de código aberto com essa finalidade o *BasicAirData GPS Logger* [10]. Esta escolha foi motivada pelo facto de que esta aplicação atende a todos esses requisitos, além de oferecer a vantagem adicional de permitir o reaproveitamento gratuito do código fonte.

O *BasicAirData GPS Logger* é uma aplicação básica, leve e focada em precisão e economia de bateria. Funciona totalmente *offline* e regista os dados tanto em modo de serviço *foreground* quanto *background*, o que permite a utilização simultânea de outras funcionalidades do dispositivo sem interferir na captura dos dados.

Foi necessário efetuar uma alteração do código para possibilitar a integração da aplicação com os serviços *GoogleCloud*. Essa alteração consiste no envio das viagens gravadas, para um tópico do serviço *PubSub*, pertmitindo que a informação recolhido chegue ao resto da *pipeline*. A vantagem desta implementação é de que possibilita que vários utilizadores possam utilizar a aplicação independentemente uns dos outros, cada um no seu dispositivo, e enviar as suas viagens para o mesmo local de forma simples, assíncrona, facilitando a recolha das mesmas num local centralizado.

A aplicação é simples, basicamente grava viagens em tempo real, e após viagem finalizada, permite exportá-las para uma diretoria local.

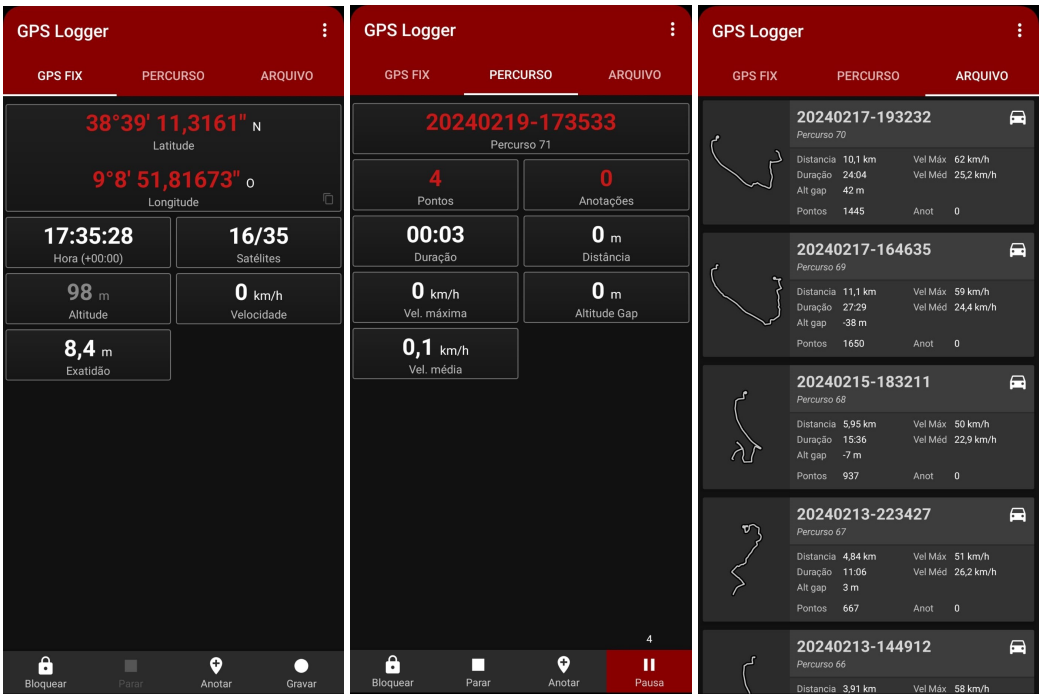

Figura 3.2: Imagens das diferentes janelas da aplicação

3.2.1.2 Serviços Google Cloud

Os serviços da *GoogleCloud* foram escolhidos devido à sua escalabilidade automática de recursos em função da demanda exigida pelo sistema implementado, que dá um grau de garantia na questão do desempenho. O facto de todos os serviços da *GoogleCloud* operarem no mesmo ambiente também facilita a integração de forma cómoda, além da segurança garantida pela a *Google*. Os serviços da *Google* utilizados foram:

- **PubSub** : é um serviço de envio e receção de mensagens assíncronas que serve de intermediário entre módulos que produzem mensagens e os módulos que as recebem, por meio de tópicos. Os que enviam as mensagens publicam no tópico e os que recebem subscrevem. Esta estrutura permite que se faça uma gestão dos tópicos não havendo a necessidade de se ter de publicar nem subscrever a todos os tópicos. O serviço *PubSub* foi escolhido para ocupar a primeira camada de integração entre a aplicação e o resto dos serviços *Cloud*, principalmente pela capacidade de centralização, isto é, para ser utilizado por vários utilizadores ao mesmo tempo, em que cada um a coleta e envia dados do seu *smartphone*, naturalmente de locais e em momentos diferentes, permitindo que os dados sejam enviados para o mesmo local de forma assíncrona em tempo real com baixa latência.

- **Cloud function** : é um ambiente de criação e de execução de serviços simples e com propósito específico que estão ligadas diretamente a eventos que aconteçam dentro da mesma infraestrutura *Cloud*. Isto significa que este serviço tem a capacidade de responder a eventos, como o upload de um arquivo feito no *Cloud Storage*, ou a chegada de uma mensagem no *PubSub* através de um tópico específico. É *serverless* o que significa que não é necessário nenhum tipo de preocupação de configurações *backend*. A escolha deste serviço como elemento de integração deste projeto deve-se ao seu potencial de escalabilidade, pois os recursos, configurações de *software* e manutenções necessárias são geridas pela *Google*, bem como a sua simplicidade.

- **BigQuery** : é um *data warehouse* projetado para armazenar e analisar grandes quantidades de dados de forma rápida e eficiente com recursos incorporados, como ML. Oferece a vantagem na questão da eficiência, escalabilidade e facilidade na integração. Incluí vários recursos, como por exemplo suporte para vários tipos de dados, análise *SQL* com queries complexas e *streaming de dados*.

3.2.2 *Dataset ECOSense*

Devido à falta de dados, fruto da fraca adesão ao uso da aplicação, recorreu-se a um repositório de um projeto da "Universidade de Oldemburgo do Departamento de Sistemas de Informação VLBA"chamado *ECOSense*. Esse projeto envolveu a participação de aproximadamente 400 voluntários e teve uma duração que compreendende o período de 01/06/2019 a 31/08/2020.

Este projeto tinha como objetivo a coleta de dados de condutores de bicicletas por meio de sensores, para apoiar a otimização da infraestrutura relativas a bicicletas, para atender à necessidade dos ciclistas e aumentar consequentemente a aderência e, ao mesmo tempo, lidar com problemas de trânsito, falta de estacionamento, ruído e poluição nos centros das cidades, além de questões relacionadas à saúde também poderem se beneficiar destas melhorias.

O *dataset* proveniente do projeto *ECOSense* foi considerado devido à disponibilidade de dados sobre a velocidade e o instante em que essa informação foi registada. Vale ressaltar que esses dados não foram recolhidos por *smartphones*, mas ainda assim fornecem informações valiosas para a análise pretendida, não invalidando as possíveis conclusões obtidas.

Este conjunto de dados consiste em 1281 viagens que precisam ser examinadas com mais detalhes para identificar possíveis irregularidades. Nesse sentido será feita uma análise onde se verifique a plausabilidados dos dados atráves da possível ausência de dados, *outliars* etc.

3.3 Processamento dos dados

O processo de análise de dados foi feita recorrendo à linguagem *python* e às suas bibliotecas de análise como *pandas*, *numpy*, *scipy*, o *scikit-learn*. Além destas bibliotecas também se utilizou a biblioteca *saxpy* para o processamento das séries temporais e o *matplotlib* para a visualização dos dados e dos gráficos.

- **Pandas**: é uma biblioteca de *python* muito utilizada para manipulação e análise de dados. Fornece estruturas de dados flexíveis, como o *DataFrame*, que é uma estrutura de dados em forma de tabela bidimensional, ou seja, composto por linhas e colunas. Além disso, o *pandas* oferece funcionalidades para leitura e escrita de dados em vários formatos *(CSV, Excel, SQL, etc.)*, limpeza e transformação de dados, agregação e agrupamento de dados, além de operações eficientes de indexação e seleção de dados. É uma ferramenta essencial para qualquer projeto que necessite de trabalhar com análise de dados em *python*.

- **NumPy**: é uma biblioteca fundamental para computação científica em *python*. Ela fornece uma estrutura de dados de matriz multidimensional, que se chamam *arrays*, que permite que sejam executadas operações eficientes nessa mesma estrutura de dados. O *numpy* oferece várias funções matemáticas e estatísticas para manipulação de arrays, bem como funcionalidades para álgebra linear e geração de números aleatórios. Sua eficiência e facilidade de uso tornam o *numpy* uma escolha popular para cálculos numéricos em *python*.

- **SciPy**: é uma biblioteca complementar ao *numpy* que fornece funcionalidades adicionais para computação científica. Ela inclui submódulos para otimização, álgebra

linear, processamento de sinais, interpolação, integração numérica, estatísticas, entre outros. A combinação do *numpy* e do *scipy* oferece um conjunto abrangente de ferramentas para resolver uma ampla gama de problemas computacionais encontrados na ciência e na engenharia.

- **Matplotlib**: é uma biblioteca *python* amplamente utilizada para visualização de dados. Ela oferece uma variedade de ferramentas para criar gráficos estáticos, incluindo gráficos de linhas, de dispersão, de barras, de pizza, histogramas, entre outros. O *Matplotlib* é altamente customizável, permitindo que os utilizadores controlem todos os aspectos da aparência dos gráficos, como cores, estilos de linha, marcadores e legendas. Além disso, o *matplotlib* é integrado com outras bibliotecas de análise de dados, como *pandas*, facilitando a criação de gráficos a partir de dados armazenados em estruturas de dados *pandas*.

- **Scikit-learn**: é uma biblioteca de ML em *python* que oferece uma ampla variedade de algoritmos de *supervised learning* e *unsupervised learning*. Ela é construída sobre as bibliotecas *numpy*, *scipy* e *matplotlib*, e fornece uma interface consistente e fácil de usar para treinar modelos, realizar seleção de recursos, *crossvalidation*, ajuste de hiperparâmetros e avaliação de desempenho de modelos.

- **Saxpy**: é uma biblioteca que implementa as ferramentas necessárias para se aplicar o algoritmo *SAX*. Foi desenvolvido recorrendo à biblioteca *numpy*.

A escolha desta linguagem e das bibliotecas utilizadas deve-se à sua ampla aceitação na comunidade de análise de dados e desenvolvimento de ML. Estas ferramentas possuem uma comunidade ativa, o que resulta em atualizações frequentes, abundante documentação, exemplos e projetos completos que servem de inspiração.

Na fase inicial do processamento, a abordagem adotada consiste na remoção de ruído do sinal de velocidade, visando garantir a análise de um sinal mais consistente, isto é, independentemente do ruído presente no sinal, este passará por um processo de uniformalização. Para isso, recorre-se a um filtro chamado de *Savitzky–Golay*. Em seguida, obtem a aceleração a partir do sinal já livre de ruído, o que é simplesmente a derivada do sinal original.

O objetivo é extrair informações do sinal da aceleração e obter um conjunto de padrões que possam ser contados, ordenados e utilizados como uma "impressão digital"que represente os principais comportamentos do condutor na espectativa que esses padrões tenham alguma relação direta com a agressividade da viagem correspondente. Esses padrões consistem em símbolos que estão diretamente relacionados com a intensidade da aceleração e de travagens registadas em determinado instante. Para realizar essa etapa, é necessário converter as séries temporais em séries temporais de menor resolução, ou seja, reduzir sua complexidade, para isso utiliza-se a técnica *Symbolic Aggregate Approximation (SAX)*. Com os padrões todos contados cada viagem será representada por palavra que representa a os padrões mais frequentes que serão objeto de estudo nesta dissertação.

O motivo de se analisar aceleração ignorando a velocidade, tem a ver com a expectativa de que este dado seja mais relevante para a análise do comportamento visto que este traduz mais a reação dos condutores ao eventos durante a condução. Ou seja, significa que espera-se que a aceleração contenha de forma mais clara informação que indica se o condutor é mais ou menos agressivo durante a condução. Por outro lado a velocidade depende do tipo de via em que se circula, ou seja, dos limites de velocidade definidos bem como as condições de trânsito.

3.3.1 Savitzky–Golay

O filtro de *Savitzky-Golay* é uma técnica de suavização de sinal amplamente utilizada em processamento digital de sinais. A principal ideia por trás do filtro de *Savitzky-Golay* é ajustar localmente uma função polinomial aos dados dentro de uma janela móvel. Essa função polinomial é ajustada usando o método dos mínimos quadrados, o que significa que ela é escolhida para minimizar a soma dos quadrados das diferenças entre os valores reais dos dados e os valores previstos pela função polinomial.

Ao ajustar uma função polinomial aos dados em vez de se utiliar uma média simples ou outro tipo de filtro, o filtro de *Savitzky-Golay* é capaz de suavizar o sinal sem causar tanto deslocamento temporal, em comparação a outros métodos de suavização. Isso é especialmente útil em situações onde é importante preservar as características temporais do sinal, como mostra a Figura 3.3, mantendo o formato e a altura dos máximos e mínimos locais sem acrescentar fase ao sinal. A Figura 3.3 mostra a azul a série temporal com o ruído, portanto antes de se aplicar o filtro, e nas outras cores o sinal após o filtro, para diferentes filtros. É relevante notar que o filtro de tamanho de janela de 7 e um uma *polyorder* de 6 significa por necessidade que a aproximação aos pontos dessa janela será perfeita, o que significa que o sinal de entrada é igual ao sinal de saida . É importante realçar que no eixo horizontal está o tempo em segundo e no eixo vertical está representada a velocidade em km\h.

A Figura3.4 mostra o efeito do filtro do domínio da frequência onde fica claro que as frequências baixas são mantidas ou menos atenuadas que as frequências altas, criando o efeito de filtro "passa baixo". A azul, a figura tem as componentes do sinal antes de ser aplicado o filtro e os resto das cores as componentes do sinal depois de ser aplicado o filtro. É importante realçar que no eixo horizontal estão as frequências em *Hz* e no eixo vertical estão as magnitudes dessas mesmas frequências.

