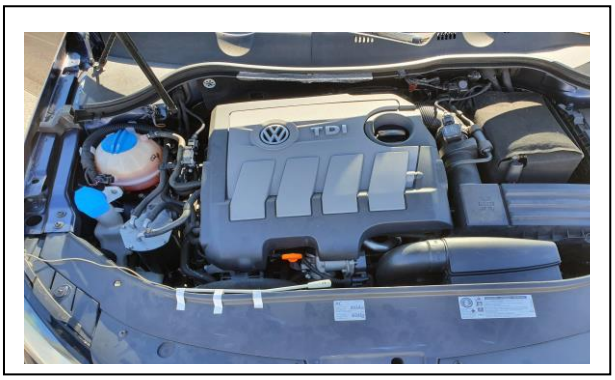
AutoMotorHelp - Deteção de falhas no motor usando o som

Projeto P025



Relatório Final da Unidade Curricular de Projeto

Licenciatura em Engenharia de Informática, Redes e Telecomunicações

Área Departamental de Eng. Eletrónica e Telecomunicações e Computadores Instituto Superior de Engenharia de Lisboa

Estudantes:
Daniel Oleiro – nº 46417
Diogo Silva – nº 46420

Orientador(es):
Prof. Joel Paulo
Prof. Paulo Trigo

Março 2021

Resumo

Aproxima-se a era em que veículos autónomos serão uma realidade comum. Provavelmente, nessa época será solicitado que estes veículos possam não só serem conduzidos sozinhos como também tenham processos de autodiagnóstico para avaliação do estado de funcionamento dos vários componentes. A esta necessidade futura, acrescentam-se problemas mais atuais, como o facto da maior parte dos condutores que realizam a conservação da sua viatura fazem-no através de duas formas de manutenção: a manutenção preventiva, na qual se perde tempo e fundos por muitas vezes ser desnecessária, ou a manutenção corretiva, que para além de não permitir a segurança dos utilizadores do veículo, também pode dar origem a custos muito elevados.

Tendo como objetivo a segurança dos passageiros e diminuição dos custos das manutenções dos veículos, os fabricantes começam já a dotar os automóveis com sistemas sofisticados por forma que as viaturas se desloquem até às oficinas apenas quando necessário, sem que estas tenham problemas acrescidos por falta da manutenção devida. Neste sentido, é necessário normalizar a manutenção baseada na verificação constante das condições físicas dos equipamentos, usando a mais variada informação sobre o estado dos vários componentes do veículo.

Este projeto pretende, através de técnicas de processamento de áudio e de técnicas de aprendizagem automática (usando redes neuronais) realizar uma análise dos sons produzidos pelo motor dos veículos de forma a identificar anomalias. Os resultados dessa análise são apresentados e armazenados em tempo real numa plataforma centralizada, para posterior análise e visualização. Esta mantém um histórico sobre o veículo e possibilita ouvir as anomalias produzidas pelo motor do mesmo.

As análises de resultados experimentais mostraram que a separação de eventos auxilia o classificador na sua análise. Nas experiências foram realizados testes nos quais se comparou a avaliação do classificador com segmentos, sempre com a mesma duração, e a avaliação do mesmo para eventos fornecidos pelo separador de eventos.

Palavras-chave – Inteligência artificial; Aprendizagem Automática; Análise de áudio; Dataset de Eventos Sonoros; Manutenção preditiva em veículos motorizados; Redes Neuronais

Abstract

The era is approaching when autonomous vehicles will become a common reality. Probably, at that time, it will be requested that these vehicles can not only be driven alone, but also have self-diagnosis processes to assess the functioning status of the various components. To this future need, more current problems are added, such as the fact that most drivers who preserve their vehicle do so through two forms of maintenance: preventive maintenance, in which time and funds are wasted because it is often unnecessary, or corrective maintenance, which in addition to not allowing the safety of vehicle users can also result in very high costs.

Aiming at passenger safety and reducing vehicle maintenance costs, manufacturers are already beginning to provide cars with sophisticated systems so that vehicles travel to workshops only when necessary, without these having added problems due to lack of proper maintenance. In this sense, it is necessary to standardize maintenance based on constant verification of the physical condition of the equipment, using the most varied information on the status of the various components of the vehicle.

This project intends, through audio processing techniques and automatic learning techniques (using neural networks) to carry out an analysis of the sounds produced by the vehicle engine to identify anomalies. The results of this analysis are presented and stored in real time on a centralized platform, for further analysis and visualization. This keeps a history of the vehicle and makes it possible to hear the anomalies produced by its engine.

The analysis of experimental results showed that the separation of events helps the classifier in its analysis. In the experiments, tests were carried out in which the evaluation of the classifier with segments, always with the same duration, was compared to its evaluation for events provided by the events separator.

Keywords - Artificial intelligence; Automatic Learning; Audio analysis; Sound Events Dataset; Predictive maintenance on motor vehicles; Neural Networks

Agradecimentos

Agradecemos a todos os orientadores por nos terem dado a oportunidade de realizar este projeto. E, por ao longo deste, nos terem disponibilizado recursos e tempo para o desenvolvimento e conclusão do mesmo.

A todos os professores, que durante o curso, nos deram uma base para poder realizar este projeto e que nos introduziram o gosto pela análise de áudio.

Índice

Resumo ..... **i**

Abstract ..... **iii**

Agradecimentos ..... **v**

Índice ..... **vii**

Lista de Acrónimos ..... **ix**

Lista de Tabelas ..... **xi**

Lista de Figuras ..... **xiii**

1 Introdução **1**

2 Trabalho Relacionado **3**

3 Modelo Proposto **5**

3.1 Fundamentos **5**

3.1.1 Características Sonoras **5**

3.1.2 Redes Neuronais **13**

3.2 Abordagem **14**

4 Implementação do Modelo **17**

4.1 Processamento **17**

4.2 Separação de Eventos **20**

4.3 Classificação **21**

4.4 Plataforma **24**

5 Validação e Testes **27**

6 Conclusões e Trabalho Futuro **39**

A Utilização de Códigos **41**

B Contacto com Entidades Externas **43**

Bibliografia **45**

Lista de Acrónimos

CNN	Convolutional Neural Network
FFT	Fast Fourier Transform
IoT	Internet of Things
MFC	Mel Frenquency Cepstrum
MFCCs	Mel Frequency Cepstral Coefficients
RMS	Root Mean Square
rmsTh	Root Mean Square Threshold
sFTh	Spectral Flux Threshold
SQL	Structured Query Language

Lista de Tabelas

Tabela 5.1 Percentagem de acertos dos Modelos testados 34

Tabela 5.2 Tempo demorado pelos modelos a classificar um som de dez segundos 34

Tabela 5.3 Classificação de anomalias pelos modelos..... 35

Lista de Figuras

Figura 3.1 Multiplicação de sinais com diferentes frequências 6

Figura 3.2 *Zero Crossing Rate* aplicada a diferentes sinusoides 6

Figura 3.3 RMS aplicada a um som de motor 7

Figura 3.4 Curvas de *Rolloff* nas diferentes percentagens de *roll* 8

Figura 3.5 *Spectral Centroid* e *Spectral Rolloff* com *roll* 50% com os mesmos valores 9

Figura 3.6 *Spectral Centroid* e *Spectral Rolloff* com *roll* 50% com valores diferentes 9

Figura 3.7 *Spectral Flux* aplicada a diferentes amplitudes e frequências 10

Figura 3.8 Visualização da *Spectrogram* 10

Figura 3.9 Escala *Mel* 11

Figura 3.10 Visualização da *Mel-Spectrogram* 11

Figura 3.11 Visualização dos MFCCs 12

Figura 3.12 Visualização da *Chromagram* 13

Figura 3.13 Notas na escala musical 13

Figura 3.14 Representação do modelo *Feedforward Neural Network* 14

Figura 3.15 Objetivo final do projeto simplificado 15

Figura 4.1 Funcionamento das *threads* da plataforma implementada 17

Figura 4.2 Normalização de *features* no fim do processamento de sinal 19

Figura 4.3 Normalização de *features* durante o processamento de sinal 19

Figura 4.4 Diagrama UML da Máquina de Estados 20

Figura 4.5 Recolha de som de um carro movido a gasóleo 23

Figura 4.6 Recolha de som de um carro movido a gasolina 23

Figura 4.7 Página inicial da aplicação Web 24

Figura 4.8 Regras implementadas na *storage* do Firebase 25

Figura 5.1 *Features* aplicadas a um sinal de um carro a iniciar duas vezes 27

Figura 5.2 *Features* aplicadas a um carro a desligar 28

Figura 5.3 *Features* aplicadas a um som de motor com buzinas 29

Figura 5.4 *Features* aplicadas a um carro a acelerar 30

Figura 5.5 *Spectral Centroid* e *Spectral Rolloff* aplicadas a um carro a mudar de mudanças 31

Figura 5.6 *Spectral Rolloff* com *roll* de 85% aplicada a um som de motor com buzinas 31

Figura 5.7 *Spectral Rolloff* com *roll* de 99% aplicada a um som de motor com buzinas 32

Figura 5.8 *Spectral Rolloff* aplicada a um carro movido a gasolina 32

Figura 5.9 *Spectral Rolloff* aplicada a um carro movido a gasóleo 33

Figura 5.10 Cálculo da *precision* e da *recall* 33

Figura 5.11 Anomalias identificadas corretamente 36

Figura 5.12 Anomalias identificadas pelo classificador sem separação de eventos 36

Figura 5.13 Anomalias identificadas pelo classificador com separação de eventos 37

Figura B.1 E-mail enviado à oficina da EVolution - Service & Battery 43

Capítulo 1
Introdução

Imagine-se o dia, em que veículos autónomos, podem marcar as suas próprias idas à oficina ou o dia no qual os condutores não precisam de estar constantemente preocupados com o estado do funcionamento do seu automóvel. Uma maneira de alcançar este horizonte é através da manutenção preditiva. Esta abordagem pode ser definida como a manutenção baseada nas condições físicas dos equipamentos, tendo como objetivo agendar a reparação do veículo antes que ocorra qualquer avaria, e pode ser realizada através de várias técnicas [1]. Neste projeto propõem-se o uso do som, produzido pelo motor da viatura, como forma de informação para monitorização.