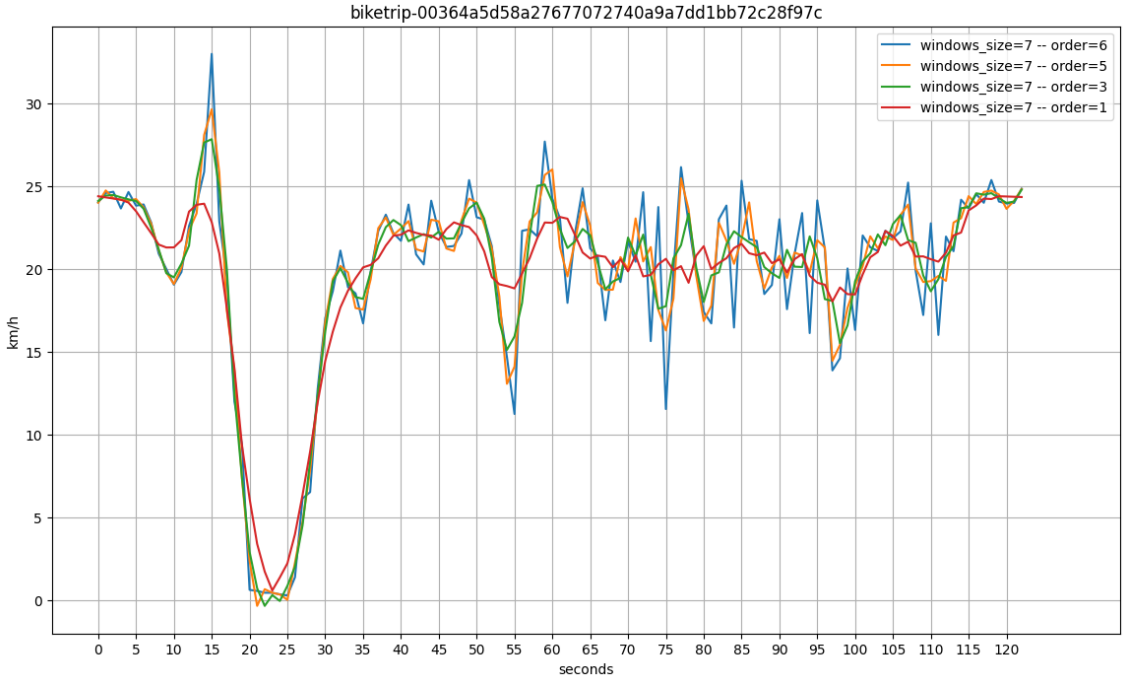

Figura 3.3: Efeito do filtro Savitzky–Golay numa série temporal para diferentes *polyorders*. O filtro com *polyorder* igual a 6 correspode à série de cor azul que é a série temporal original.

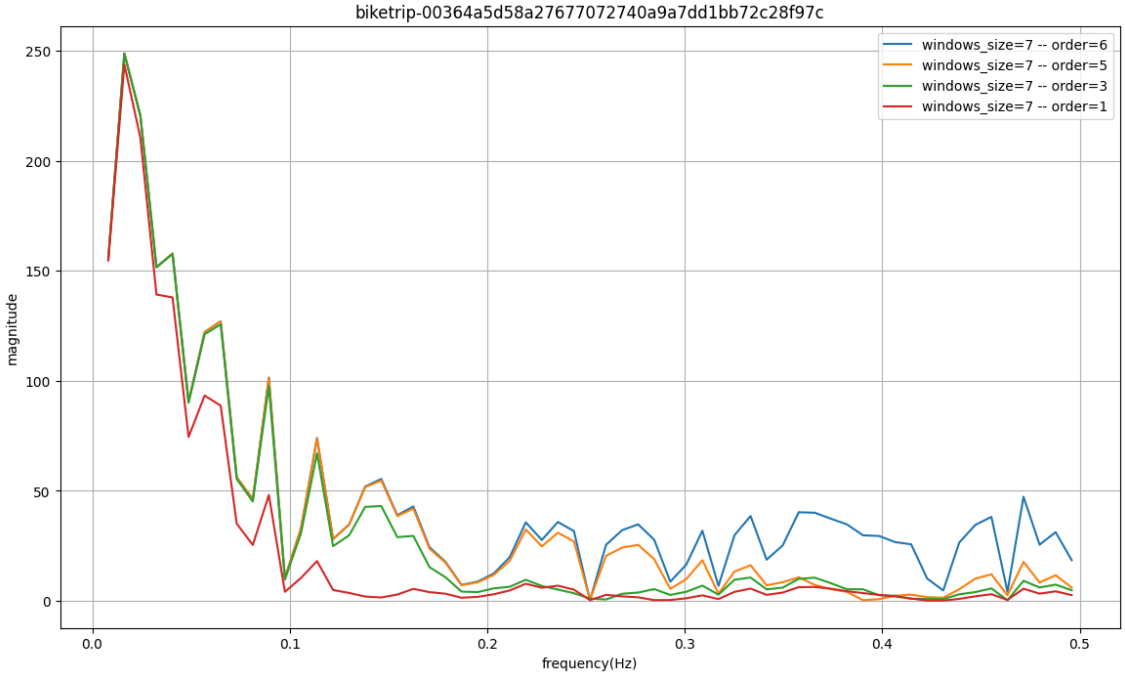

Figura 3.4: Efeito do filtro Savitzky–Golay no domínio da frequência de uma série temporal para diferentes *polyorders*. O filtro com *polyorder* igual a 6 correspode à série de cor azul que é a série temporal original.

Além disso, o filtro de *Savitzky-Golay* pode ser adaptado para suavizar sinais em diferentes graus, dependendo da *polyorder* utilizada e do tamanho da janela móvel. Quanto maior a janela, maior a capacidade de suavização, e menor será a sensibilidade às variações, já a ordem, quanto maior, melhor é a capacidade de captação dos detalhes do sinal. Essa personalização permite uma flexibilidade significativa que leva a que a aplicação do filtro permita a obtenção de diferentes resultados.

3.3.2 SAX

O método SAX é aplicado após a obtenção da derivada do sinal limpo da velocidade, que corresponde à série temporal da aceleração. Este algoritmo tem como objetivo reduzir a dimensionalidade da aceleração, proporcionando uma representação mais simplificada da série temporal. Isso significa que o resultado final terá menos complexidade e menos valores, o que facilita a identificação de padrões no sinal.

Para obter essa representação simplificada, o processo começa com a normalização dos valores da aceleração, recorrendo ao método "*Z-standarization*", que é possível porque assume-se que os valores da aceleração das viagens seguem uma distribuição normal, como por consta na Figura 3.5, logo os dados não sofrerão distorções significativas. Em seguida, o eixo vertical, que contém os valores normalizados, é dividido em intervalos equiprováveis.

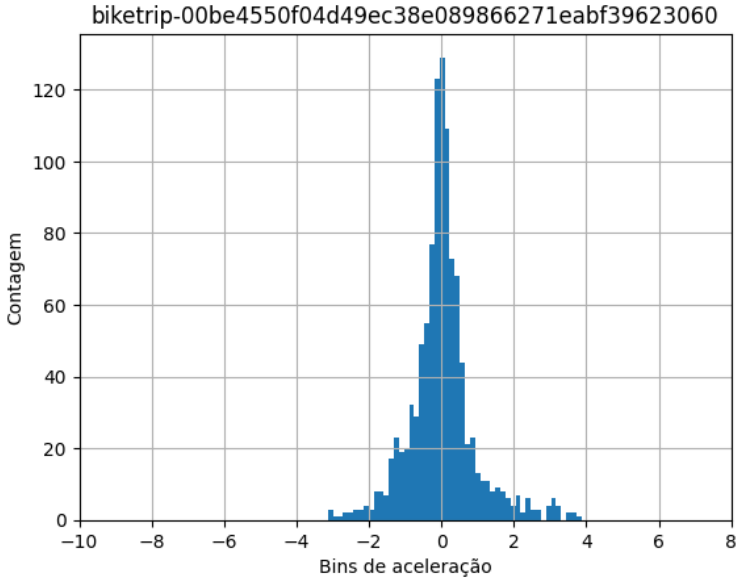

Figura 3.5: Exemplo de uma viagem cujos dados de aceleração obedecem a uma distribuição normal, com valores de média $\mu = 0.2$ e desvio padrão $\sigma = 0.92$

Após dividir o eixo vertical em intervalos equiprováveis e normalizar os valores da aceleração, o próximo passo é associar cada ponto da série temporal normalizada ao intervalo ao qual pertence, como exemplificado na Figura3.6.

Essa etapa de associação dos pontos aos intervalos é o passo final na aplicação do algoritmo SAX. Ao representar os pontos da série temporal por meio de símbolos que correspondem aos intervalos equiprováveis, consegue-se assim atingir a representação simplificada da série, obtendo o resultado esperado que será utilizado na a análise.

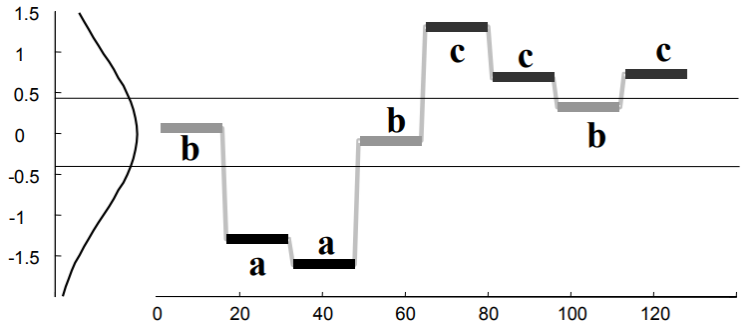

Figura 3.6: Exemplo do resultado da aplicação do algoritmo SAX [19]. A sequência de símbolos é "baabccbc".

A ideia é, depois de obter a sequência de símbolos, aplicar uma janela deslizante que abrange um conjunto de pontos, que será considerado um padrão. Uma vez que a janela deslizante percorre toda a sequência de símbolos, cada padrão encontrado é contabilizado e ordenado por sua frequência de ocorrência. Isso significa que os padrões mais comuns na série temporal serão identificados e registados. No final deste processo os padrões mais frequentes serão então convertidos numa sequência de símbolos que representa o *ranking* de padrões da viagem que será utilizada para análise posterior.

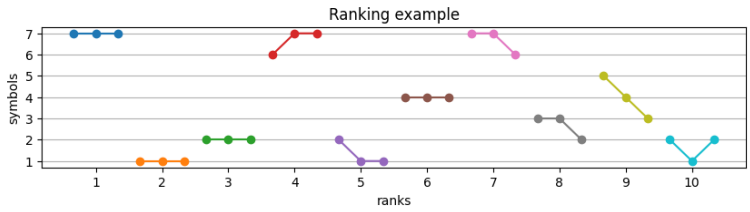

Figura 3.7: Exemplo de um *ranking*. A sequência de símbolos que representa este *ranking* é "777111222677211444776332543212".

É esperado que estas palavras contenham informações relevantes e suficientes para se obter conclusões pertinentes sobre o comportamento do autor da serie temporal. Na Figura 3.7, levando em conta que cada conjunto da sequência de 3 símbolos representam um padrão, o padrão mais frequente nesta viagem é o "777"que é um padrão composto por símbolos representam acelerações muito agressivas. O segundo padrão mais frequente é o padrão "111"que representa um padrão composto por símbolos representam travagens muito agressivas. O terceiro padrão é o "222"que apesar de ser composto por símbolos que representam travagens, não são tão agressivas como a anterior. Todos os próximos padrões seguem a mesma lógica.

3.3.3 Classificação dos dados

Após os dados terem sido processados recorrendo ao SAX será efetuada a classificação dos mesmos. O primeiro passo no processo de classificação é a aplicação da equação 2.4 que atribui uma pontuação à viagem com base no grau de agressividade, no qual a pontuação é o resultado de um somatório no qual cada parcela representa a contribuição de um padrão no resultado final das viagens.

A equação é plausível, porque assumindo que cada padrão é composto por três símbolos, sendo que cada símbolo é representado por números que refletem comportamentos relacionados a acelerações e travagens. A média desses padrões é ponderada na equação, refletindo-se num aumento da agressividade quanto mais extremo forem os símbolos que compõem os padrões, sejam estes acelerações ou travagens. Para garantir que as travagens sejam consideradas tão agressivas quanto as acelerações, é subtraído o valor que representa acelerações próximas de zero (no caso [5] foi 3 porque os símbolos vão de 0 a 6, que significa que o padrão que representa uma aceleração próxima de zero), e o módulo desse resultado é então calculado.

Outro parâmetro considerado na equação é o desvio padrão dos símbolos contidos no padrão. É razoável supor que uma aceleração mais constante num curto espaço de tempo é preferível a uma aceleração que varia, pois isso expressa estabilidade na condução. Este valor é multiplicado por 0.5 para incorporar esse aspecto na pontuação final. A razão pela qual este fator foi escolhido não foi especificada no artigo onde foi apresentada [5], mas pode estar relacionada com a importância relativa deste parâmetro em relação aos outros na determinação da agressividade da condução.

Após a soma desses dois valores, a pontuação é então multiplicada pela frequência relativa de cada padrão na mesma viagem. Essa multiplicação permite comparar viagens de diferentes durações de forma mais justa, levando em consideração a frequência com que determinados padrões ocorrem ao longo da viagem.

Após a pontuação ter sido atribuída será utilizada para dividir as viagens disponíveis em duas classes diferentes, neste caso agressivo e não agressivo, que corresponderão intervalados delimitados por valores de *SCORES* arbitrariamente escolhidos.

3.4 Análise

Em seguida, a implementação de um algoritmo que recorra a técnicas de ML que seja capaz de utilizar os dados coletados para encontrar alguma propriedade, ou padrões na tal sequência de símbolos que representam o *ranking*, que permitam de alguma forma a automatização da identificação ou distinção de tipos de comportamentos, para que posteriormente se inferira algum perfil dos condutores.

A abordagem escolhida foi a implementação de árvores de decisão.

3.4.1 Árvores de decisão

Como menciodado anteriormente, após os dados terem sido classicados, serão aplicados modelos a fim de entender se estas classes têm alguma relação relevante com os padrões presentes nos *rankings* das viagens, permitindo averiguar se é possível acelerar o método a avaliação de viagens sem a necessidade de avaliar o *SCORE* das mesmas. Os modelos considerados foram as árvores de decisão.

A escolha da árvore de decisão como algorítmo para tentar solucionar o problema está ligado com as suas vantagens como a clareza no processo da tomada de decisão da árvore que facilita a sua interpretação e a sua simplicidade de implementação e de otimização recorrendo ao *pruning*. O *pruning* é o processo de remover partes das árvores que não são essenciais, ou seja, que não contribuem para descrever o universo que se está a tentar modelar. Isto permite que se evite o *overfitting* que acontece quando os modelos têm alto desempenho para dados já vistos, ou seja, os dados do conjunto de treino relativamente ao desempenho em conjunto de dados ainda não vistos, ou seja, conjunto de teste.

A qualidade da técnicas será então avaliada numa primeira instância através da *accuracy*, para se perceber se os modelos são viáveis ou não, e posteriormente caso sejam viáveis serão também analisados a *precision*, o *recall* e o *f1-score*.

A *accuracy* mede proporção de previsões corretas em relação ao total, sendo calculada pela divisão do número de previsões corretas pelo número total de previsões. As previsões corretas são a soma de verdadeiros positivos (TP) e de verdadeiros negativos (TN), e o total de previsões são as anterior adionado aos falsos positivos (FP) e aos falsos negativos (FN).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.1}$$

A *precision* mede a proporção de previsões positivas corretas em relação ao total previsões positivas, ou seja, é a capacidade do modelo de não classificar uma instância negativa como positiva.

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

A *recall* mede a proporção de previsões positivas corretas em relação ao total de instâncias positivas no conjunto de dados. Em outras palavras, é a capacidade do modelo de encontrar todas as instâncias positivas.

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

O *f1-score* é a uma métrica que calcula a média harmónica entre a *precision* e a *recall*.

$$F1score = 2 * \frac{precision * recall}{precision + recall} \tag{3.4}$$

4

Implementação

Este capítulo pretende a apresentar as implementações feitas no desenvolvimento da presente dissertação, contém o desenvolvimento da *pipeline* de ingestão de dados, bem como todo o desenvolvimento de processamento e análise desenvolvidos nesta dissertação. Conta com a alteração feita na aplicação *Android* que implementa a funcionalidade que permite enviar os dados para a base de dados *BigQuery*, a implementação da *CloudFunction*. Será explicado como se procedeu à preparação dos dados e como se desenvolveu o *SAX* e a árvore de decisão.

4.1 Aplicação Android

Para proceder à coleta de dados, foi utilizada uma aplicação *Android*. Como já foi mencionado, a aplicação não foi feita de raiz, foi aproveitada de um repositório do *GitHub*, com um projeto chamado *GPS LOGGER* [10]. É um projeto gratuito e *open-source* o que significa que o código é disponível ao público para uso. Foi necessária uma adaptação que implementa a funcionalidade que permite integrar o *Android* com os serviços *GoogleCloud*. Esta funcionalidade permite que os dispositivos enviem os dados recolhidos para um mesmo local através dos serviços *PubSub* que conecta ao resto da *pipeline*.

A alteração foi realizada utilizando o ambiente chamado *Android Studio*. O *Android Studio* é um ambiente de desenvolvimento integrado *IDE* oficial para o desenvolvimento de aplicações *Android*, concebido pela *Google*. É baseado no *software IntelliJ IDEA* da *JetBrains* e foi adaptado especificamente para o desenvolvimento em *Android*.

O projeto original é composto por 35 classes que juntas implementam funcionalidades como a gravação das viagens e exportação das mesmas para pasta local, para isso ser alcançado, está implementado a conectividade com multiplos satélites para que haja redundância na informação lida e assim haver uma confiança nos dados recolhidos. Dessas classes, aquela responsável pela exportação dos dados, a *Exporter* permite a geração dos ficheiros nos seus respetivos formatos e guardá-las uma diretoria.

A alteração feita no projeto incidiu nesse ficheiro, o *Exporter*, com o objetivo de aproveitar a sua propriedade e adicionalmente enviar os dados para os serviços *GoogleCloud*.

Alterou-se a forma com que os dados são construídos, para que haja uma eficiência na transferência nos dados entre as duas partes, o que resultou na construção de um estrutura *JSON* que será enviado através dos serviços *PubSub* da *Google* que dariam o seguimento para o resto da *Pipeline*. A estrutura de dados utilizada contêm os seguintes dados:

- *DeviceID*
- *TripID*
- *array*
  - latitude
  - longitude
  - *speed*
  - *timestamp*

Para que a aplicação seja capaz de se comunicar com os serviços da *Google* é necessário proceder a uma configuração que envolve a devida comunicação à aplicação da informação necessária, nomeadamente as credenciais que a irão permitir ter a acesso ao dominio específico onde está o resto da *pipeline*. Esta informação está no formato de ficheiro *JSON* obtido na plataforma dos serviços *GoogleCloud*, durante a criação do ambiente onde o projeto irá ser construido na aba de chaves das *contas de serviço*, onde é possível adicionar e eliminar chaves com base nas necessidades do projeto. O ficheiro é composto por:

- *type*
- *project_id*
- *private_key_id*
- *private_key*
- *client_email*
- *client_id*
- *auth_uri*
- *token_uri*
- *auth_provider_x509_cert_url*
- *client_x509_cert_url*
- *universe_domain*

Este ficheiro é depois incluído no projeto da aplicação e as credencias presentes nesse ficheiro carregadas pelo código do projeto desenvolvido para que os envios dos dados para o serviço *PubSub* sejam realizadas com sucesso.

4.2 Cloud Function

A *CloudFunction* foi desenvolvida com a simples função de recolher os dados enviados pela aplicação, recebidos pelo serviço *PubSub*, e inserí-los no *BigQuery*. Foi implementada antecipando a possibilidade dos utilizadores enviarem a mesma viagem múltiplas vezes, e no caso disso acontecer, essa tal viagem seria ignorada e, portanto, não inserida na tabela.

É importante realçar que a inserção dos dados será feita por ponto registado, ou seja, cada instância inserida na tabela corresponde a um ponto lido. As colunas as serem inseridas são *DeviceID*, *tripID*, latitude, longitude, *speed* e *timestamp*. A implementação está descrita na Figura4.1.

```python
import base64
import json
from google.cloud import bigquery

def hello_pubsub(event, context):
    # Extract the Pub/Sub message and decode the data
    pubsub_message = base64.b64decode(event['data']).decode('utf-8')

    # Parse the JSON data
    data = json.loads(pubsub_message)
    print("jsonarray size:",len(data["trip"]))
    table_id = "projectbk2-71159.Trips.TripTable"
    column_name = "tripID"
    desired_value = data["tripID"]
    query = f"""
    SELECT COUNT(*)
    FROM `{table_id}`
    WHERE {column_name} = '{desired_value}'
"""
    client = bigquery.Client()
    query_job = client.query(query)
    for row in query_job.result():
        count = row[0]

    print("data in bigquery",count)
    # Initialize BigQuery client and table
    table = client.get_table(table_id)

    # Insert the data into BigQuery table(second try)
    if count == 0:
        rows_to_insert = [(data['deviceID'],data['tripID'],
                           coord['lat'],coord['long'],coord['speed_ms'],
                       coord['timestamp']) for coord in data["trip"]]
        errors = client.insert_rows(table, rows_to_insert)

        if errors:
            print(f'Error inserting rows: {errors}')
        else:
            print(f'Successfully inserted rows: {rows_to_insert}')
```

Figura 4.1: Implementação da Cloud Funtion

Não foram utilizadas as configurações de padrão das *CloudFunction*, nas quais foram aumentadas o tempo limite na execução de um *request*, que consiste no tempo definido que o processo tem para executar o *request* antes de se abortar a execução do mesmo. A alteração feita foi de 60 segundos para 540 segundos, porque viagens longas podem demorar demasiado tempo a processar, e com esta alteração previne-se a recepção de dados incompletos.

4.3 Estrutura do BigQuery

A estruta da tabela *BigQuery* utilizada foi simples e está representada na Tabela 4.1, adequada para armazenar as informações processadas pela *CloudFunction*.

Colunas	tipo de dado
DeviceID	string
tripID	string
latitude	float
longitude	float
speeed	float
timestamp	string

Tabela 4.1: Colunas da tabela Bigquery

4.4 Processamento

O processamento dos dados é onde se pretende fazer a sua transformação para a forma adequada para se proceder à análise proposta. Será abordada a forma como se eliminou o ruído dos dados recolhidos das viagens bem como a transformação dos mesmos em *rankings*.

4.4.1 Tratamento do sinal

A implementação da parte de processamento dos dados foi realizada em *python* devido à sua flexibilidade e à vasta gama de ferramentas e bibliotecas disponíveis para análise de dados. A escolha do *python* como ambiente de desenvolvimento proporcionou uma abordagem simples, facilitando a implementação de algoritmos de filtragem e análise.

O objetivo principal desse processamento é gerar um *ranking* que represente os padrões mais comuns presentes nos dados de aceleração. No entanto, devido à natureza dos dados obtidos diretamente dos sensores, é necessário lidar com o ruído presente nos sinais de velocidade. Esse ruído pode distorcer os padrões e afetar a precisão das análises posteriores. Portanto, uma etapa crucial do processamento é a filtragem do sinal de velocidade para remover o ruído indesejado.

4.4.1.1 *Savitzky–Golay*

Para realizar essa filtragem, recorreu-se à biblioteca *scipy*, que oferece uma implementação eficiente do filtro *Savitzky–Golay*. Esse filtro é amplamente utilizado em processamento de sinais para suavizar e remover ruído de séries temporais. Ao aplicar o filtro *Savitzky–Golay* ao sinal de velocidade, é possível obter um sinal mais limpo e livre de ruído, que aumenta o grau de confiança nos resultados das análises subsequentes.

A função do filtro *Savitzky–Golay* requer alguns parâmetros, como o tamanho da janela deslizante e a ordem dos polinómios utilizados para interpolação. A escolha desses parâmetros pode influenciar a eficácia da filtragem e, portanto, requer cuidado e ajuste adequado para cada conjunto de dados específico. Nessa lógica os o valor da janela utilizado nesta dissertação foi de 7 pontos, e as diferentes ordens de polinómios utilizados

foram 1, 3, 5 e 6. Estes valores foram escolhidos por tentativa erro, analisando visualmente o efeito destes parametros na eliminação do ruído das séries temporais.

Além disso, a biblioteca *scipy* oferece na função do filtro *Savitzky–Golay* a capacidade de calcular a derivada do sinal filtrado. É especialmente útil no contexto da análise de aceleração, visto que será esse o dado que será efetivamente analisado, pois permite obter diretamente esse sinal a partir do sinal de velocidade filtrado, num só passo. Essa abordagem é mais precisa e eficiente do que calcular a aceleração a partir das diferenças consecutivas no sinal de velocidade. Ao se utilizar essa funcionalidade, obtem-se o sinal a aceleração que implica perder um ponto, pois não é possivel obter o valor da aceleração a partir do primeiro ponto de velocidade pois nenhum outro antecede este para se calcular a diferença. De qualquer forma esta perda é irrelevante ao resto a da análise, devido à irrelevâcia estatística desses pontos.

4.4.1.2 *SAX*

Com o sinal de aceleração obtido, o próximo passo é identificar e contar os padrões presentes nesse sinal. Para isso, é necessário reduzir a dimensionalidade do sinal de aceleração, a fim de facilitar a detecção dos padrões mais comuns. Essa redução é realizada utilizando a biblioteca *saxpy*, que contém as ferramentas que permite implementar o algoritmo SAX.

O algoritmo SAX é uma técnica eficaz para representar séries temporais complexas numa sequência discreta de símbolos, tornando mais fácil identificar padrões recorrentes. A biblioteca *saxpy* oferece ferramentas adicionais para manipulação de séries temporais, incluindo a normalização do sinal e a divisão do eixo do vertical em n intervalos.

A normalização é o primeiro passo necessário para converter os valores da aceleração em valores que sejam justamente comparáveis com os valores das demais viagens. A fórmula implementada pela biblioteca *saxpy* é a *Z-standarization* que é a que se pretende usar de acordo a metodologia proposta. A normalização do sinal da aceleração é plausível pois assume-se que os valores da aceleração das viagens já obedecem a uma distribuição Gaussiana [5], que implica que a fórmula não irá distorcer de forma significativa os dados durante o processo.

O próximo passo é dividir o eixo vertical em intervalos que serve o propósito de permitir que os valores da aceleração normalizada possam ser convertidos em símbolos, visto que esta divisão será feita assumindo que os intervalos equiprováveis são de uma distribuição normal reduzida, cada um deles corresponderá a um símbolo, atingindo assim a redução de dimensionalidade. A biblioteca oferece tanto as ferramentas para a divisão do eixo vertical em intervalos equiprováveis quanto a conversão de séries temporais em sequências de símbolos. O número de intervalos escolhidos foi de 7, ou seja, cada ponto da série temporal da aceleração será convertida num de 7 símbolos pois além de ser ímpar que permite que as acelerações próximas a 0 tenham representação, também foi o valor mais viável para os dados disponíveis, mas poderia ser qualquer número de intervalos

desde que seja ímpar. A biblioteca *saxpy* efetua essa redução de dimensionalidade fazendo corresponder os intervalos a letras, mas nesta dissertação será efetuada a conversão dessas letras para números para facilar os cálculos como feito no artigo [5]. Através de uma janela deslizante que se escolheu ser de 3 pontos consecutivos, irá se contar os padrões presentes nessa sequência de símbolos, sendo que cada padrão é composto por 3 símbolos. O tamanho de um padrão foi escolhido também baseado na ideia defendida em [5], de que é o tempo necessário para que uma pessoa defina o comportamento que pretende que o veículo tenha efetivamente. Esses padrões são então contados e ordenados por frequência, permitindo a criação do *ranking* com os padrões mais comuns.

4.5 Análise

4.5.1 Abordagem supervisionada

A implementação foi feita em *python* com ênfase na biblioteca *sklearn*. O objetivo principal consitiu na obtenção de modelos com desempenhos que justifiquem o metodologia proposta. Os algoritmos escolhidos para lidar com o problema foram as àrvores de decisão.

4.5.1.1 Classificação

Como já mencionado, os dados disponíveis não estão classificados, o que exigiu a sua classificação recorrendo à equação 2.4. Esta equação foi implementada, levando em conta informação relevante obtida no passo feito no processamento dos dados, o *SAX*, no qual obteve-se a contagem do padrões por viagem que será utilizado, juntamente com a duração das mesmas, para calcular a frequência relativa de cada padrão de cada viagem, parâmetro necessário para se calcular o *score*.

Com o *score* calculado é então escolhido um valor para definir quais são as viagens agressivas das viagens não agressivas, obtendo assim um *dataset* que tem a sequência de símbolos de cada viagem devidamente classificada como agressivo ou não agressivo, permitindo assim a a análise supervisionada.

4.5.1.2 Árvore de decisão

As árvores de decisão, sem nenhuma condição que limite o seu crescimento, tendem a descrever em demasia o conjunto de treino, levando a que o seu desempenho no conjunto de testes não esteja a altura, esse fenómeno chamas-se de *overfitting*. Para evitar o *overfitting* é necessário efetuar *pruning*. O *pruning* é o processo de remover partes das árvores que não são essenciais, ou seja, que não contribuem para descrever o universo que se está a tentar modelar.

O *pruning* recorrendo ao *sklearn* passa por avaliar diferentes árvores de decisão com diferentes complexidades. Essas diferentes complexidades são obtidas definindo à *priori*

as características da árvore que se se vai construir, podem ser definidas as seguintes características:

- max depth
- min samples split
- min samples leaf
- min weight fraction leaf
- min impurity decrease
- max leaf nodes

Numa primeira fase da análise foi usada um *crossvalidation* em árvores de decisão com *max depth* igual a 7 e *max leaf nodes* igual a 10, com dois propósitos, a primeira seria a de avaliar a *accuracy* para mostrar a viabilidade do modelo.A segunda seria para mostrar o desempenho tanto para o conjunto de treino quanto para o conjunto de teste e mostrar que para os valores de *pruning* escolhidos para esta fase não há *overfitting*, ou seja, o desempenho no conjunto de treino não é assim tão superior ao desempenho no conjunto de teste.

Numa segunda fase serão geradas árvores de decisão finais para o *dataset* classificado com *thresholds* diferentes, onde estas serão avaliadas com *accuracy*, *precision*, *recall* e *f1-score* recorrendo sempre às ferramentas disponíveis pela biblioteca *sklearn*.

A sequência de símbolos que representa os símbolos dos padrões dos *rankings* das viagens, será dividido em conjuntos de 3 símbolos, pois assim representam os respetivos padrões e o valor resultante da operação $abs[mean(padrão) - 4]$ de cada padrão será utilizada como variável de entrada dos modelos em conjunto com a respetiva classe de cada viagem.