Através do uso de inteligência artificial, aplicando técnicas de aprendizagem automática e processamento digital de sinais de áudio, é possível realizar uma análise constante do som gerado pelo motor e consequentemente, conhecer o estado que este se encontra.

Para avaliar a condição do motor são criados dois módulos: um de classificação e um de separação de eventos.

A classificação é composta por redes neuronais treinadas com inúmeras amostras de padrões de som de motores, que se encontraram organizadas num conjunto de dados: *dataset*. Este módulo tem como objetivo identificar a existência de anomalias.

A separação de eventos é constituída por uma máquina de estados que tem o propósito de indicar a existência de eventos, entre os segmentos de áudio recolhidos, e separar o áudio recolhido em eventos. Esta separação tem como intuito facilitar a identificação de anomalias por parte do módulo de classificação.

Os módulos referidos apresentam resultados com base na análise de som. Esta análise é realizada a partir de *features* sonoras, isto é, de propriedades individuas mensuráveis do som [2].

A criação de uma plataforma possibilita manter um histórico de várias viaturas, com as avaliações realizadas, bem como permite aos utilizadores ouvir as anomalias produzidas pelos motores das suas viaturas.

Capítulo 2
Trabalho Relacionado

O som é uma grandeza que transporta grande quantidade de informação sobre o mundo que nos rodeia. Por exemplo, uma pessoa com limitações visuais consegue orientar-se essencialmente com base nos estímulos sonoros existentes à sua volta. Desta forma, cada vez mais o som é um elemento importante a ter em conta quando se pretende construir sistemas automáticos de análise de situações onde existe som. Assim, existem inúmeros projetos que utilizam o som como informação básica para tomada de decisão.

Um dos trabalhos que serve de base para a realização deste projeto tem o objetivo de detetar anomalias em máquinas, através de sons anómalos, completamente, desconhecidos [3]. Para isso os autores criaram um *dataset* composto por sons de miniaturas de máquinas e máquinas reais. Estes sons são constituídos por sons dos equipamentos a funcionarem de forma correta e de sons destes com anomalias. Para as anomalias foi necessário danificar as máquinas deliberadamente. A análise deste trabalho permitiu perceber como se constrói um *dataset* e de que forma este permite treinar um modelo composto por redes neuronais para detetar anomalias com sons desconhecidos. Esta base é útil, pois (tal como no trabalho mencionado) este projeto não tem acesso às anomalias que se propõe detetar. Em vez disso, são utilizados sons de motores a funcionar de forma correta e incorreta provenientes de base de dados na internet como o *AudioSet* [4] disponibilizado pela Google.

Noutro projeto estudado é proposta análise de falhas de máquinas com base nos sons produzidos por estas. Para tal, é utilizado um classificador do tipo CNN que é treinado através de imagens obtidas da *feature Mel-Spectrogram* [5]. A análise desta abordagem permitiu estruturar o trabalho proposto e aprofundar o conhecimento em relação às arquiteturas de *Machine Learning*.

Este projeto propõe também realizar outros objetivos que não estão presentes nos trabalhos mencionados, tal como a separação de eventos, pois um motor de um veículo não produz um som constante como as máquinas de fábricas e por também se pretender detetar quando o motor se encontra a funcionar.

Capítulo 3
Modelo Proposto

Este trabalho tem como propósito identificar as anomalias produzidas pelo motor de veículos. Para alcançar este propósito é necessário obter conhecimentos fundamentais, não só com o intuito de se realizarem opções inicias como para se definir a abordagem pretendida.

3.1 Fundamentos

De modo a implementar este projeto é imprescindível possuir uma base teórica tanto de características do som como de modelos de redes neuronais.

3.1.1 Características Sonoras

O trabalho foca-se principalmente na separação de eventos e na avaliação do som do motor, e para tal é necessária a existência de um conhecimento teórico de algumas características sonoras. Neste projeto analisam-se características como: *Zero Crossing Rate*, *Spectral Centroid*, *Spectral Rolloff*, *Spectral Flux* e *Root Mean Square*, para separar o som recolhido e, *Spectrogram*, *Mel-Spectrogram*, *Chromagram* e *Mel Frequency Cepstral Coefficients*, para a avaliação de eventos. Faz-se de seguida uma descrição dos várias *features* utilizadas no projeto.

Zero Crossing Rate

A *Zero Crossing Rate* tem como objetivo descrever o número de vezes que o sinal passa por zero numa certa quantidade de tempo. Esta apresenta valores elevados no caso das frequências elevadas, pois estas fazem com que o sinal passe muitas vezes por zero em pouco tempo, e quanto menor é a frequência menor é o valor apresentado pela *Zero Crossing Rate*. Esta é muito afetada por frequências de baixo valor, porque, como se pode observar na Figura 3.1, quando se soma uma frequência baixa por uma frequência alta, o sinal com maior frequência passa a delinear o sinal com menor frequência.

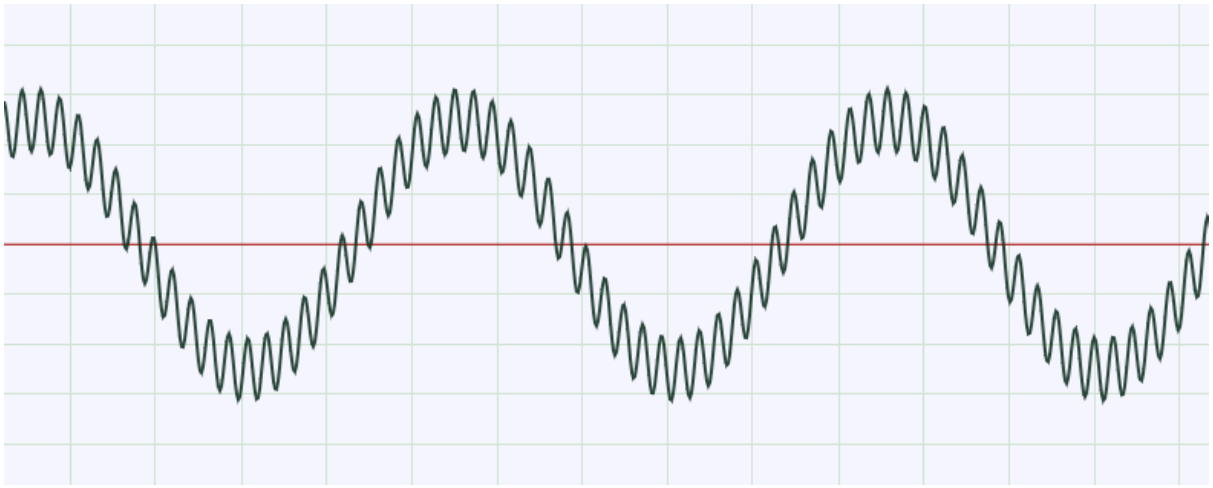


Figura 3.1 Multiplicação de sinais com diferentes frequências

Na Figura 3.2 apresentam-se 3 tons realizados com sinusoides: na primeira parte encontra-se uma sinusoide com -30 dB de amplitude e com frequência de 10 kHz, na segunda parte uma sinusoide com -18 dB de amplitude e com frequência de 440 Hz, por fim, a terceira parte é composta pela sobreposição de ambas as sinusoides.

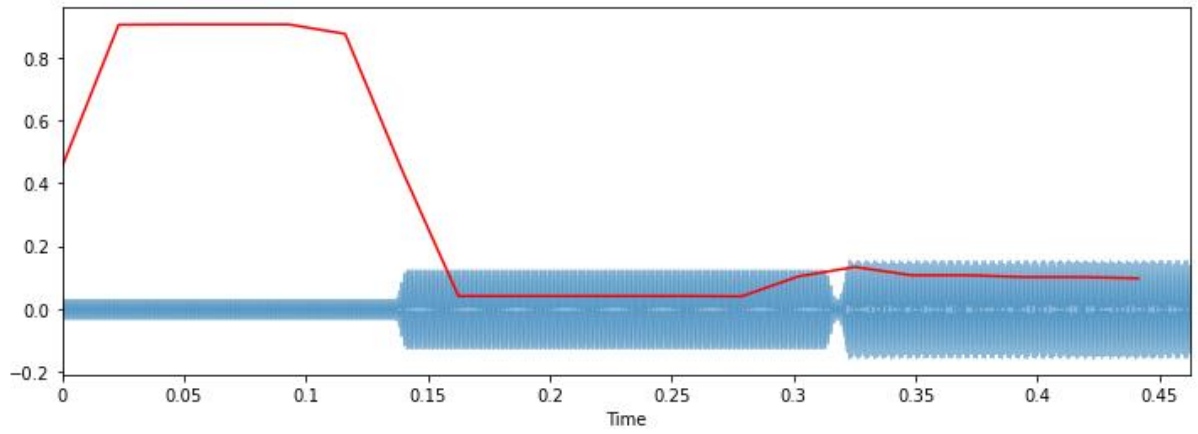


Figura 3.2 *Zero Crossing Rate* aplicada a diferentes sinusoides

Como se pode observar as frequências elevadas provocam um aumento do valor da *Zero Crossing Rate*, enquanto as baixas provocam uma diminuição do número de vezes que o sinal passa por zero, por fim unem-se as duas sinusoides para comprovar que a frequência baixa, quando somada com uma frequência alta, tem mais peso no valor da *Zero Crossing Rate*.

Root Mean Square

A característica *Root Mean Square*, tal como o nome indica calcula a raiz quadrada dos valores das amostras ao quadrado [6]. É uma medida da energia ao longo do tempo, portanto, relaciona-se com a intensidade sonora.

$$RMS = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)} \quad (1)$$

Nesta característica os valores utilizados referem-se aos valores da amplitude do sinal, logo quando maior for a energia do sinal maior é o valor da *Root Mean Square*. A Figura 3.3 apresenta a *Root Mean Square* normalizada.