5

Análise e Resultados

Este capítulo discute-se a análise dos resultados obtidos com a metodologia proposta, explorando os dados disponíveis e os resultados dos modelos de *machine learning*, nomeadamente as árvores de decisão.

O objetivo principal é demonstrar a viabilidade da metodologia, destacando os contextos em que é mais eficaz.

A análise dos dados revelou anomalias importantes de se levar em conta na interpretação dos resultados obtidos pelos modelos desenvolvidos e bem como as propostas para as ultrapassar na medida do possível.

Os resultados dos modelos de árvores de decisão variaram significativamente com diferentes classificações aos dados disponíveis, destacando a importância da adaptabilidade da abordagem.

Em suma, este capítulo oferece uma análise abrangente dos resultados, destacando a eficácia e relevância da metodologia proposta para a aplicação em questão.

5.1 Análise dos resultados dos dados

As viagens disponíveis são de diferentes durações. A distribuição dessas durações está descrita na Figura 5.1.

Pode-se observar que a maioria das viagens têm durações abaixo dos 10 minutos, cerca de 650, havendo uma queda brusca de número de viagens no seguintes intervalos. Há cerca de 200 viagens com durações entre 10 e 15 minutos, 90 com durações entre 15 e 20 minutos e a partir daí os intervalos têm menos de 50 viagens, onde os números vão diminuindo quanto maior for a duração que o intervalo representa.

É necessário ter em consideração que como a maior parte das viagens tem duração inferior a 10 minutos, a análise pode ser condiciona pela possibilidade de que os *rankings* não tenham dados o suficientes para terem convergido para valores que realmente definem o perfil do condutor.

Este dados são na sua maioria dados bem coletados dos o que significa que a maioria não precisou de nenhum tipo de correção ou desqualificação, no entanto no *dataset* das

viagens de bicicleta [8] existem viagens com defeitos de naturezas diferentes.

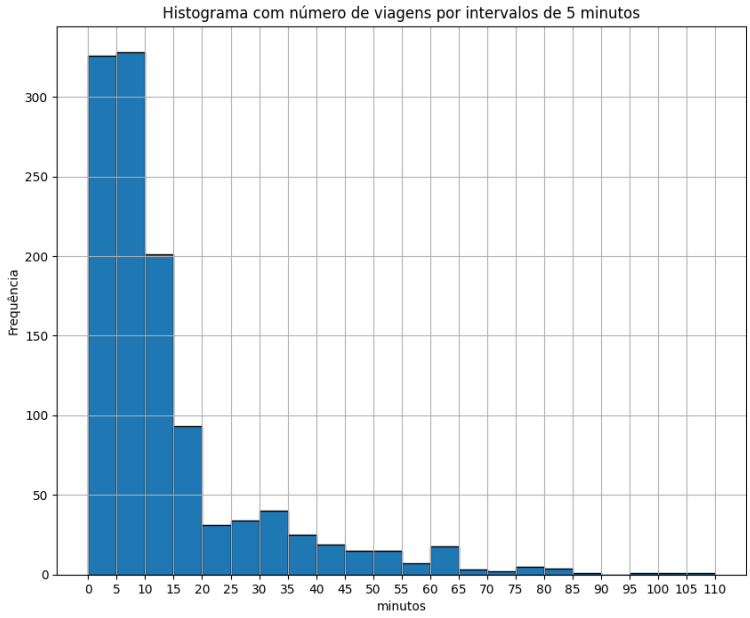

Figura 5.1: Histograma do número de viagens por duração das viagens dividido em intervalos de 5 minutos

Os defeitos identificados nos dados distinguem-se em dois grupos diferentes. Um dos tipos de defeito são aqueles que resultam em leituras de pontos que representam picos que se destacam dos valores lidos no resto da viagem como consta na Figura 5.2, onde nos primeiros segundos da viagem há valores lidos que atingem a orderm dos 800km\h, valores impossíveis de se obter na prática do ciclismo, chamemos de defeito tipo 1. O outro tipo de defeito representa uma leitura com consistência de valores de ordens impossíveis para viagens feitas de bicicletas, como consta na Figura 5.3, que será denominado de defeito tipo 2, em que os valores lidos durante a viagem toda está na ordem dos 550km\h.

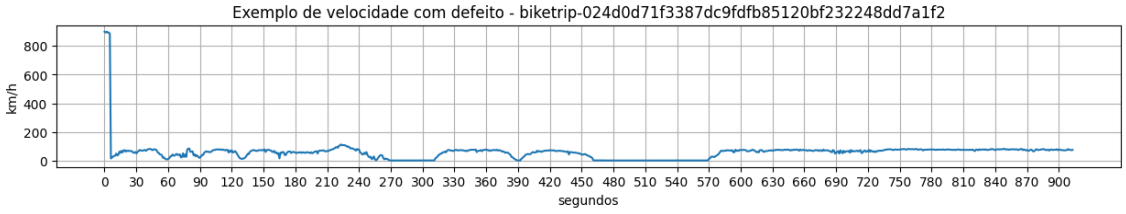

Figura 5.2: Exemplo de gráfico de velocidade com defeito tipo 1

Figura 5.3: Exemplo de gráfico de velocidade com defeito tipo 2

Como os picos das viagens com erros de tipo 1 ocorrem na sua maioria das vezes no início da viagem ou no final, foi proposta a remoção dos primeiros e os últimos 40 segundos das viagens com mais de 5 minutos, que tivessem defeito. O limite escolhido foi de 5 minutos porque remover pontos de viagens mais curtas teria um impacto muito significativo devido à escassez de pontos nas mesmas. As seguintes Figuras 5.4 e 5.5 e as Figuras 5.6 e 5.7 mostram o antes e depois da retificação das viagens com defeito tipo 1 e tipo 2 respetivamente:

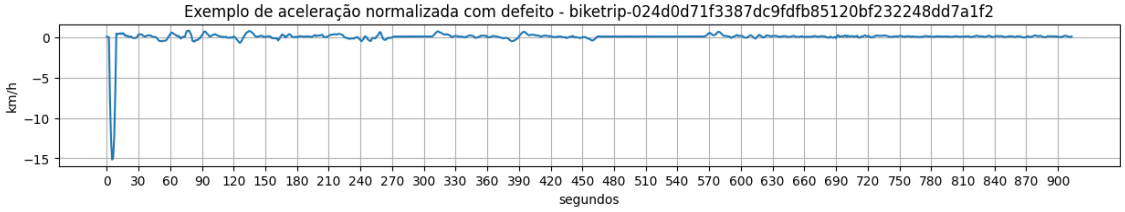

Figura 5.4: Exemplo de gráfico de aceleração normalizada com defeito tipo 1

Figura 5.5: Exemplo de gráfico de aceleração normalizada com defeito tipo 1 com correção

Figura 5.6: Exemplo de gráfico de aceleração normalizada com defeito tipo 2

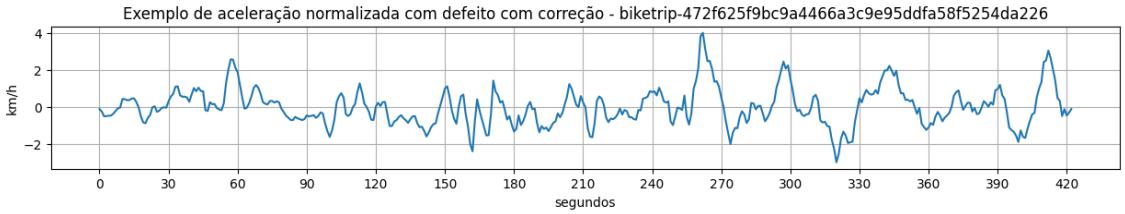

Figura 5.7: Exemplo de gráfico de aceleração normalizada com defeito tipo 2 com correção

A conclusão que se retira é que a análise das viagens com erros do tipo 1 são mais afetadas que as viagens com erros do tipo 2. De facto a retificação das viagens com erros tipo 2 não são necessárias visto a diferença que causam no *ranking* final não é significativo em comparação com a retificação das viagens com erro tipo 1. As Figuras 5.8 e 5.9 e as Figuras 5.10 e 5.11 demostram o impacto que a retificação tem nas viagens com defeito tipo 1 e nas viagens com defeito tipo 2, respetivamente.

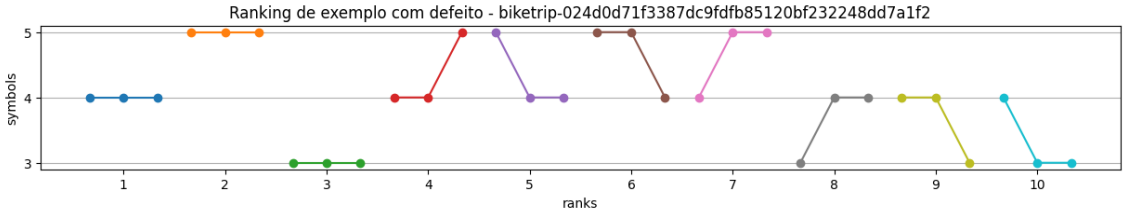

Figura 5.8: Exemplo de *ranking* de viagem com defeito tipo 1

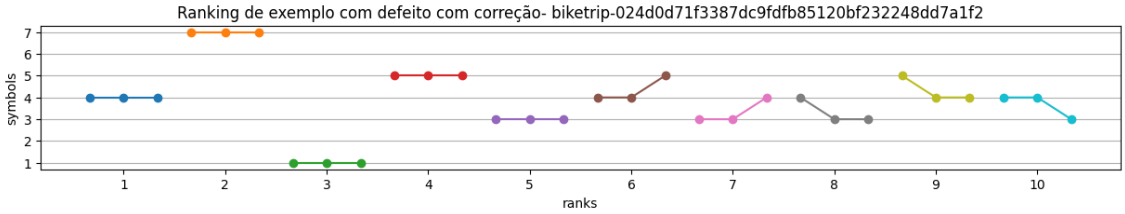

Figura 5.9: Exemplo de *ranking* de viagem com defeito tipo 1 com correção

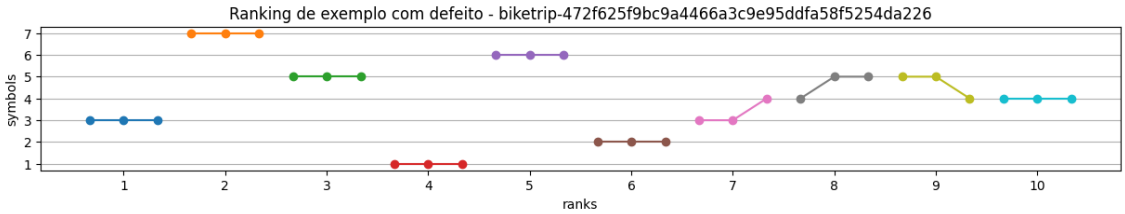

Figura 5.10: Exemplo de *ranking* de viagem com defeito tipo 2

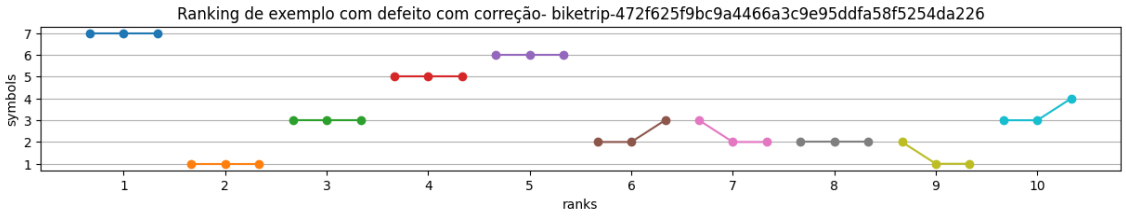

Figura 5.11: Exemplo de *ranking* de viagem com defeito tipo 2 com correção

Existem casos em que as viagens têm um defeito do tipo 1 no meio da viagem como consta na Figura 5.12, onde é difícil definir uma regra que não comprometa o processamento das outras viagens. Como estas viagens originam *rankings* defeituosos como consta na Figura 5.13, ou seja, os padrões presentes nos *rankings* que corresponde a este tipo de viagens normalmente não são compostos por uma variedade de símbolos como as outras viagens, onde os padrões são apenas compostos por 3 ou 4 símbolos diferentes.

Figura 5.12: Exemplo de gráfico de velocidade com defeito tipo 1 no meio da viagem

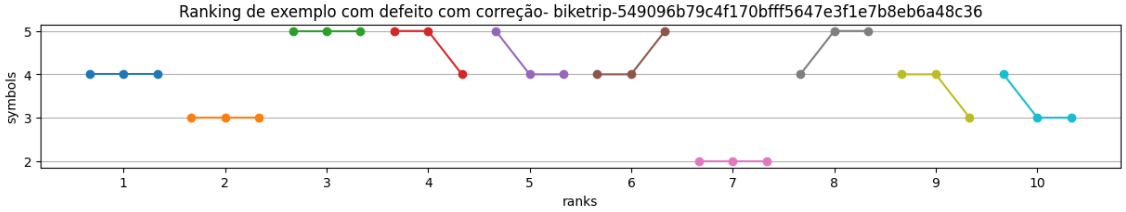

Figura 5.13: Exemplo de *ranking* de viagem com defeito tipo 1 no meio da viagem

Foram testados vários filtros diferentes aos dados a fim de se perceber quais eram mais adequados para o tratamento dos dados. O filtro escolhido foi *Savitzky-Golay* como já mencionado, com o valor do tamanho da janela definido a 7 para todos, e os valores para a *polyorder* de 1, 3, 5, 6. É relevante realçar que os filtros não são capazes de remover os erros tipo 1 das séries temporais, servindo apenas o propósito de remover ruído de altas frequências, como um filtro "passa baixo". A conclusão resultante destes testes é que as *polyorders* de 5 e 6 não são adequadas para a análise proposta porque na maioria dos casos os *rankings* originados por dados resultantes destes filtros têm o mesmo problema da falta de variedade de símbolos mencionados na questão das viagens defeituosas do

tipo 1 no meio da viagem, e também contêm padrões que não fazem sentido aparecerem nos 10 padrões mais frequentes da viagem no sentido prático da questão, como consta na Figura 5.14.

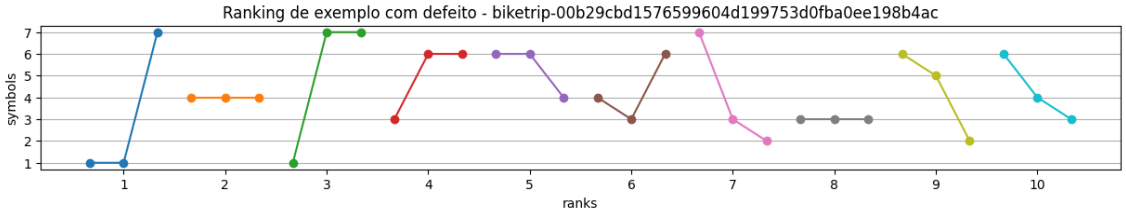

Figura 5.14: Exemplo de *ranking* com defeito por polyorder 5 e 6(neste caso é o mesmo *ranking* para os dois filtros)

De qualquer forma, será analisado numa primeira fase todos os filtros para averiguar se a metodologia funciona para qualquer conjunto de dados, mesmo para dados com muitos defeitos.

Relativamente à distribuição dos *scores* das viagens, nota-se que dependendo da duração das viagens obtem-se intervalos diferentes de *scores* como consta nas Figuras 5.15, 5.16, 5.17, 5.18:

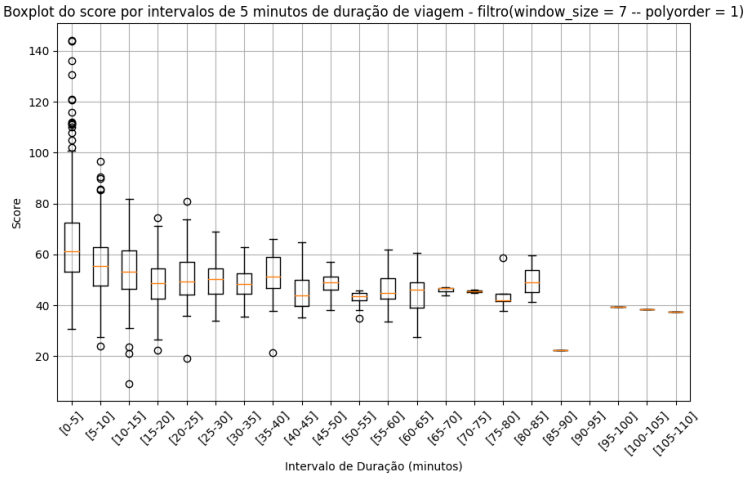

Figura 5.15: Boxplot do score das viagens cujo filtro aplicado foi de windows_size = 7 e polyorder = 1 por duração de tempo das viagens dividido em intervalos de 10 minutos

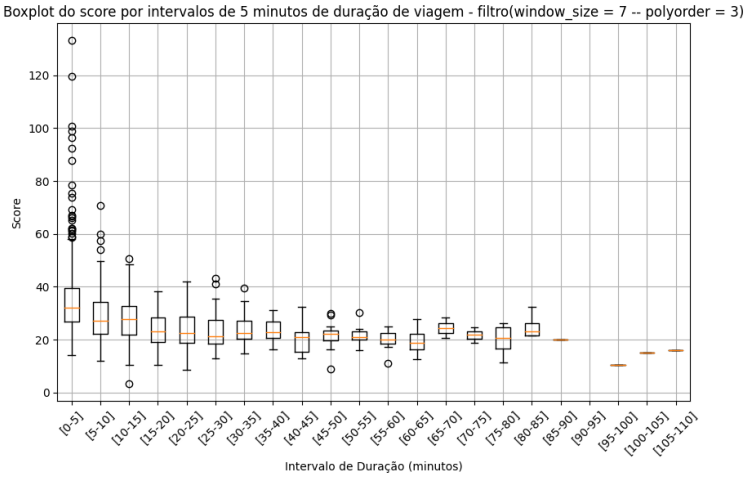

Figura 5.16: Boxplot do score das viagens cujo filtro aplicado foi de windows_size = 7 e polyorder = 3 por duração de tempo das viagens dividido em intervalos de 10 minutos.

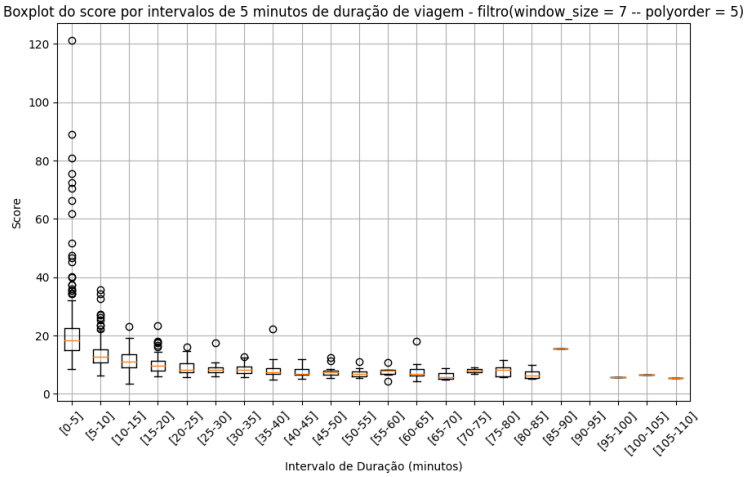

Figura 5.17: Boxplot do score das viagens cujo filtro aplicado foi de windows_size = 7 e polyorder = 5 por duração de tempo das viagens dividido em intervalos de 10 minutos

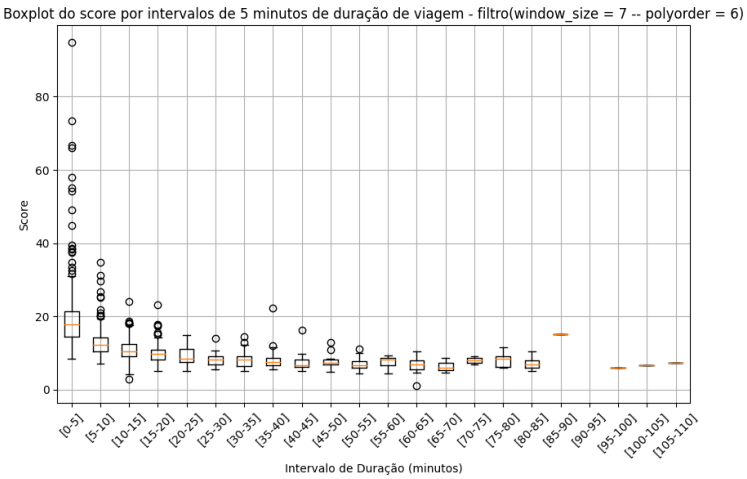

Figura 5.18: Boxplot do score das viagens cujo filtro aplicado foi de windows_size = 7 e polyorder = 6 por duração de tempo das viagens dividido em intervalos de 10 minutos

Pode-se observar nestas figuras que as viagens com durações abaixo dos 10 minutos têm intervalos de *scores* que se destacam das demais. Este fenómeno é justificado pelo facto da equação 2.4 levar em conta a frequência relativa dos padrões, fazendo com que geralmente os padrões presentes nos *rankings* de viagens curtas tenham uma frequência relativa superior à dos padrões presentes nos *rankings* de viagens longas, o que leva a que a contribuição de cada padrão para o *score* final seja mais alta, resultando neste fenómeno.

Para garantir que este fenómeno não afeta a análise que se pretende fazer, além de se analisar as viagens após aplicado diferentes filtros, também foram feitas a análise com as viagens com durações acima de 10 minutos e com todas as viagens. Para se classificar os dados foi utilizado o *score* para se dividir o *dataset* em duas classes diferentes, o agressivo e o não agressivo, onde essa divisão é feita escolhendo diferentes valores arbirtrários, a fim de se avaliar o desempenho dos modelos tendo em conta diferentes desbalanceamento de classes. Foram testados 5 valores *scores* diferentes escolhidos a partir de percentis, para que seja justa a comparação do desempenhos entre os diferentes modelos com diferentes filtros aplicados, visto que separar os dados desta forma leva a que os dados sejam separados da mesma forma, isto é, os dados são sempre divididos com o mesmo número de elementos de cada classe, para o mesmo percentil.

Como o *dataset* foi dividido em duas classes recorrendo ao *score*, mas ao mesmo tempo tem-se que a distribuição dos *scores* varia consoante a duração das viagens, com destaque nas viagens curtas abaixo de 10 minutos, então pretende-se analisar as viagens com durações acima de 10 minutos e com todas as viagens, separadamente. Também serão apresentado esses resultados as viagens com diferentes filtros aplicados, pois esperam-se diferentes resultados dependendo da quantidade de ruído removido do sinal das viagens.

5.2 Resultados dos modelos

A análise foi inicializada com *crossvalidation* com 10 *folds*, com o crescimento das árvores de decisão limitados a uma profundidade de 7 e um máximo de folha de 10 a fim de se averiguar o desempenho da *accuracy* para das diferentes restrições e diferentes filtros. Estas limitações impostas no crescimento das árvores de decisão deve-se ao facto de ser esse o tamanho onde o desempenho das mesmas no conjunto de treino e no conjunto de teste são diferentes, mas não muito diferentes a ponto de não configurar *overfitting*. Esta abordagem tem como objetivo de selecionar as restrições e os filtros onde os modelos são mais viáveis e portanto que têm melhor desempenho.

Para apresentar os resultados desta primeira fase de análise foram apresentadas tabelas onde constam o número de classes dos *datasets* e os diferentes desempenhos para as respetivas restrições aos dados disponíveis, com as Tabelas 5.1, 5.3, 5.5, 5.7, 5.9 que contêm as contagens das classes por cada percentil diferente e as Tabelas 5.2, 5.4, 5.6, 5.8, 5.10 que contêm o resultados da *accuracy* obtida do *crossvalidation* por cada percentil também:

Viagens com mais de 10 minutos		Todas as viagens	
agressivo	não agressivo	agressivo	não agressivo
411	103	936	224

Tabela 5.1: Tabela com as contagens das classes dos *datasets* divididos através do valor correspondente ao percentil 0.2

	mais de 10 minutos		todas as viagens	
	test set score	training set score	test set score	training set score
polyorder = 1	0.80	0.85	0.81	0.84
polyorder = 3	0.81	0.87	0.81	0.84
polyorder = 5	0.78	0.81	0.79	0.82
polyorder = 6	0.77	0.82	0.79	0.81

Tabela 5.2: Tabela com os resultados do *crossvalidation* com 10 *folds* para uma divisão dos dados disponíveis através do valor correspondente ao percentil 0.2

Viagens com mais de 10 minutos		Todas as viagens	
agressivo	não agressivo	agressivo	não agressivo
308	206	702	468

Tabela 5.3: Tabela com as contagens das classes dos *datasets* divididos através do valor correspondente ao percentil 0.4

	mais de 10 minutos		todas as viagens	
	test set score	training set score	test set score	training set score
polyorder = 1	0.71	0.80	0.72	0.77
polyorder = 3	0.72	0.78	0.72	0.76
polyorder = 5	0.67	0.75	0.69	0.74
polyorder = 6	0.68	0.73	0.68	0.71

Tabela 5.4: Tabela com os resultados do *crossvalidation* com 10 *folds* para uma divisão dos dados disponíveis através do valor correspondente ao percentil 0.4

Viagens com mais de 10 minutos		Todas as viagens	
agressivo	não agressivo	agressivo	não agressivo
257	257	585	585

Tabela 5.5: Tabela com as contagens das classes dos *datasets* divididos através do valor correspondente ao percentil 0.5

	mais de 10 minutos		todas as viagens	
	test set score	training set score	test set score	training set score
polyorder = 1	0.69	0.80	0.70	0.75
polyorder = 3	0.69	0.77	0.72	0.76
polyorder = 5	0.69	0.78	0.71	0.75
polyorder = 6	0.70	0.75	0.66	0.71

Tabela 5.6: Tabela com os resultados do *crossvalidation* com 10 *folds* para uma divisão dos dados disponíveis através do valor correspondente ao percentil 0.5

Viagens com mais de 10 minutos		Todas as viagens	
agressivo	não agressivo	agressivo	não agressivo
206	308	468	702

Tabela 5.7: Tabela com as contagens das classes dos *datasets* divididos através do valor correspondente ao percentil 0.6

	mais de 10 minutos		todas as viagens	
	test score	training set score	test set score	training set score
polyorder = 1	0.73	0.81	0.69	0.75
polyorder = 3	0.73	0.81	0.71	0.76
polyorder = 5	0.78	0.83	0.73	0.77
polyorder = 6	0.78	0.82	0.73	0.75

Tabela 5.8: Tabela com os resultados do *crossvalidation* com 10 *folds* para uma divisão dos dados disponíveis através do valor correspondente ao percentil 0.6

Viagens com mais de 10 minutos		Todas as viagens	
agressivo	não agressivo	agressivo	não agressivo
103	411	234	936

Tabela 5.9: Tabela com as contagens das classes dos *datasets* divididos através do valor correspondente ao percentil 0.8

	mais de 10 minutos		todas as viagens	
	test set score	training set score	test set score	training set score
polyorder = 1	0.81	0.87	0.79	0.83
polyorder = 3	0.77	0.87	0.81	0.85
polyorder = 5	0.89	0.94	0.83	0.87
polyorder = 6	0.89	0.93	0.82	0.86

Tabela 5.10: Tabela com os resultados do *crossvalidation* com 10 *folds* para uma divisão dos dados disponíveis através do valor correspondente ao percentil 0.8

Os resultados nesta primeira fase da análise, já são bastante positivos. A primeira conclusão é que independetemente dos dados utilizados, seja para diferentes filtros aplicados, seja para diferentes desbalanceamentos das classes ou até utilizando todas as viagens ou apenas aquelas com durações acima dos 10 minutos, não há nenhum valor de *accuracy* abaixo dos 60%, sendo que a maioria delas está acima dos 70%, que revela desempenhos promissores para validar a hipótese desta dissertação. O mínimo de desempenho obtido foi de 66% e o máximo foi de 89%, que sugere que em algum nível os padrões presentes nos *rankings* das viagens descrevem o perfil dos condutores.

Para os diferentes valores de percentis utilizados, aqueles nos quais se obteve melhor desempenho, foram nos modelos treinados com dados mais desbalanceados com os resultados presentes nas Tabelas 5.2 e 5.10, que correspondem aos percentis de 0.2 e 0.8 respetivamamente, nos quais o desempenho atinge valores à volta dos 80%.

Na Tabela 5.10, referente ao percentil 0.8, para as *polyorders* 5 e 6, nas viagens com duração acima dos 10 minutos, os desempenhos atingem valores excepcionais, com desempenhos à volta dos 90%, em comparação com as *polyorders* 1 e 3, que chegam a ter desempenhos com cerca de 10 valores percentuais a menos. Se for comparado aos resultados das Tabelas 5.2 e 5.4, referentes a percentis de 0.2 e 0.4 respetivamente, por exemplo, entre as *polyorders* 1 e 3 e as *polyorders* 5 e 6, os valores excepcionais de desempenho já não acontecem. Além disso a diferença no desempenho para as *polyorders* 5 e 6, entre os modelos treinados com as viagens de durações acima de 10 minutos e os modelos treinados com todas as viagens disponíveis, presentes nas Tabelas 5.8 e 5.10, referente aos percentis 0.6 e 0.8 respetivamente, são mais acentuadas que nas demais tabelas, que mostra menor robustez aos dados utilizados nos casos mencionados, logo, menor confiabilidade no resultado obtido nesse desempenho pois poderá ter sido apenas um acaso.

Na segunda fase da análise, será demonstrado concretamente os resultados da *polyorder* igual a 1, poderia ter sido a *polyorder* igual a 3 pois ambas têm desempenhos parecidos,

resultam em *accuracies* melhores de forma geral, além dos resultados serem mais confiáveis devido à robustez dos resultados para os diferentes conjuntos de dados utilizados, e do impacto indesejado que os filtros de *polyorder* 5 e 6 têm nos *rankings*. O objetivo é analisar o desempenho segundo outras métricas como *precision*, *recall* e *f1-score*, bem como observar o tamanho das árvores resultantes. Assim sendo serão criadas diferentes árvores de decisão, resultantes da melhoria após efetuado o *pruning*, que será executado visando atender a alguns critério como:

- menor profundidade possível
- menor menor numero de folhas possível
- evitar que duas folhas com origem no mesmo nó tenham a mesma classe
- evitar *overfitting* e *underfitting*

5.2.1 Modelos treinados com *datasets* com viagens acima dos 10 minutos

Os resultados apresentados a seguir dizem respeito aos modelos treinados com as viagens com durações a acima dos 10 minutos cujo filtro aplicado tinha a *polyorder* igual a 1. O *pruning* destas árvores já está feito e foi feito manualmente.