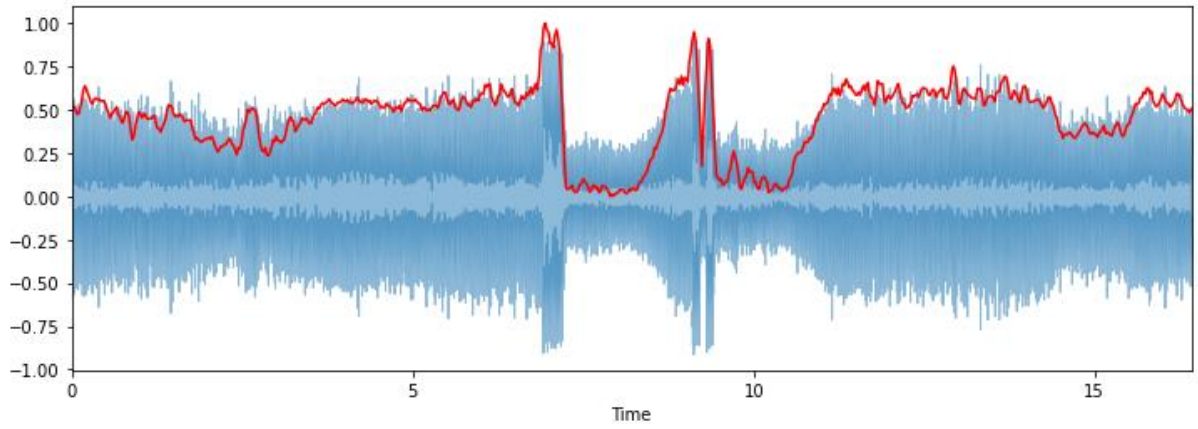


Figura 3.3 RMS aplicada a um som de motor

Observa-se, que a RMS descreve de forma muito fiel a variação de valores da amplitude do sinal em questão. Como tal, é muito dependente do ruído exterior, o que pode invalidar os resultados.

Spectral Rolloff

A *Spectral Rolloff*, tem como objetivo indicar até que frequência se encontra uma certa percentagem de energia do sinal. Esta percentagem começa a ser contabilizada a partir dos 0 Hz, e para obter a frequência pretendida, separam-se as frequências em vários *bins* e com base no peso de cada um (valor de energia) obtém-se a frequência central do *bin*, que juntamente com os *bins* abaixo dele possuem a percentagem de energia procurada. Na Figura 3.4 apresenta-se um espectrograma com o objetivo de auxiliar a visualização da *feature* em causa [7].

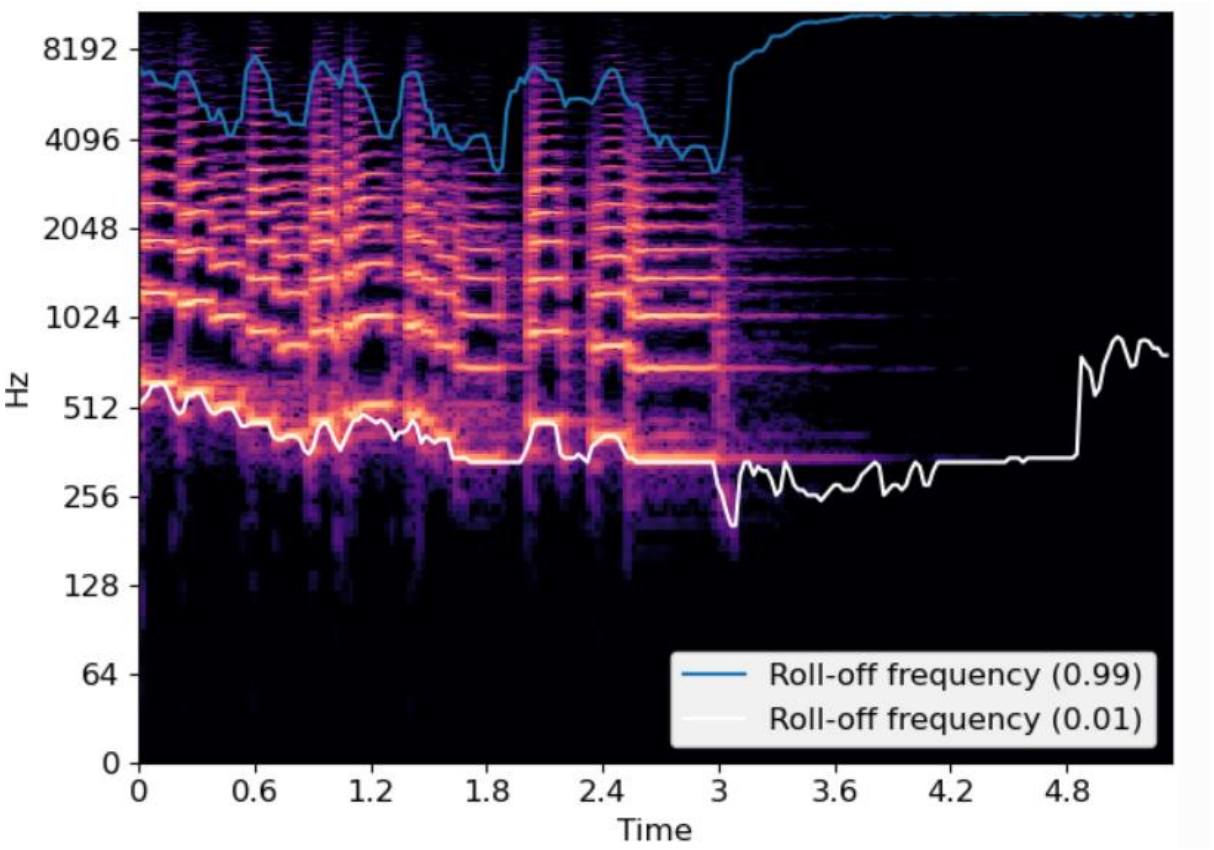


Figura 3.4 Curvas de *Rolloff* nas diferentes percentagens de *roll*

Observa-se, que quando menor é a percentagem de energia pretendida (*roll percent*) menor é o valor da *Spectral Rolloff*.

Spectral Centroid

A característica *Spectral Centroid* indica onde se localiza o centro de massa do espetro, e por isso apresenta uma conexão robusta com o brilho do espetro do som [8]. Para o cálculo desta característica também são usados *bins* de frequências, tal como demonstra a próxima fórmula:

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)} \quad (2)$$

onde x(n) representa o peso do valor da frequência do *bin* com número n, e f(n) representa o centro da frequência do *bin* [9].

Como a *Spectral Centroid* representa o centro de massa, esta não só tem em conta o peso dos *bins* como também a forma como estes estão distribuídos. Ao contrário da *Spectral Rolloff* que apenas tem em conta o peso dos *bins*. Isto significa que calcular a *Spectral Rolloff* com uma percentagem de 50% (*roll percent*=50%) não é o mesmo que calcular a *Spectral Centroid*. De forma a demonstrar a diferença de comportamento de ambas as características apresentam-se as Figuras 3.5 e 3.6.

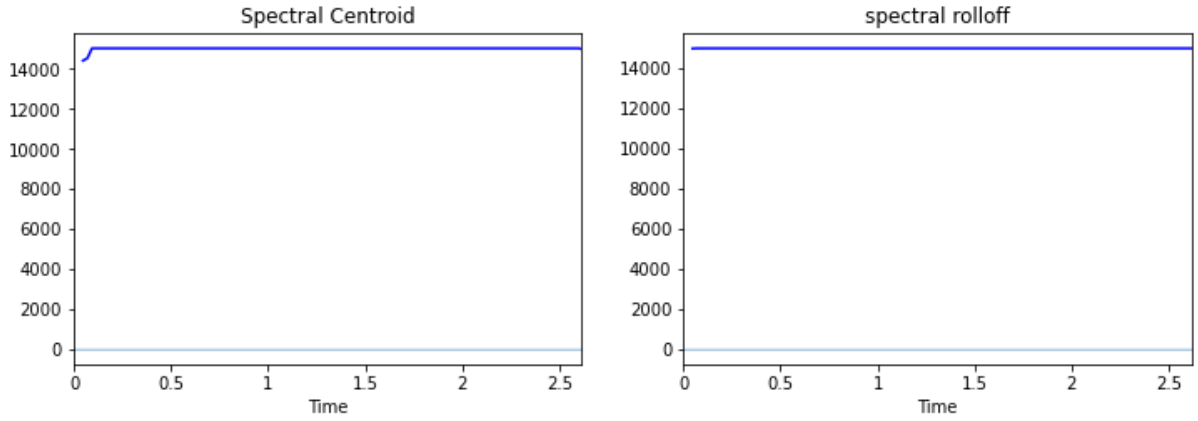


Figura 3.5 *Spectral Centroid* e *Spectral Rolloff* com *roll* 50% com os mesmos valores

Na Figura 3.5 é representado o resultado da aplicação da *Spectral Centroid* e da *Spectral Rolloff* com *roll percent* de 50% para um sinal composto por três sinusoides, com valores de frequência de 10 kHz, 15 kHz e 20 kHz, todas com a mesma amplitude.