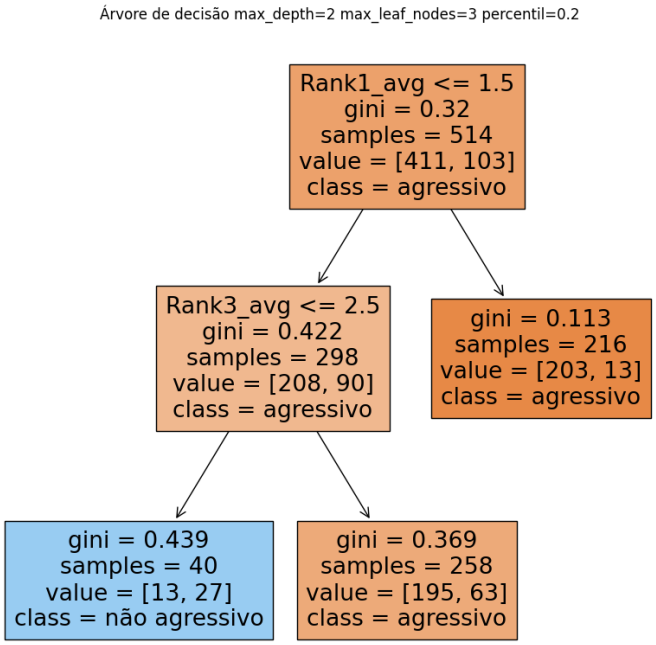

Figura 5.19: Árvore de decisão correspondente à divisão do *dataset* com viagens superiores a 10 minutos pelo percentil de 0.2

	precision	recall	f1-score	support
não agressivo	0.70	0.27	0.38	103
agressivo	0.84	0.97	0.90	411
accuracy	0.83			

Tabela 5.11: Relatório correspondente à árvore de decisão presente na Figura 5.19

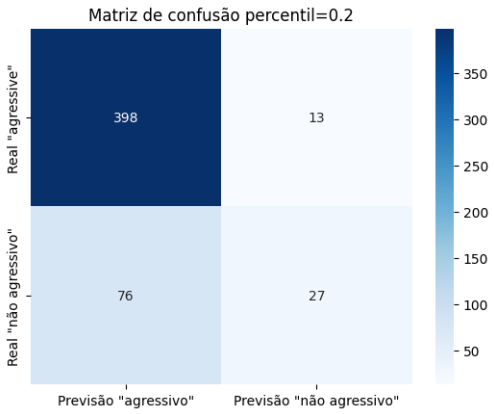

Figura 5.20: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.19

Para um percentil de 0.2 a àrvore mais eficiente é a presente na Figura 5.19 cuja a generalização está associada aos padrões da primeira e terceira posição, para uma profundidade de 2 e número de folhas igual a 3. A matriz de confusão 5.20 e a Tabela 5.11 mostram um alto desempenho de *precision* para ambas as classes, com uma tendência para a classe dominante. No caso do *recall* essa diferença já é bastante significativa, com um desempenho é de 97% para as classes agressivas e de 27% para as classes não agressivas.

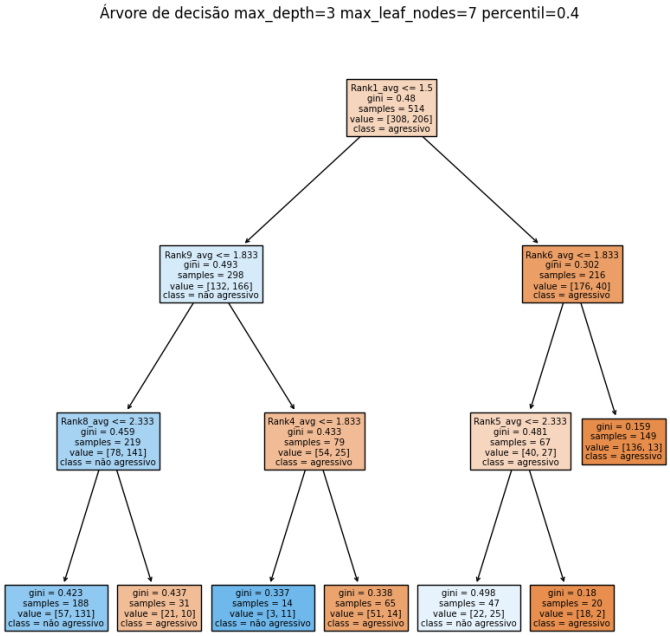

Figura 5.21: Árvore de decisão correspondente à divisão do *dataset* com viagens superiores a 10 minutos pelo percentil de 0.4

	precision	recall	f1-score	support
não agressivo	0.60	0.71	0.64	206
agressivo	0.78	0.70	0.73	308
accuracy	0.70			

Tabela 5.12: Relatório correspondente à árvore de decisão presente na Figura 5.21

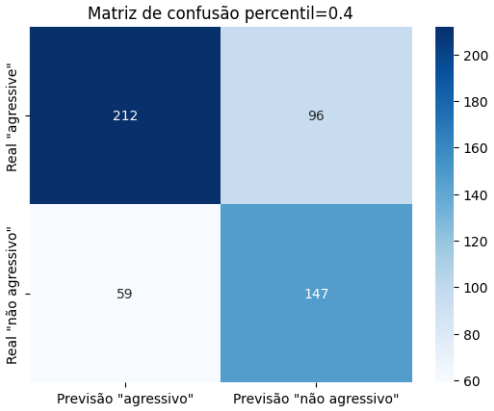

Figura 5.22: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.21

A árvore correspondente ao percentil 0.4 está presente na Figura 5.21, tem uma profundidade de 3 e número de folhas igual a 7, que consequentemente recorreu a mais padrões durante a classificação sendo estes, o primeiro, o quarto, o quinto, o sexto, o oitavo

e o nono. Além disso o desempenho em termos de *precision* é mais uma vez superior na classe dominante já o *recall* é aproximadamente igual para ambas as classes, como demonstrado na matriz de confusão 5.22 e na Tabela 5.12.

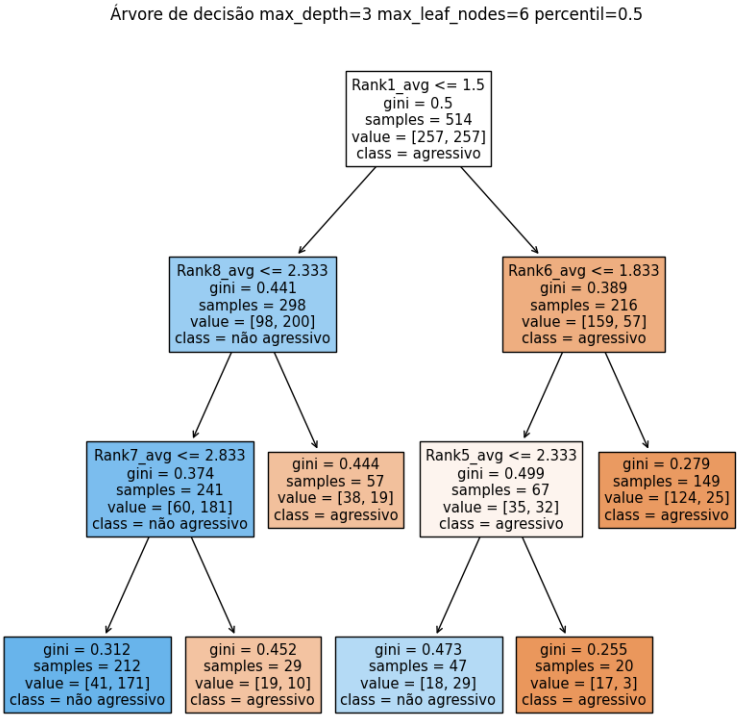

Figura 5.23: Árvore de decisão correspondente à divisão do *dataset* com viagens superiores a 10 minutos pelo percentil de 0.5

	precision	recall	f1-score	support
não agressivo	0.70	0.73	0.71	257
agressivo	0.70	0.70	0.69	257
accuracy			0.71	

Tabela 5.13: Relatório correspondente à árvore de decisão presente na Figura 5.23

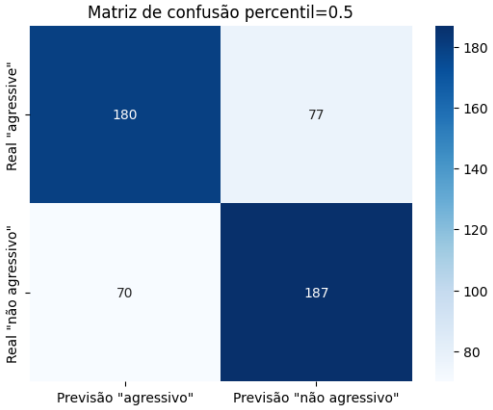

Figura 5.24: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.23

A árvore correspondente ao percentil 0.5 está presente na Figura 5.23, tem uma profundidade de 3 e número de folhas igual a 6 e recorreu ao primeiro, quinto, sexto, sétimo e oitavo padrão. Além disso o desempenho em termos de *precision* e de *recall* são parecidos para ambas as classes, à volta dos 70%, como consta na matriz de confusão 5.24 e na Tabela 5.13.

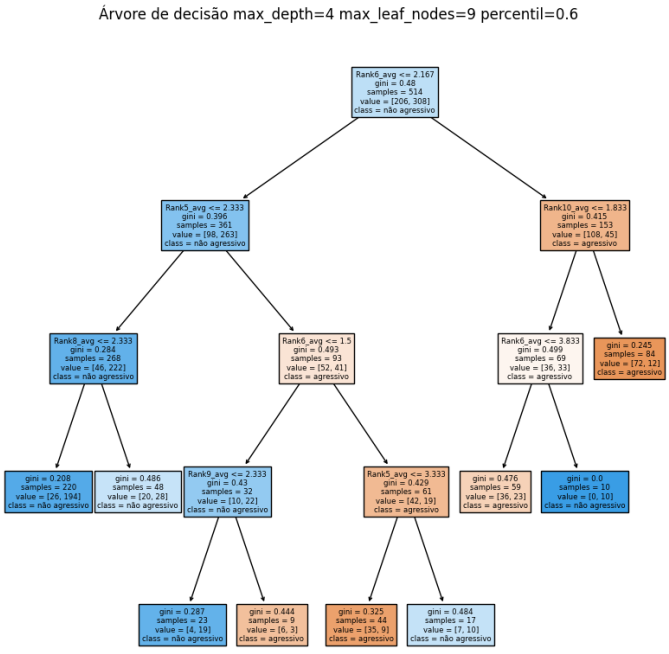

Figura 5.25: Árvore de decisão correspondente à divisão do *dataset* com viagens superiores a 10 minutos pelo percentil de 0.6

	precision	recall	f1-score	support
não agressivo	0.77	0.81	0.77	308
agressivo	0.67	0.66	0.65	206
accuracy	0.74			

Tabela 5.14: Relatório correspondente à árvore de decisão presente na Figura 5.25

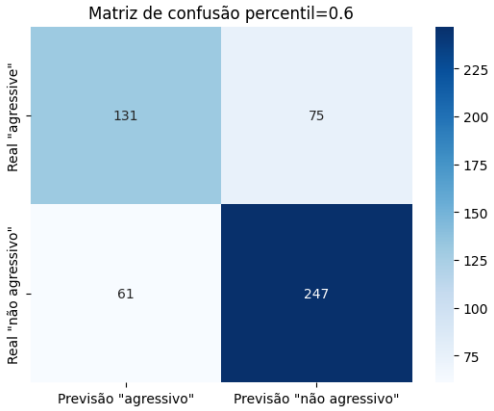

Figura 5.26: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.25

A árvore correspondente ao percentil 0.6 está presente na Figura 5.25 que é até agora a maior de todas, com profundidade de 4 e número de folhas igual a 9, que recorreu ao quinto, sexto, oitavo, nono e décimo padrão. O desempenho é positivo para ambas as classes com destaque para a classe não agressiva, que é a classe dominante, tanto na *precision* como no *recall*, como demonstra a matriz de confusão 5.26 e a Tabela 5.14.

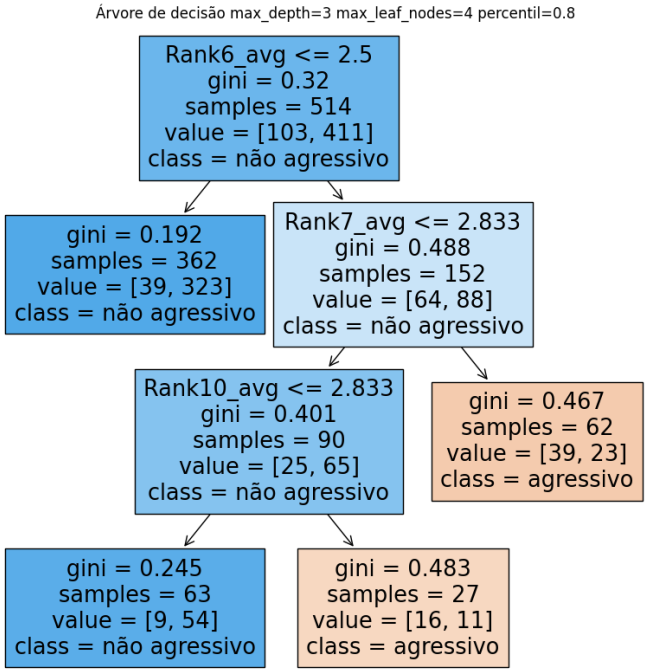

Figura 5.27: Árvore de decisão correspondente à divisão do *dataset* com viagens superiores a 10 minutos pelo percentil de 0.8

	precision	recall	f1-score	support
não agressivo	0.84	0.93	0.87	411
agressivo	0.52	0.34	0.37	103
accuracy			0.79	

Tabela 5.15: Relatório correspondente à árvore de decisão presente na Figura 5.27

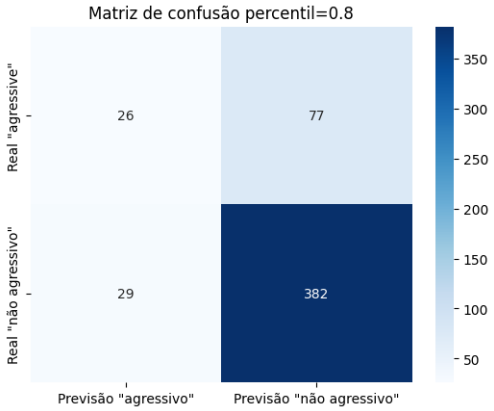

Figura 5.28: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.27

Para um percentil de 0.8 a àrvore mais eficiente econtrada é a apresentada na Figura 5.27, com profundidade de 3 e número de folhas igual a 4, cujos padrões utilizados são o sexto, sétimo e décimo, para uma profundidade de 2 e número de folhas igual a 4. A matriz

de confusão 5.28 e a Tabela 5.15 mostram um alto desempenho de *precision* e de *recall* para a classe não agressiva, com valores de 84% e 93%, respetivamente, e um desempenho não tão bom para a classe agressiva, com valores de 52% e de 34% respetivamente.