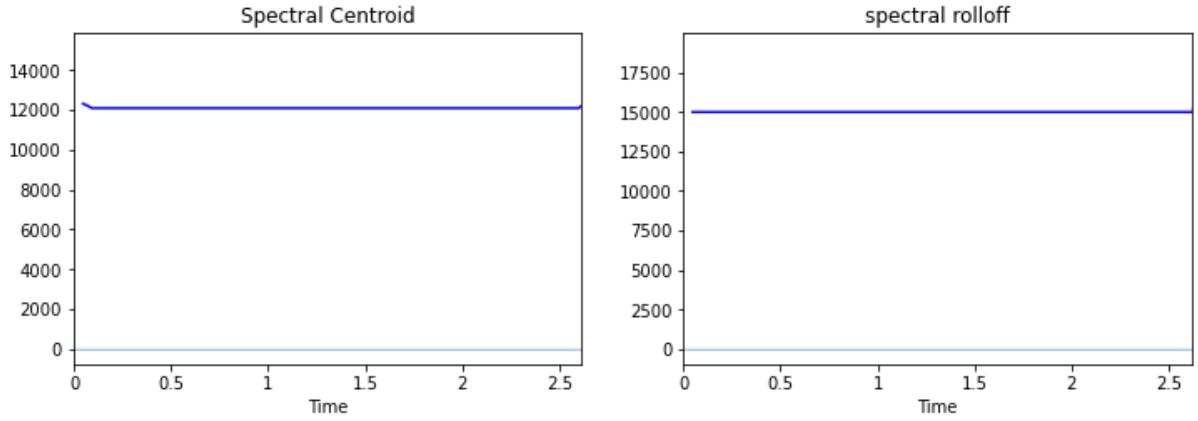


Figura 3.6 *Spectral Centroid* e *Spectral Rolloff* com *roll* 50% com valores diferentes

Já na Figura 3.6 é retratado o resultado da aplicação da *Spectral Centroid* e da *Spectral Rolloff* com *roll percent* de 50% para um sinal composto por três sinusoides, com valores de frequência de 1 kHz, 15 kHz e 20 kHz, todas com a mesma amplitude.

Constata-se, que enquanto na Figura 3.5 a *Spectral Rolloff* apresenta o mesmo valor que a *Spectral Centroid*, na Figura 3.6 isto não se verifica, pois em oposição à *Spectral Rolloff*, a *Spectral Centroid* varia. Esta variação pode ser explicada pela alteração do valor de frequência da primeira sinusoide de 10 kHz para 1 kHz. Ao se alterar o valor da frequência, altera-se também a distribuição de peso no espetro e como consequência o centro de massa deste deixa de se localizar nos 15 kHz. A *Spectral Rolloff* não altera o seu valor, pois 50% da energia do espetro continua a localizar-se entre 0 Hz e 15 kHz.

Spectral Flux

A *Spectral Flux* tem como propósito medir o quão rápido o espetro de energia de um sinal se altera [10]. Este cálculo é realizado ao se comparar o espetro de energia de uma *frame* com o espetro da *frame* anterior. A Figura 3.7 representa um sinal composto por três partes, cada qual com uma sinusoide com diferentes valores. A primeira possui uma frequência de 5 kHz e uma amplitude de -30 dB, a segunda uma frequência de 5 kHz e uma amplitude de -10 dB, e a terceira uma frequência de 10 kHz e uma amplitude de -10 dB.

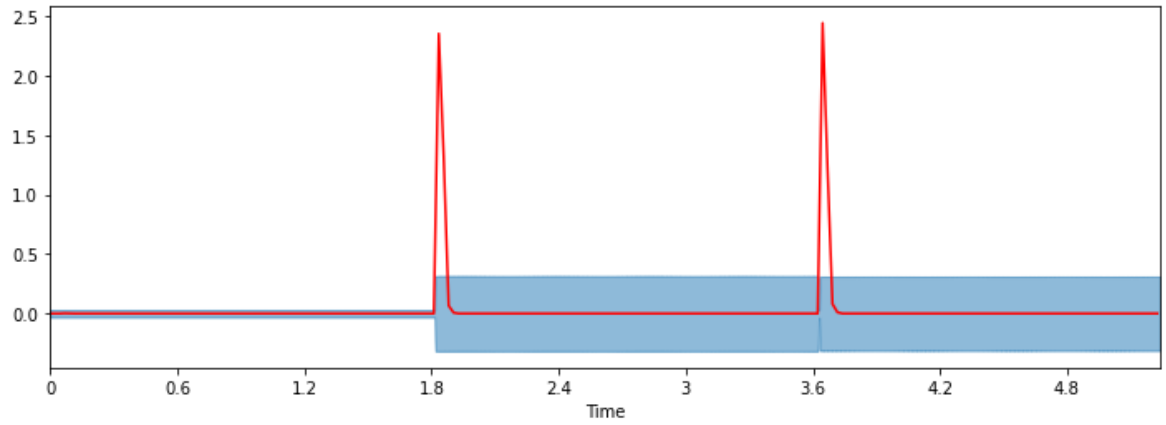


Figura 3.7 *Spectral Flux* aplicada a diferentes amplitudes e frequências

Verifica-se, que a variação da *Spectral Flux* é encontrada tanto na transição das sinusoides com amplitudes diferentes como nas frequências diferentes. Isto permite, que esta característica realize *onset detection*, ou seja, possibilita a identificação do início de um novo som [11].

Spectrogram

A *Spectrogram* é uma *feature* que representa a forma como a amplitude do sinal varia ao longo do tempo nas diferentes frequências. Esta é calculada através da sobreposição várias FFT de diferentes segmentos do sinal [12].

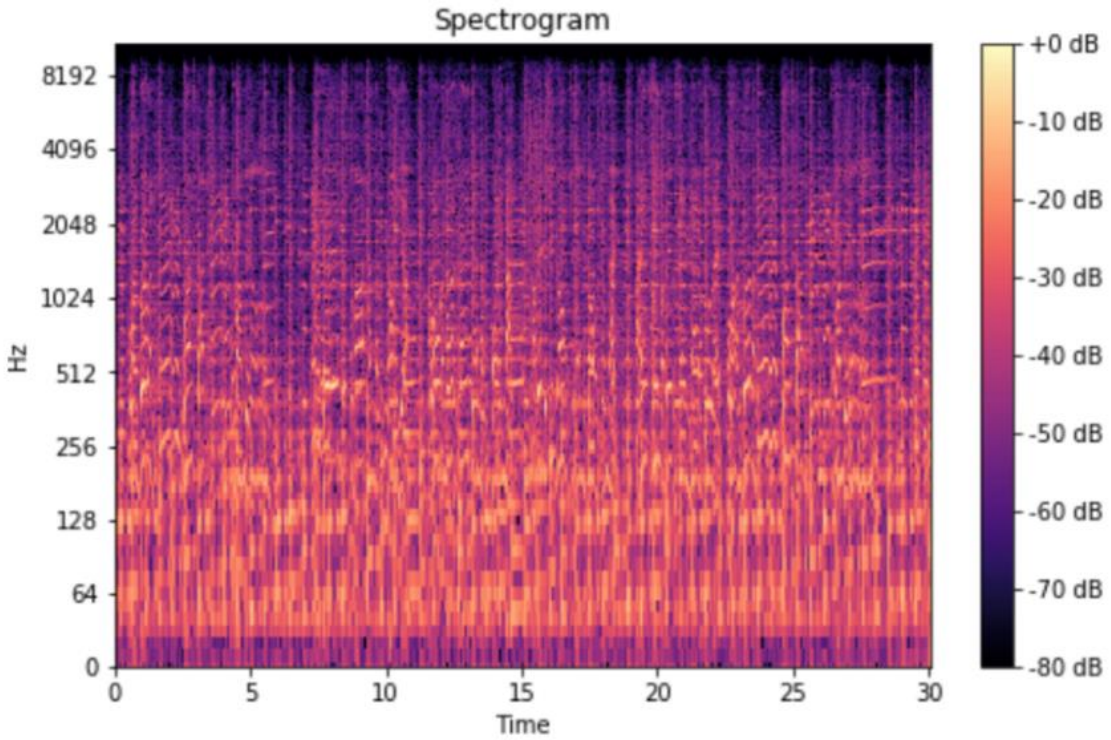


Figura 3.8 Visualização da *Spectrogram*

Na Figura 3.8 observa-se, que na *Spectrogram* a frequência encontra-se nas ordenadas e o tempo nas abcissas. Através da escala apresentada à direita da *Spectrogram* é possível compreender que quando mais clara é a cor de uma área maior é a quantidade de energia presente nesta. É importante esclarecer que a escala de cores é representada em decibel, porque os seres humanos compreendem apenas uma pequena faixa concentrada de amplitudes.

Mel-Spectrogram

A *Mel-Spectrogram* tal como o nome indica provém da *Spectrogram*. Esta surge devido ao facto dos seres humanos não compreenderem as frequências de forma linear. Ou seja, quanto maior é a frequência maior tem de ser a variação de frequência para ser notada pelo ouvido humano. Para solucionar este problema realizaram-se testes com inúmeras pessoas que deram origem à escala *Mel*.

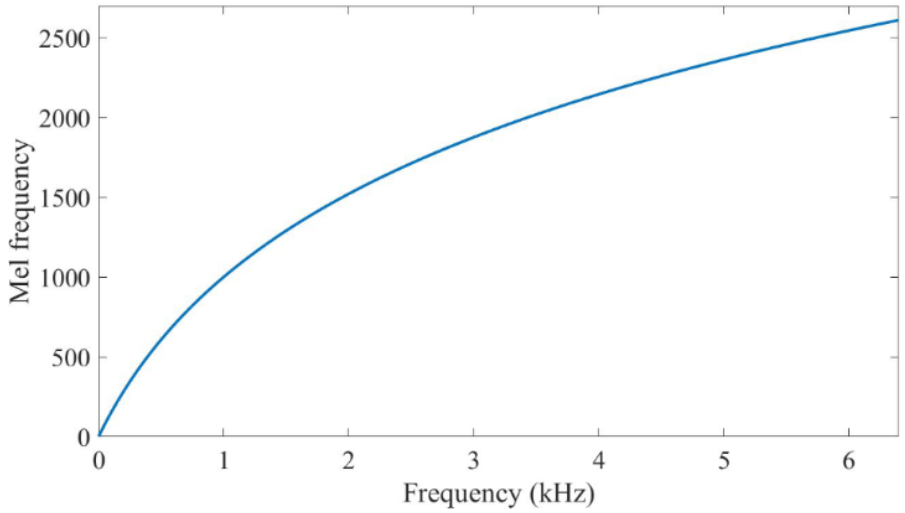


Figura 3.9 Escala *Mel*

Esta escala, como demonstrado na Figura 3.9 [13], tem forma logarítmica e adapta as distâncias das frequências à distância percebida pelos humanos.

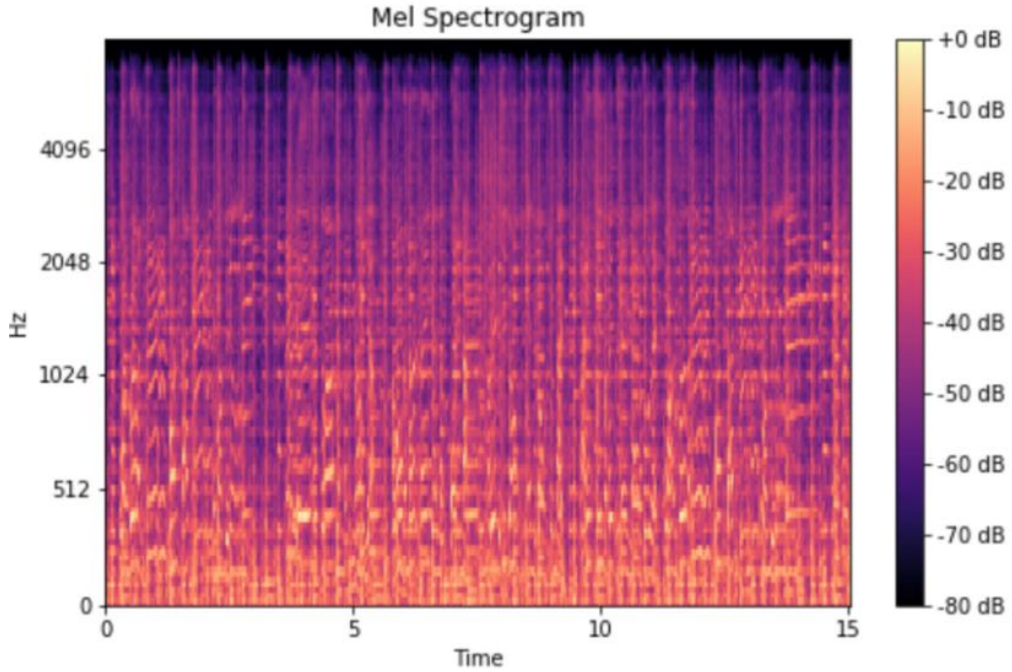


Figura 3.10 Visualização da *Mel-Spectrogram*

Como apresenta a Figura 3.10, a *Mel-Spectrogram* tem o mesmo formato que a *Spectrogram*. A única diferença é que as frequências se encontram na escala *Mel* [12].

Mel Frequency Cepstral Coefficients

Os *Mel Frequency Cepstral Coefficients* são os coeficientes que coletivamente formam o *Mel Frenquency Cepstrum*. A palavra *Cepstrum* deriva da inversão das primeiras quatro letras de *Spectrum*. Este "jogo de letras" surge porque para se obter um *Cepstrum* é necessário calcular um *spectrum* de um *spectrum*. A diferença entre o *Cepstrum* e o MFC é que na última as bandas de frequência encontram-se espaçadas igualmente na escala *Mel*.

Para calcular os MFCCs começa-se por realizar a Transformada de Fourier do sinal. Após obtido o *spectrum*, mapeiam-se os valores de energia deste na escala *Mel*. Realiza-se os logaritmos das energias de cada *Mel frequency*, e depois de obtida a lista destes efetua-se a Transformada Discreta de Cosseno (*Discrete Cosine Transform*) da mesma. Os MFCCs são as amplitudes do *spectrum* resultante [14].

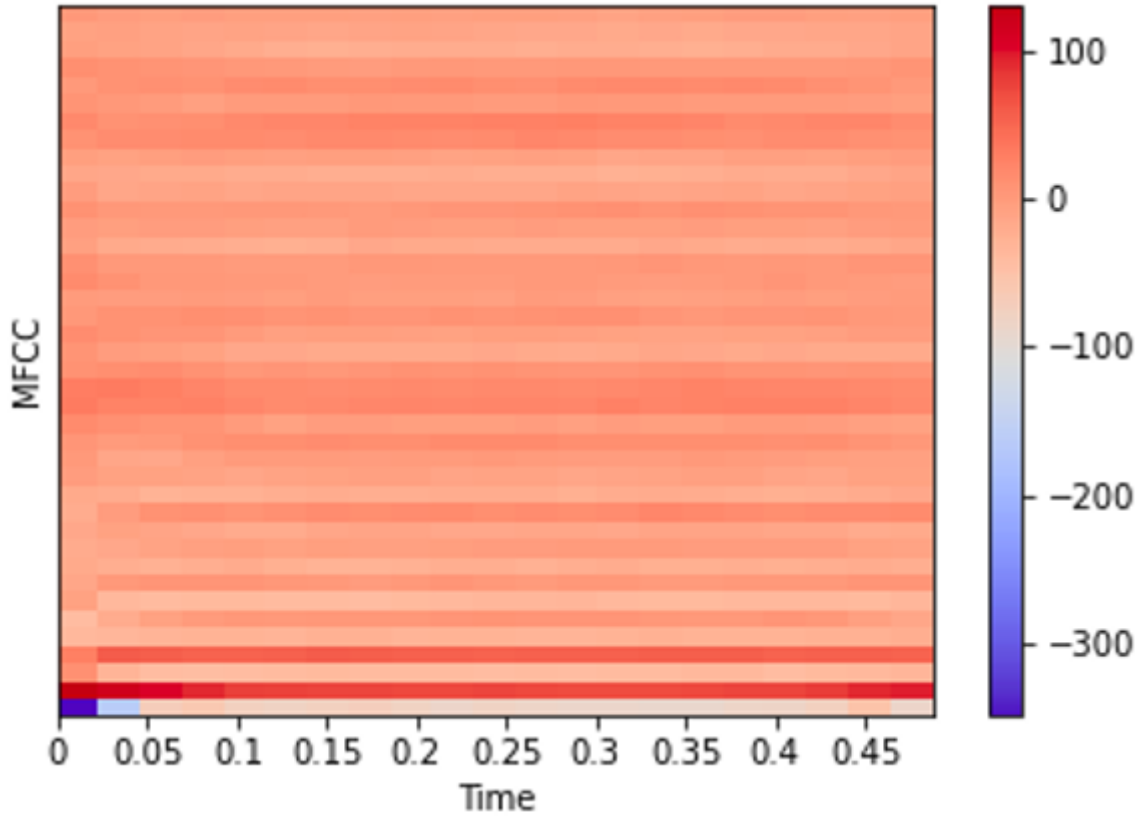


Figura 3.11 Visualização dos MFCCs

Como demonstra a Figura 3.11, os MFCCs são apresentados nas ordenadas na imagem resultante.

Chromagram

A *Chromagram* tem como objetivo representar a intensidade do sinal nas doze classes de tom (conjunto de todos os tons separados por oitavas [15]). Para isto, esta separa as frequências do sinal pelas doze notas musicais existentes [16].

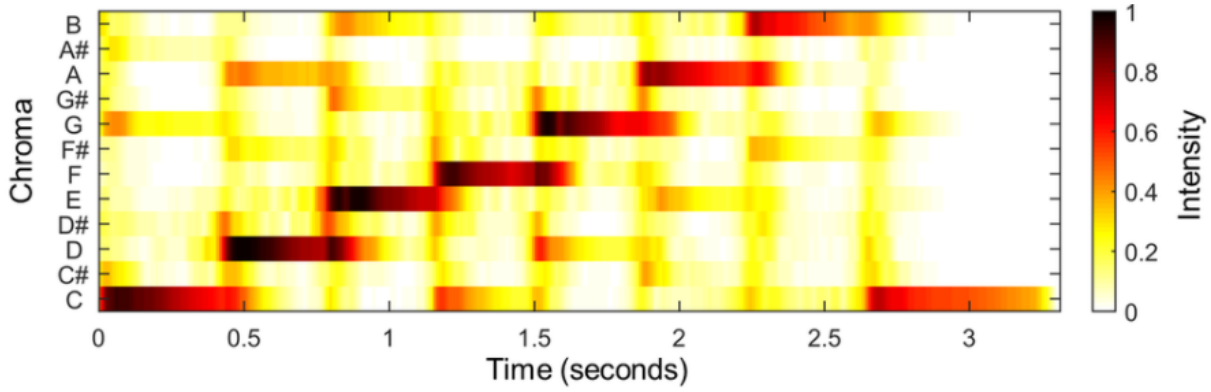


Figura 3.12 Visualização da *Chromagram*

Pela Figura 3.12 constata-se, que a *Chromagram* tem semelhanças com a *Spectrogram*, visto que as suas abcissas representam o tempo e esta também possui uma escala de cores. Mas em vez de ter a frequência nas ordenadas esta tem as notas musicais.

A Figura 3.12 é o resultado das notas musicais representadas na escala da Figura 3.13.

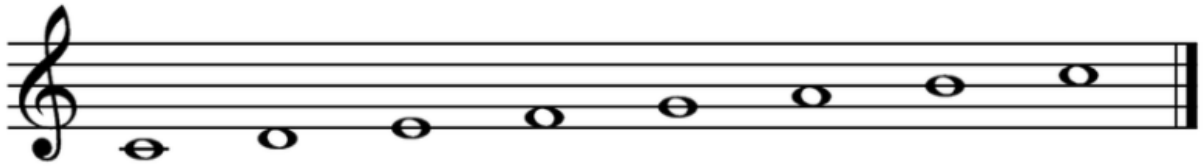


Figura 3.13 Notas na escala musical

Como se pode observar, apesar da primeira e última nota da Figura 3.13 serem representadas pela nota C na Figura 3.12, estas não possuem a mesma frequência. Devido ao facto das notas musicais se repetirem, cada nota retrata várias frequências. O dobro de uma frequência representada pela nota C também é caracterizada pela nota C, ou seja, uma oitava acima. Salienta-se este aspeto, porque as intensidades das notas presentes na *Chromagram* podem ser influenciadas por mais de n frequências.

3.1.2 Redes Neuronais

Para a construção do classificador é necessário um modelo com base em redes neuronais. As redes neuronais artificiais são modelos computacionais inspirados pelo sistema nervoso central (mais especificamente pelo cérebro), que são capazes de realizar aprendizagem automática bem como o reconhecimento de padrões. Estas possuem nós ou unidades de processamento. Sendo que, cada unidade possui ligações para outras unidades, nas quais recebem e enviam sinais. Uma rede neuronal pode conter múltiplas camadas de nós de processamento [17].