5.2.2 Modelos treinados com *datasets* com todas as viagens disponíveis

Os resultados apresentados a seguir dizem respeito aos modelos treinados com todas as viagens disponíveis cujo filtro aplicado tinha a *polyorder* igual a 1. O *pruning* destas árvores já está feito e foi feito manualmente.

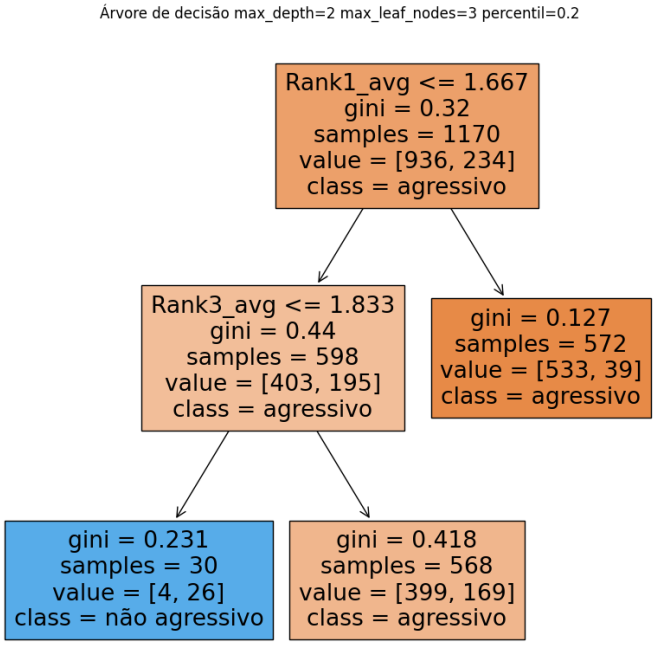

Figura 5.29: Árvore de decisão correspondente à divisão do *dataset* com todas as viagens disponíveis classificadas pelo percentil de 0.2

	precision	recall	f1-score	support
não agressivo	0.56	0.13	0.20	234
agressivo	0.82	0.97	0.89	936
accuracy	0.80			

Tabela 5.16: Relatório correspondente à árvore de decisão presente na Figura 5.29

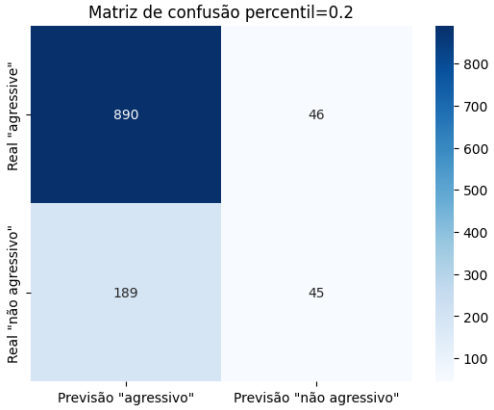

Figura 5.30: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.29

Para um percentil de 0.2 a àrvore está presente na Figura 5.29 é praticamente igual à árvore do modelo treinado apenas com as viagens superiores a 10 minutos 5.19, com uma profundidade de 2 e número de folhas igual a 3, contendo os mesmos padrões na sua composição, com a diferença que neste caso o desempenho tanto na *precision* quanto no *recall* é significativamente inferior para a classe das não agressivas, apesar de manter o desempenho para a classe das agressivas, como se pode verificar na matriz de confusão 5.30 e a Tabela 5.16.

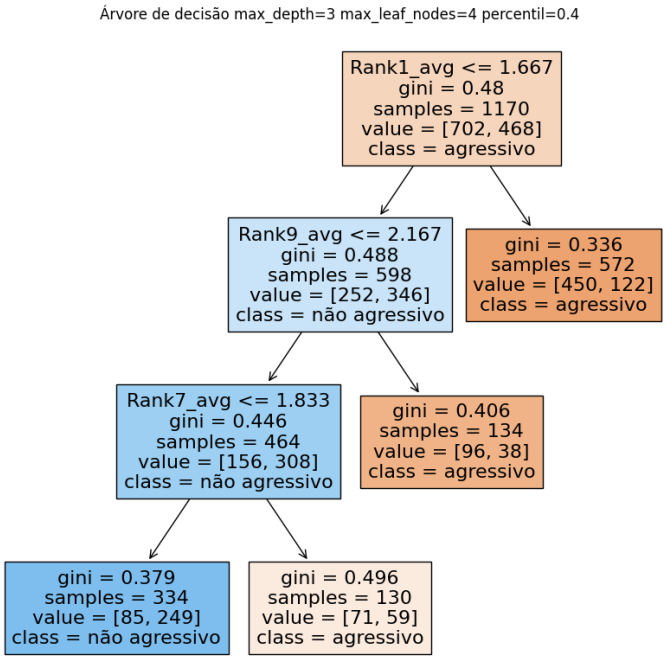

Figura 5.31: Árvore de decisão correspondente à divisão do *dataset* com viagens com todas as viagens disponíveis classificadas pelo percentil de 0.4

	precision	recall	f1-score	support
não agressivo	0.67	0.55	0.60	468
agressivo	0.73	0.82	0.77	702
accuracy	0.71			

Tabela 5.17: Relatório correspondente à árvore de decisão presente na Figura 5.31

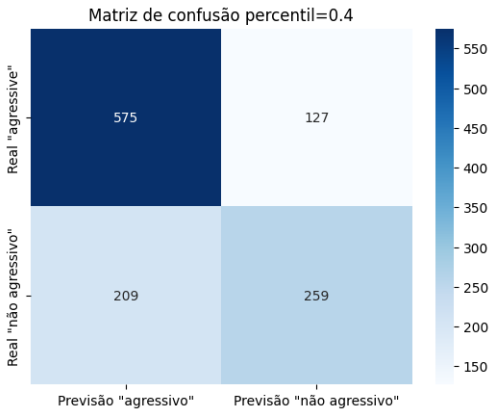

Figura 5.32: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.21

Para um percentil de 0.4 a àrvore está presente na Figura 5.31, tem uma profundidade de 3 e número de folhas igual a 4, e contém o primeiro, sétimo e nono na sua composição. O desempenho em termos de *precision* é parecido com o desempenho do modelo treinado apenas com as viagens superior a 10 minutos 5.21. Em termos de *recall* o modelo perde desempenho para as classes não agressivas e ganha para as classes agressivas, como podemos verificar na matriz de confusão 5.32 e an Tabela 5.17.

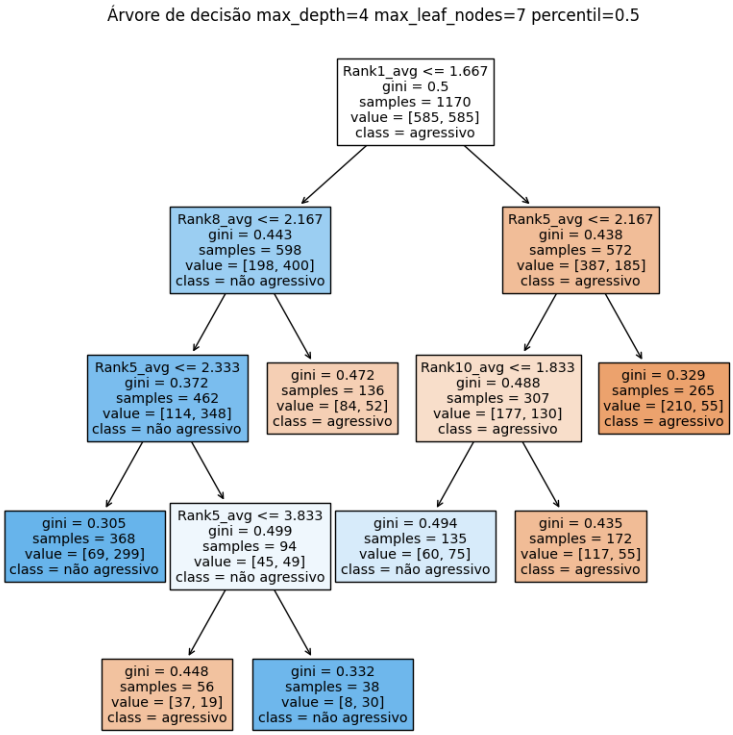

Figura 5.33: Árvore de decisão correspondente à divisão do *dataset* com todas as viagens disponíveis classificadas pelo percentil de 0.5

	precision	recall	f1-score	support
não agressivo	0.71	0.63	0.66	585
agressivo	0.67	0.75	0.70	585
accuracy	0.69			

Tabela 5.18: Relatório correspondente à árvore de decisão presente na Figura 5.33

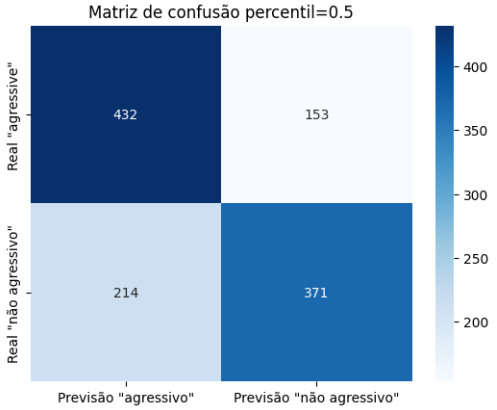

Figura 5.34: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.33

Para um percentil de 0.5 a àrvore está presente na Figura 5.33, tem uma profundidade

de 4 e número de folhas igual a 7 e contém o primeiro, quinto, oitavo e ao décimo padrão na sua composição. O desempenho em termos de *precision* e de *recall* é parecido com o desempenho do modelo treinado apenas com as viagens com duração superior a 10 minutos 5.23, com um perda significativa no *recall* da classe não agressiva, como podemos verificar na matriz de confusão 5.32 e a Tabela 5.18.

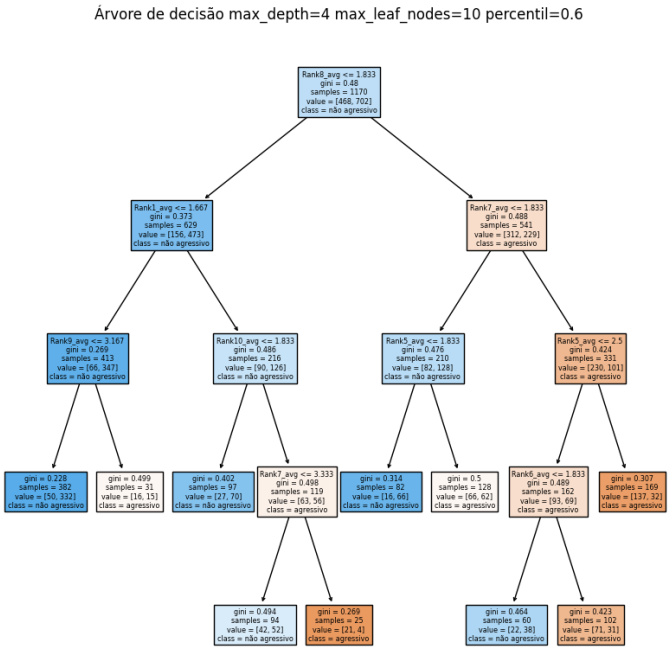

Figura 5.35: Árvore de decisão correspondente à divisão do *dataset* com todas as viagens disponíveis classificadas pelo percentil de 0.6

	precision	recall	f1-score	support
não agressivo	0.75	0.72	0.73	702
agressivo	0.61	0.64	0.62	468
accuracy		0.69		

Tabela 5.19: Relatório correspondente à árvore de decisão presente na Figura 5.35

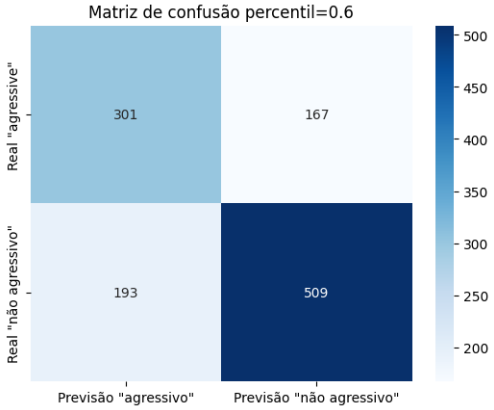

Figura 5.36: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.35

Para um percentil de 0.6 a àrvore está presente na Figura 5.35, e tem uma profundidade de 4 e número de folhas igual a 10, sendo a maior árvore gerada até agora, com o desempenho em termos de *precision* e de *recall* parecidos com o modelo treinado com as viagens com duração acima dos 10 minutos 5.25, com um ganho significativo no *recall* da classe não agressiva, como podemos verificar na matriz de confusão 5.36 e a Tabela 5.19.

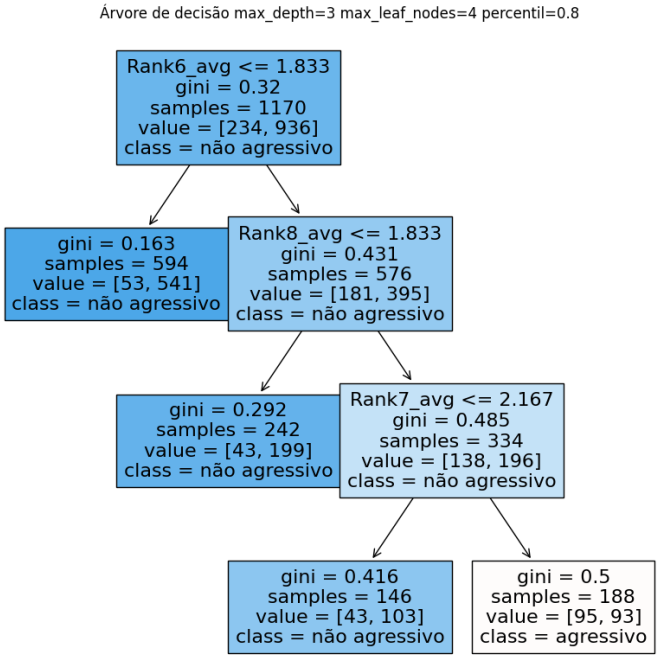

Figura 5.37: Árvore de decisão correspondente à divisão do *dataset* com todas as viagens disponíveis classificadas pelo percentil de 0.8

	precision	recall	f1-score	support
não agressivo	0.83	0.92	0.87	936
agressivo	0.28	0.34	0.25	234
accuracy	0.78			

Tabela 5.20: Relatório correspondente à árvore de decisão presente na Figura 5.37

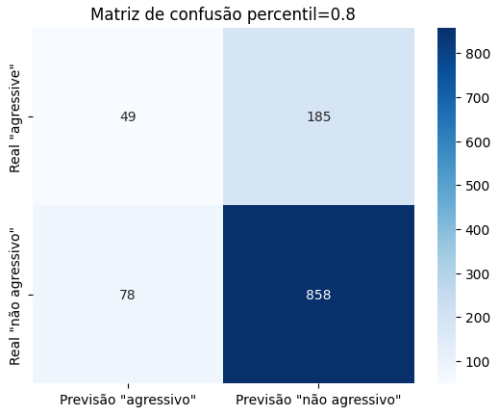

Figura 5.38: Matriz de confusão correspondente à árvore de decisão presente na Figura 5.27

Para um percentil de 0.8 a àrvore está presente na Figura 5.37, e tem uma profundidade de 3 e número de folhas igual a 4, voltando a ser uma árvore pequena, com o desempenho em termos de *precision* e de *recall* parecidos com o modelo treinado com as viagens com duração acima dos 10 minutos 5.27, com uma perda significativa na *precison* na classe agressiva, como podemos verificar na matriz de confusão 5.38 e a Tabela 5.20.