O modelo *Feedforward Neural Network* é tipo um de rede neuronal, onde a informação move-se numa única direção "*forward*" (para a frente) desde os *input nodes* (nós de entrada) até aos *output nodes*, passando pelos *hidden nodes* (se estes existirem). Neste modelo não existem ciclos nem *loops*, tal como demonstrado na Figura 3.14 [18].

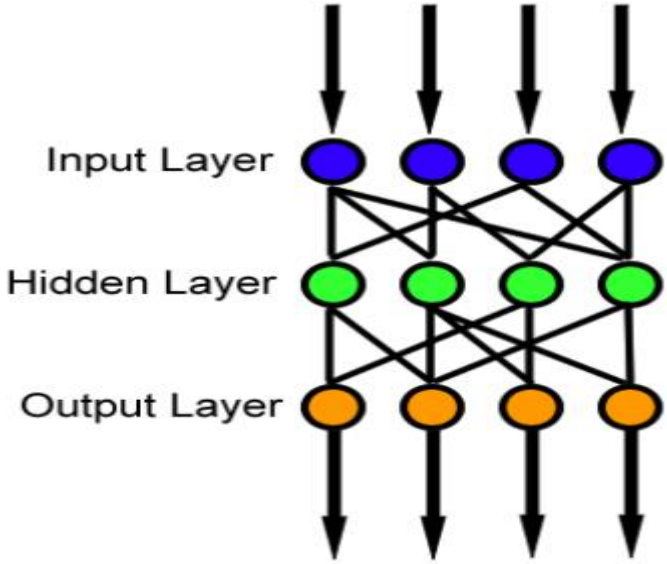


Figura 3.14 Representação do modelo *Feedforward Neural Network*

O tipo de rede neuronal referido possui várias arquiteturas. Como o principal objetivo do trabalho passa pelo estudo dos sinais sonoros e das suas características, optou-se pela escolha da arquitetura mais simples, sendo esta a *Dense Only Network*.

A arquitetura *Dense Only Network* implica que as *hidden layers* (camadas) sejam do tipo *Dense*. Isto significa que cada neurónio nessas camadas recebe input de todos os neurónios da camada anterior. Nesta camada é realizada a multiplicação do input por um vetor, sendo que o vetor é atualizado ao longo do treino do modelo [19].

3.2 Abordagem

Este projeto tem como objetivo a realização da avaliação do som produzido por um motor de um veículo e a apresentação dos resultados obtidos ao condutor do mesmo. Para tal, é proposto o esquema da Figura 3.15.

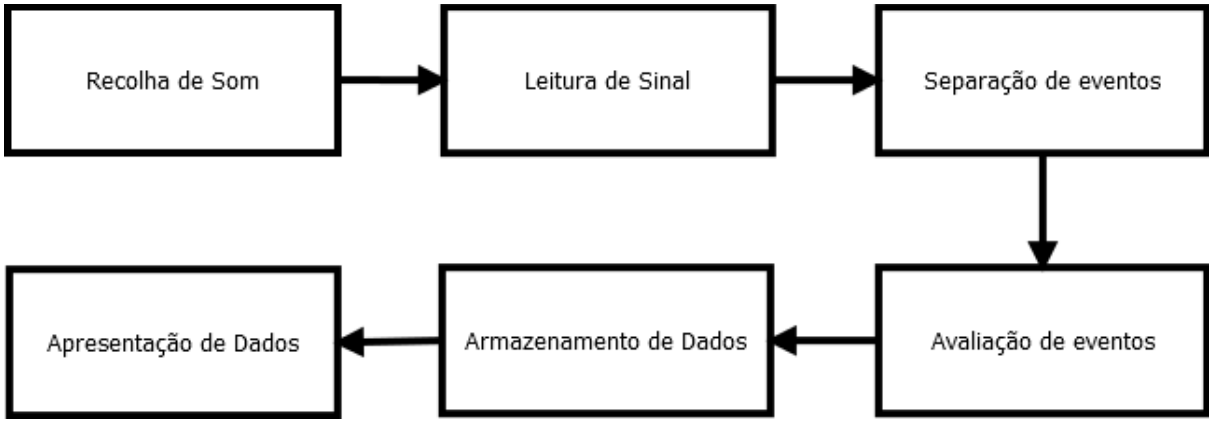


Figura 3.15 Objetivo final do projeto simplificado

Como está demonstrado na Figura 3.15, pretende-se começar por obter o som do motor da viatura (através de um microfone), ler (em contínuo ao longo do tempo) o sinal recebido, separar este em eventos, classificar os eventos separados e por fim, guardar os dados dos eventos e mostrá-los (numa aplicação Web).

Como se pretende a atualização constante dos dados, e tendo em conta que o som recolhido poderá ter uma duração de várias horas, é necessário realizar a análise com a separação do sinal obtido em segmentos de um determinado número de amostras. Estes segmentos são à posteriori passados ao módulo de separação de eventos, que serve não só para facilitar a avaliação dos eventos, por parte do classificador, pois sem este seriam enviados sempre segmentos de cinco segundos (independentemente do seu conteúdo), como também permite a diminuição de consumos energéticos, porque tenciona-se que este considere que apenas existem eventos quando o motor está ligado. Esta funcionalidade é necessária porque neste projeto não se pretende conectar o dispositivo ao painel de controlo do carro. Quando o separador de eventos considera que existe um evento este passa-o ao classificador.

No classificador visa-se perceber se o som obtido é normal ou uma anomalia, e com base nesta avaliação enviar o som recebido para a plataforma.

Na plataforma o utilizador poderá observar o comportamento do motor do seu veículo ao longo do tempo, bem como também poderá ouvir as anomalias produzidas pelo mesmo.

Para o classificador pretende-se ter um *dataset* com dados de motores a funcionar de forma correta e de forma incorreta. Como o separador de eventos poderá considerar que existe um evento mesmo com o motor do carro desligado, também se propõe colocar nas anomalias sons citadinos e de motores que não pertencem a veículos. Devido à proximidade e do elevado som produzido pela buzina, tenciona-se ter sons de buzinas a representar o funcionamento correto do veículo.

Capítulo 4
Implementação do Modelo

Na execução deste projeto escolheu-se a linguagem de programação *python* para a realização do código, que permite a criação e treino do modelo de redes neuronais. Esta também foi utlizada na implementação do módulo de processamento de sinal. A escolha deve-se ao facto da linguagem *python* ser simples e consistente, mas principalmente pelo seu número elevado de bibliotecas, que permitem realizar tarefas complexas de uma forma simples e rápida [20]. Dá-se uso a bibliotecas como o *librosa* para realizar a análise de áudio, o *keras* e o *tensorflow* que possibilitam a utilização de técnicas de *machine learning* (sem a necessidade de ter um conhecimento aprofundado sobre este tema).

4.1 Processamento

No código *python* implementado para além do *main*, existem mais duas *threads* a funcionar simultaneamente (Apêndice A). Uma resulta do *callback* da biblioteca *PyAudio*, que permite receber o áudio obtido no microfone do computador, e outra *thread* responsável pela separação, classificação de eventos e pelo envio dos sons das anomalias para a plataforma. Estas tarefas têm de estar separadas, porque se estivessem na mesma, enquanto o código realizaria operações sobre o sinal recolhido não estaria a obter o sinal proveniente do microfone. Para tal, é utilizado o padrão *Producer/Consumer*, no qual os processos são designados como produtores ou consumidores. Os produtores são responsáveis por adicionar elementos a uma estrutura de dados, enquanto os consumidores são responsáveis por removê-los dessa mesma estrutura [21].

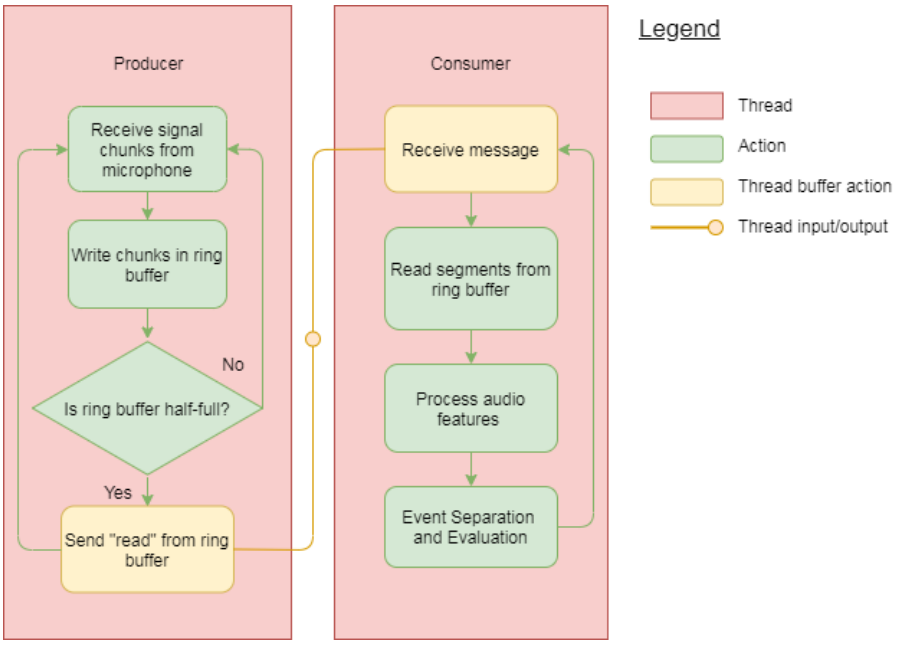


Figura 4.1 Funcionamento das *threads* da plataforma implementada

Como se observa na Figura 4.1 enquanto a *thread Consumer* realiza a operações mais demoradas (processamento da *features* de áudio, a separação de eventos e avaliação dos mesmos), a *thread Producer* apenas lê pedaços do sinal proveniente do microfone e passa estes para o *buffer*. Isto permite à aplicação obter o sinal ao mesmo tempo que realiza operações sobre este.

Para poder ter um código a funcionar em tempo real criou-se um *buffer*. Este não só permite passar os dados recolhidos de uma *thread* para a outra, como também possibilita escolher o tamanho dos segmentos que se pretende analisar. Para tal, utiliza-se um *buffer* circular, com dois ponteiros que permitem saber o quão cheio o *buffer* se encontra, e consequentemente alertar à *thread* consumidora que deve ler o seu conteúdo. Também é muito útil, pois consegue facilmente obter os valores de uma variável *array* e quando os seus valores são lidos para outro *array* esses mesmos valores são eliminados do *buffer*. Além disso, o *buffer* lança uma exceção em caso de *overflow*, se os novos valores sobrescrevem os valores anteriores [22].

Na parte de separação de eventos utilizam-se as *features* RMS, *Spectral Rolloff* e *Spectral Flux*. A *feature Zero Crossing Rate* não é utilizada, porque depois de alguns testes esta demonstra ter a mesma utilidade que a RMS, em termos de verificar se o motor se encontra a funcionar. Devido à maior complexidade da variação e da escolha de um *threshold* para esta característica acabou-se por optar pela escolha da RMS, que também tem a vantagem de apresentar uma representação do sinal mais intuitiva para o "olho humano". Também não se utiliza a característica *Spectral Centroid* por esta avaliar os mesmos aspetos que a *Spectral Rolloff*, apesar de possuir algumas diferenças. A escolha da *Spectral Rolloff* deveu-se apenas à observação do comportamento de ambas quando aplicadas a vários sons. Não foi observada uma diferença muito grande entre elas nos testes, como seria expectável. Normalmente a *Spectral Rolloff* aparenta ser mais previsível, depois de serem ouvidos os áudios nos quais as duas características são testadas.

É importante referir, que na maior parte das vezes, as *features* apresentadas encontram-se normalizadas, isto é, as imagens mostradas não apresentam os seus valores reais, e por isso características como a RMS, parecem descrever melhor o sinal do que na realidade o fazem. Apesar da normalização ser muito útil para observar o comportamento das *features* ao longo do sinal, esta não pode ser utilizada no separador de eventos, por este realizar o cálculo das *features* em segmentos com apenas algumas *frames*. A seguir apresentam-se as Figuras 4.2 e 4.3, nas quais as *features* estudadas são aplicadas a um som produzido por um motor, com o objetivo de demonstrar o problema do uso da normalização em *real time*.

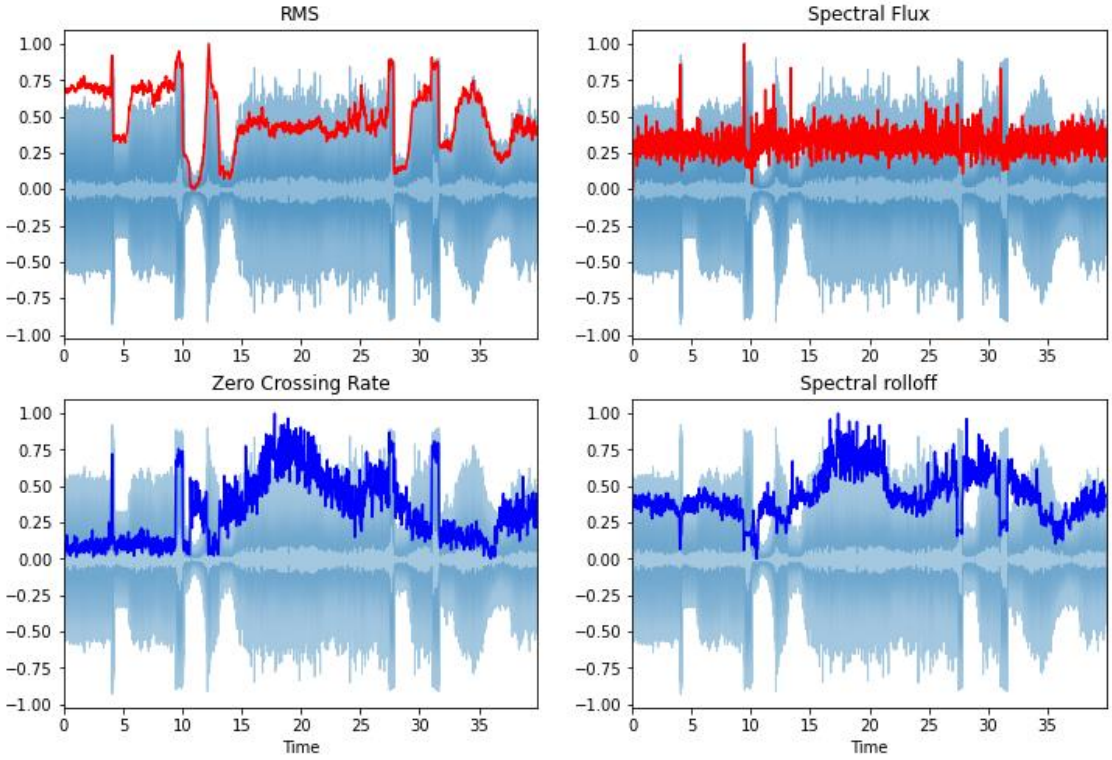


Figura 4.2 Normalização de *features* no fim do processamento de sinal

Na Figura 4.2 o cálculo e a normalização das *features* é realizado no fim do sinal.

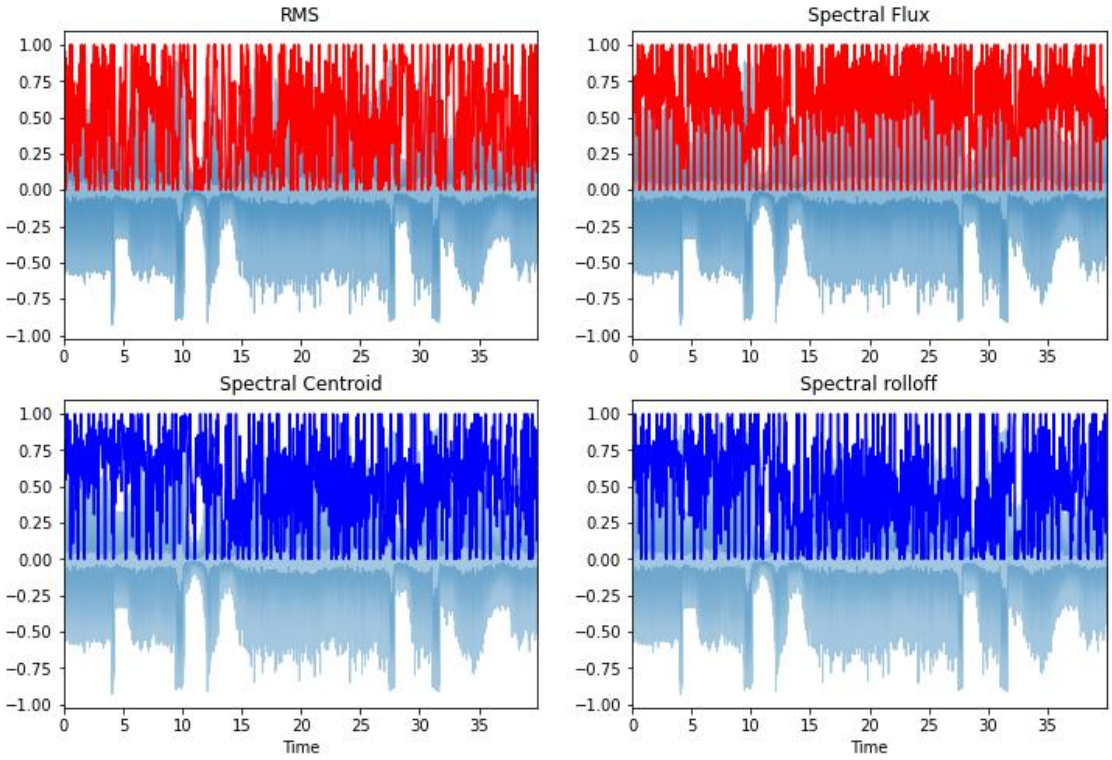


Figura 4.3 Normalização de *features* durante o processamento de sinal

Já na Figura 4.3 o cálculo das *features*, bem como a sua normalização é feito em segmentos do sinal.

Percebe-se que as *features* deixam de ser percetíveis quando se realiza normalização de segmentos de sinal, isto deve-se ao facto da normalização adaptar as *features* para à escala pretendida. A normalização que está a ser utlizada tem como nome *Rescaling* ou *min-max normalization* [23] e pode ser calculada através da seguinte expressão:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3}$$

Sendo x' o valor normalizado, observa-se que na expressão devido ao facto do mínimo e do máximo de cada segmento ser considerado, todos os segmentos terão, pelo menos, um valor mínimo e máximo, mesmo que isso não represente o sinal em questão.

4.2 Separação de Eventos

Para utilizar as *features* escolhidas de uma forma eficiente e lógica construiu-se uma máquina de estados como apresenta a Figura 4.4.