5.3 Análise dos resuldados obtidos

As árvores de decisão obtidas com melhor desempenho são as treinadas com os conjunto de dados mais desbalaceados, também são as de menor tamanho, o que as tornam mais eficientes a classificar.

Assumindo que as seguradoras beneficiam-se de que os seus clientes não acionem o seguro, estas entidades estão interessadas em identificar os condutores não agressivos, para facilitar o processo de encontrar possíveis clientes, minimizando os falsos positivos (condutores agressivos classificados como não agressivos), ou garantir que identificam corretamente os condutores agressivos, a fim de minimizar a possibilidade de se concretizar maus negócios, minimizando os falsos negativos (condutores agressivos classificados como não agressivos). No primeiro caso é pertinente maximizar a *precision* pois esta métrica indica a proporção de viagens não agressivas previstas pelo classificador que realmente são não agressivas. No segundo caso já é mais pertinente maximizar o *recall* pois é a métrica que indica qual é a proporção de viagens agressivas que foram corretamente identificadas. Em ambos os casos as seguradoras benefeciariam-se.

De forma geral as árvores com melhor desempenho são as árvores treinadas com dados com classes mais desbalaceadas. Estas árvores são pequenas o que se traduzem em eficiência na classificação.

As árvores de decisão com o melhor desempenho em termos de *precision* na classe não agressiva são as presentes nas Figuras 5.27 e 5.37, que correspondem às árvores contruídas com dados divididos pelos percentis de 0.8, treinadas com as viagens com durações acima dos 10 minutos e com todas as viagens disponíveis respetivamamente, com o valores de desempenho à volta dos 83%. Não deixar de reparar no desempenho em termos de *recall* estão à volta dos 92%. A árvore treinada com as viagens com durações acima dos 10 minutos tem uma precisão nas classes agressiva de 52%, consideravelmente superior à árvore treinada com todas as viagens disponíveis que tem um valor de 28%.

As árvores de decisão com o melhor desempenho em termos de *recall* na classe agressiva são as presentes nas Figuras 5.19 e 5.29, que correspondem às árvores contruídas com dados divididos pelos percentis de 0.2, treinadas com as viagens com durações acima dos 10 minutos e com todas as viagens disponíveis respetivamamente, com o valores de desempenho iguais a 97%. Não deixar de reparar que o desempenho das classes em termos de *precision* para a árvore treinada com as viagens superiores a 10 minutos são de 70% e de 84%, para as classes não agressivas e agressivas respetivamente, já para as a árvore treinada com todas as viagens disponíveis foi de 56% e 82%. Pode-se assumir que é preferível treinar os modelos com dados referentes a viagens longas, que é coerente, porque quanto mais dados for utilizado para gerar os *rankings*, mais os padrões convergem.

Estes resultados mostram que não só é possível utilizar as árvores de decisão para modelar as viagens classificadas em duas categorias diferentes, segundo o método proposto, de modo a classificá-las entre agressivas e não agressivas, bem como mostra que é possível obter árvores que tenham desempenhos ótimos em termos de *precision* ou em termos de *recall*, atendendo a diferentes aplicações da metodologia. Isto indica que assumindo que os dados processados digam respeito a padrões que definam um condutor que um desempenho parecido será obtido, provando a hipótese que se pretendia.

6

Conclusão

Esta dissertação visava testar a hipótese de que é possível modelar a condução de condutores. Para tal foi desenvolvido um sistema de recolha e analise de dados, cuja implementação foi inspirada no estudo [5] que desenvolveu uma maneira de distinguir os condutores em termos de agressividade, através de *scores*.

Este projeto resultou numa aplicação *Android* integrada com os serviços *Google Cloud* que serve o propósito de coletar dados e armazená-los para serem analisados.

Essa análise passa por converter as viagens disponíveis em *rankings* e classificá-las com o auxílio do *score* obtido pela equação 2.4, como viagens agressivas e não agressivas e a partir daí, o objetivo foi de provar que é possível treinar árvores de decisão com informação presente nesses *rankings* e obter modelos com desempenhos que mostrem que é possível modelar as viagens agressivas e não agressivas, permitindo que se infira que é possível obter resultados semelhantes caso os os modelos sejam treinados com informação relativa a condutores que estajam classificados como agressivos e não agressivos.

Os resultados são satisfatórios, com os melhores modelos com desempenhos em termos de *accuracy*à volta dos 80%. O melhor modelo em termos de *precision* tem desempenho de 83% para a classe não agressiva, enquanto que o melhor modelo em termos de *recall* tem desempenho 97% para a classe agressiva. Estes dois modelos são vantajosos à sua maneira, o primeiro é melhor em detetar viagens não agressivas enquanto que o segundo é melhor em garantir que deteta as viagens agressivas. Ambas as árvores de decisão são pequenas o que se traduz em eficiência em classificar.

6.1 Trabalhos futuros

Num trabalho futuro para garantir que os resultados são realmente aplicáveis como ferramenta de seguradoras, deverá ser repetida a experiência com dados de melhor qualidade, maior quantidade em termos de duração das viagens, além dos dados deverem estar enquadrados nalgum contexto em específico. As viagens gravadas deverão estar separadas por condutor e assim toda o processo de classificação deverá ser relativo aos condutores e não às viagens.

Para complementar o todo o processo, a *pipeline* poderá ser melhorada para que os *smartphones* apenas enviem a sequência de símbolos do *ranking* com a informação sobre o perfil de cada condutor. Do lado dos serviços *Google Cloud* a implementação deverá consitir na automatização tanto do treino dos modelos em função dos dados que chegam ao armanezamento, quanto na previsão de novas instâncias.

Naturalmente qualquer iniciativa que visa obter melhores desempenhos também são uma mais valia, desde experimentar outros modelos, a utilizar outras formas de classificar, utilizar outros dados de entrada para treinar os modelos ou até aumentar o número de classes.

Bibliografia

[1] A. Belgium. *AXA drive*. [Accessed 17-03-2024]. 2013. URL: //www.axa.be/ab/FR/apps/Pages/Applications-mobiles.aspx (ver p. 2).

[2] D. J. Bemdt e J. Clifford. *Using Dynamic Time Warping to Find Patterns in Time Series*. 1994. URL: www.aaai.org (ver p. 13).

[3] L. M. Bergasa et al. «DriveSafe: An app for alerting inattentive drivers and scoring driving behaviors». Em: Institute of Electrical e Electronics Engineers Inc., 2014, pp. 240–245. ISBN: 9781479936380. DOI: 10.1109/IVS.2014.6856461 (ver p. 16).

[4] G. Castignani et al. «Driver behavior profiling using smartphones: A low-cost platform for driver monitoring». Em: *IEEE Intelligent Transportation Systems Magazine* 7 (1 2015-01), pp. 91–102. ISSN: 19391390. DOI: 10.1109/MITS.2014.2328673 (ver p. 2).

[5] C. Chen et al. «A graphical modeling method for individual driving behavior and its application in driving safety analysis using GPS data». Em: *Transportation Research Part F: Traffic Psychology and Behaviour* 63 (2019), pp. 118–134. ISSN: 1369-8478. DOI: https://doi.org/10.1016/j.trf.2019.03.017. URL: https://www.sciencedirect.com/science/article/pii/S1369847818308428 (ver pp. 4, 5, 11, 12, 20, 31, 37, 38, 66).

[6] S. F. M. A. I. Company. *State Farm driver feedback*. [Accessed 17-03-2024]. URL: Available:%20http://www.statefarm.ca/about/mobile/ (ver p. 2).

[7] T. Dietterich et al. *Introduction to Machine Learning Second Edition Adaptive Computation and Machine Learning*. 2012 (ver pp. 17–19).

[8] *EcoSense dataset*. [Accessed 17-03-2024]. URL: https://mcloud.de/web/guest/suche/-/results/suche/relevance/ecosense/0/detail/694F2C6D-E80C-4EF0-9A1A-101D0FCFDE7A (ver pp. 21, 41).

[9] H. Eren et al. «Estimating driving behavior by a smartphone». Em: 2012, pp. 234–239. ISBN: 9781467321198. DOI: 10.1109/IVS.2012.6232298 (ver pp. 4, 6).

[10] *GitHub - BasicAirData/GPSLogger: A GPS logger for Android mobile devices — github.com*. https://github.com/BasicAirData/GPSLogger/tree/master. [Accessed 17-03-2024] (ver pp. 23, 33).

[11] Greenroad. *Introducing the revolutionary GreenRoad Smartphone Edition*. [Accessed 17-03-2024]. 2013. URL: http://greenroad.com/uk/tour/smartphone/ (ver p. 2).

[12] J. A. Healey e R. W. Picard. «Detecting stress during real-world driving tasks using physiological sensors». Em: *IEEE Transactions on Intelligent Transportation Systems* 6 (2 2005-06), pp. 156–166. ISSN: 15249050. DOI: 10.1109/TITS.2005.848368 (ver pp. 4, 6).

[13] S. Hochreiter e J. Schmidhuber. «Long Short-Term Memory». Em: *Neural Computation* 9 (8 1997-11), pp. 1735–1780. ISSN: 08997667. DOI: 10.1162/neco.1997.9.8.1735 (ver p. 15).

[14] D. A. Johnson e M. M. Trivedi. *Driving Style Recognition Using a Smartphone as a Sensor Platform*. IEEE, 2011. ISBN: 9781457721977. DOI: 10.1109/ITSC.2011.6083078 (ver pp. 4, 7–9).

[15] D. Kifokeris e Y. Xenidis. «Risk source-based constructability appraisal using supervised machine learning». Em: *Automation in Construction* 104 (2019), pp. 341–359 (ver p. 19).

[16] T. Kohonen. «The self-organizing map». Em: *Proceedings of the IEEE* 78 (9 1990), pp. 1464–1480. ISSN: 00189219. DOI: 10.1109/5.58325 (ver p. 13).

[17] S. Lawrence e C. Giles. «Overfitting and neural networks: conjugate gradient and backpropagation». Em: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*. Vol. 1. 2000, 114–119 vol.1. DOI: 10.1109/IJCNN.2000.857823 (ver p. 18).

[18] J. Lin et al. «Experiencing SAX: A novel symbolic representation of time series». Em: *Data Mining and Knowledge Discovery* 15 (2 2007-10), pp. 107–144. ISSN: 13845810. DOI: 10.1007/s10618-007-0064-z (ver p. 5).

[19] J. Lin et al. «Finding Motifs in Time Series». Em: *The 2nd Workshop on Temporal Data Mining, the 8th ACM Int'l Conference on KDD*. 2002 (ver p. 30).

[20] C. Lopez et al. «An unsupervised machine learning method for discovering patient clusters based on genetic signatures». Em: *Journal of biomedical informatics* 85 (2018), pp. 30–39 (ver p. 19).

[21] M. Mohri, A. Rostamizadeh e A. Talwalkar. *Foundations of Machine Learning* (ver pp. 2, 17, 18).

[22] E. Ohn-Bar et al. «Head, eye, and hand patterns for driver activity recognition». Em: Institute of Electrical e Electronics Engineers Inc., 2014-12, pp. 660–665. ISBN: 9781479952083. DOI: 10.1109/ICPR.2014.124 (ver p. 6).

[23] E. E. Ozbas et al. «Hydrogen production via biomass gasification, and modeling by supervised machine learning algorithms». Em: *International Journal of Hydrogen Energy* 44.32 (2019), pp. 17260–17268 (ver p. 19).

[24] A. PLC. *Aviva RateMyDrive*. [Accessed 17-03-2024]. URL: http:%20//www.aviva.co.uk/drive/ (ver p. 2).

[25] V. Roman. *Unsupervised Learning: Dimensionality Reduction — towardsdatascience.com*. https://towardsdatascience.com/unsupervised-learning-dimensionality-reduction-ddb4d55e0757. [Accessed 06-02-2024] (ver p. 19).

[26] E. Romera, L. M. Bergasa e R. Arroyo. «Need data for driver behaviour analysis? Presenting the public UAH-DriveSet». Em: Institute of Electrical e Electronics Engineers Inc., 2016-12, pp. 387–392. ISBN: 9781509018895. DOI: 10.1109/ITSC.2016.7795584 (ver pp. 14, 15).

[27] H. Sakoe e S. Chiba. «Dynamic Programming Algorithm Optimization for Spoken Word Recognition». Em: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26 (1 1978). Referencia de uma area que tem algoritmos uteis para se devenvolver algoritmos para deteção de driving behaviour, pp. 43–49. ISSN: 00963518. DOI: 10.1109/TASSP.1978.1163055 (ver p. 6).

[28] K. Saleh, M. Hossny e S. Nahavandi. «Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks». Em: Institute of Electrical e Electronics Engineers (IEEE), 2018-07, pp. 1–6. DOI: 10.1109/itsc.2017.8317835 (ver pp. 4, 5, 15).

[29] M. Savelonas et al. «Hybrid Time-series Representation for the Classification of Driving Behaviour». Em: Institute of Electrical e Electronics Engineers Inc., 2020-10. ISBN: 9781728159195. DOI: 10.1109/SMAP49528.2020.9248460 (ver pp. 16, 17).

[30] M. A. Shipp et al. «Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning». Em: *Nature medicine* 8.1 (2002), pp. 68–74 (ver p. 19).

[31] M. I. Silva e R. Henriques. «Exploring time-series motifs through DTW-SOM». Em: (2020-04). [Accessed 17-03-2024]. URL: http://arxiv.org/abs/2004.08176 (ver pp. 4, 11, 13).

[32] M. I. Silva e R. Henriques. «TripMD: Driving patterns investigation via Motif Analysis». Em: (2020-07). URL: http://arxiv.org/abs/2007.03727 (ver pp. 4, 11, 13, 14).

[33] Y. Tanaka. *Discovery of Time-Series Motif from Multi-Dimensional Data Based on MDL Principle*. 2005, pp. 269–300 (ver p. 11).

[34] V. Vaitkus, P. Lengvenis e G. Žylius. «Driving style classification using long-term accelerometer information». Em: Institute of Electrical e Electronics Engineers Inc., 2014-11, pp. 641–644. ISBN: 9781479950812. DOI: 10.1109/MMAR.2014.6957429 (ver pp. 4, 5, 8, 10).

[35] L. Wei et al. «Multi-source information fusion for drowsy driving detection based on wireless sensor networks». Em: 2013, pp. 850–857. ISBN: 9781467352215. DOI: 10.1109/ICSensT.2013.6727771 (ver pp. 5, 6).