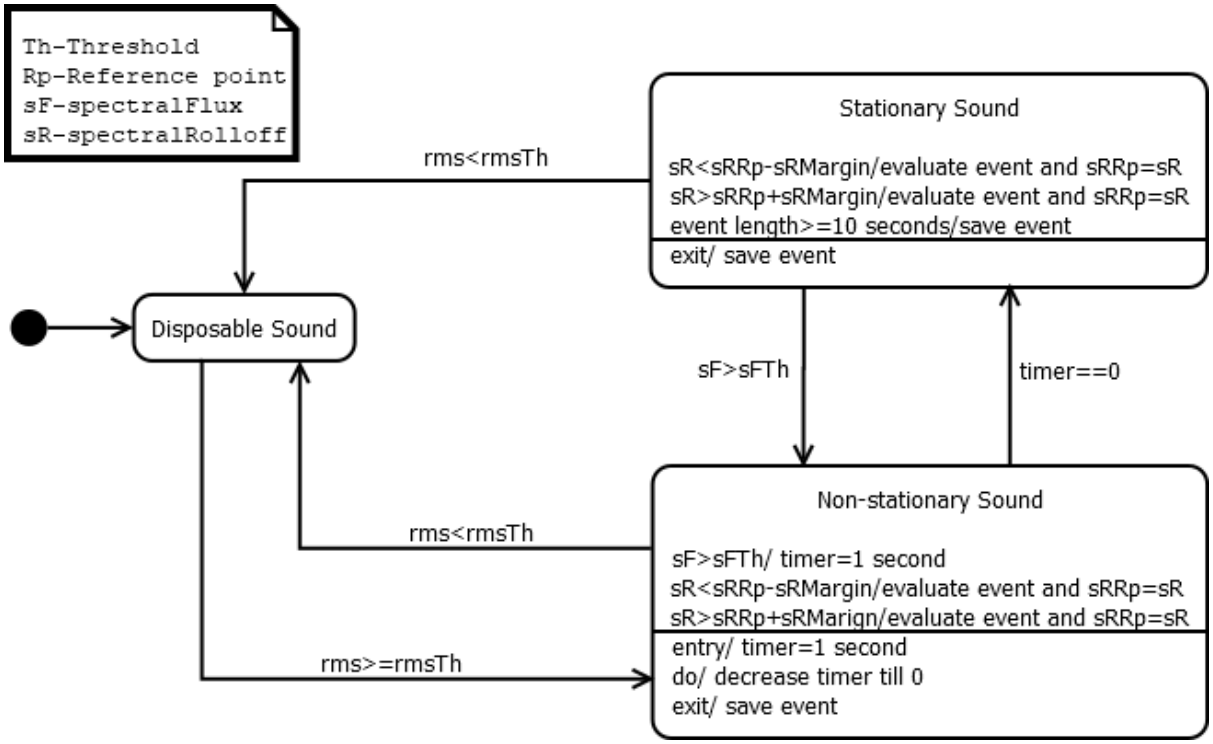


Figura 4.4 Diagrama UML da Máquina de Estados

Como já foi referido, a separação de eventos é realizada através de uma máquina de estados. A máquina em questão possui três estados: Estado 1 - *Disposable Sound* para o som que não é considerado evento, Estado 2 - *Non-Stationary Sound* que considera que o som que provém do microfone tem a possibilidade de ser de um evento não estacionário e Estado 3 - *Stationary Sound* que separa eventos considerados estacionários.

A máquina inicia-se pelo estado Estado 1 - *Disposable Sound*. Permanece neste estado

enquanto considera que o som tal como o nome indica é descartável. Neste estado apenas é comparado o *threshold* da RMS, rmsTh, com a RMS do sinal. Se o valor RMS for superior à rmsTh a máquina de estados passa para o estado Estado 2 - *Non-Stationary Sound*.

A máquina passará para Estado 2 - *Non-Stationary Sound* sempre que considerar que há uma possibilidade do som ser não estacionário, como por exemplo quando o carro inicia ou o condutor toca a buzina. Neste estado continua-se a verificar o valor da RMS, para se saber se o motor continua a funcionar. Este também possui um *timer*, que é decrementado ao longo do tempo. Inicialmente coloca-se uma margem de tempo no *timer*, que permite verificar se o evento é não estacionário, esta verificação é realizada ao longo deste estado pela *Spectral Flux*. Sempre que esta *feature* ultrapassa o seu *threshold*, sFTh, o *timer* é atualizado para o seu valor inicial, indicando assim que está a decorrer um evento não estacionário. Quando o *timer* descrito for decrementado e atingir zero considera-se que se está no evento estacionário e muda-se para o estado Estado 3 - *Stationary Sound*. A característica que é utilizada para se separar eventos neste estado é a *Spectral Rolloff*. Esta possui uma margem de frequência que indica se existe ou não uma grande variação de frequência, tanto para valores superiores como inferiores. Sempre que esta *feature* tem um valor fora da sua margem, o seu ponto de referência é atualizado. Caso a RMS seja menor que o seu *threshold* volta-se para o estado inicial.

Quando a máquina de estados se encontra no estado Estado 3 - *Stationary Sound* pode-se presumir que nos encontramos num evento estacionário. Assim, continua-se a verificar a RMS e a *Spectral Rolloff* como no estado anterior, a diferença deste estado para o anterior é que quando a *Spectral Flux* ultrapassa o seu *threshold* volta-se para o estado Estado 2 - *Non-Stationary*. Neste regime, é provável haverem eventos muitos longos, isto é, haver pouca variação das *features* durante muito tempo, por isso também se separa um evento sempre que este possua um valor temporal superior a 10 segundos.

Na máquina de estados quando há uma separação de evento ou se muda de estado (exceto na passagem do estado Estado 1 - *Disposable Sound* para o Estado 2 - *Non-Stationary*), usa-se a função *evaluate_event*. Esta função verifica se o evento passado possui uma duração superior a meio segundo, se sim, envia o evento para o classificador e inicia um novo evento, se não, não realiza nenhuma operação. Esta verificação existe para evitar enviar eventos demasiado curtos para o classificador, pois muitas vezes as *features* variam com diferenças de tempo muito pequenas. Também se deve ao facto de frequentemente esta verificação impedir o envio de eventos que não são do motor do carro. Por exemplo, nos testes constatou-se que o bater de uma porta do carro pode fazer com que o RMS ultrapasse o seu *threshold*. Como o "bater de porta" é um evento muito curto com esta averiguação este já não seria considerado evento.

4.3 Classificação

Na avaliação de eventos utilizou-se a *feature Mel-Spectrogram*, devido à suas vantagens logarítmicas em relação ao espectrograma e ao seu menor processamento quando comparado com os MFCCs. Inicialmente começou-se pelo uso dos MFCCs, mas devido à complexidade da computação do mesmo, não se estava a aproveitar todas as suas capacidades. Usou-se a média e o desvio de padrão da imagem apresentada pelos MFCCs. Quando se realizaram testes, com a média e o desvio de padrão, o *Mel-Spectrogram* foi a *feature* que mais vezes conseguiu distinguir, de forma correta, o som de motor normal de um com anomalias, como representado na Tabela 5.3. A *feature Chromagram* apesar de ter sido estudada, apresentou os piores resultados, como seria de esperar, devido ao facto desta *feature* ser mais indicada para a classificação de música, pois funciona com base na classificação de notas musicais.

Ao se utilizar a média e o desvio de padrão das imagens perde-se informação presente nestas. Realizou-se esta abordagem, pois pretende-se avaliar sons, normalmente, estacionários e devido à elevada complexidade dos modelos alternativos.

O modelo de redes neuronais usado é o *Dense Only Network*, este é do tipo *Sequential*, isto é, as camadas deste comunicam de forma linear [24]. A primeira camada do modelo é do tipo *flatten*, esta tem como objetivo tornar os *arrays* multidimensionais recebidos em *array* de apenas uma dimensão [25]. Os *arrays* passados ao modelo têm duas dimensões, sendo que a primeira representa as médias da *Mel-Spectrogram* em cada frequência e a segunda contém os desvios de padrão da *Mel-Spectrogram* em cada frequência. A arquitetura implementada possui três camadas do tipo *Dense*. As duas primeiras camadas têm *activation* do tipo *relu*, isto significa que todos os resultados obtidos nestas que sejam inferiores a zero são alterados para zero [26]. Também possuem *dropout* de 0.5, ou seja, metade dos valores obtidos por estas camadas são alterados para zero [27]. Isto permite prevenir o *overfitting* (sobreajuste) do modelo. O *overfitting* é um termo usado para indicar que um modelo se ajusta demasiado bem a um conjunto de dados. Ou seja, que o modelo apresenta bons resultados na classificação de dados conhecidos (dataset de treino), mas maus resultados na classificação de novos dados (dataset de teste) [28]. A última camada que compõe a arquitetura tem activation *softmax*, isto é, converte um vetor de valores numa distribuição de probabilidades [26], sendo estas apresentadas como resultado final da avaliação.

O *dataset* é separado em dois grupos, um de treino e outro para teste (80% treino e 20% teste). Nesta separação é utilizado o modo *stratify* que, obriga a que a separação do *dataset* seja uniforme (são retirados 80% para treino de cada classe e 20% para teste de cada classe).

O modelo é treinado em 100 *epochs* com *batches* de tamanho 32. Sendo que, *batch* é um conjunto de amostras que é processado antes do modelo ser atualizado. Este pode possuir diversos tamanhos, os mais utilizados são 32, 64 ou 128. O *epoch* é um ciclo por todo o *dataset* de treino. O número de *epochs* pode ser fixo, mas deve-se orientá-lo com base no erro no conjunto de validação. Um *epoch* é composto por um ou mais *batches* [29]. Depois de ser treinado e testado o modelo encontra-se pronto para ser utilizado.

Inicialmente pensou-se em utilizar sons de *datasets* online como o *AudioSet* [4]. Porém, devido à escassez e à falta de qualidade dos sons, decidiu-se por tentar procurar e contactar algumas entidades externas (Apêndice B), com o intuito de se gravar os motores dos carros disponíveis e se possível de carros com problemas nos motores.

Através dos orientadores do projeto foi possível realizar uma reunião com um representante da CarClass, que se mostrou disponível para colaboração no projeto disponibilizando um carro híbrido, permitindo assim obter os sons produzidos por este e guardá-los no *dataset*. Mas devido à falta de tempo acabou por não ser possível realizar o agendamento das gravações.

Assim, acabou-se por se tentar gravar os carros disponíveis aos elementos do grupo. Tal como demonstra a imagem de capa do trabalho, de princípio tentou-se gravar o som do motor com um microfone disponível ligado a um computador, mas devido à falta de qualidade do equipamento acabou-se por se gravar o som com um telemóvel. Este foi colocado numa zona segura próxima do motor, como apresenta a Figura 4.5.

Figura 4.5 Recolha de som de um carro movido a gasóleo

A imagem da Figura 4.5 é de um motor de um carro movido a gasóleo, de onde provém grande parte dos sons presentes no *dataset*. Mais tarde, com o objetivo de se diversificar o *dataset*, também se realizou uma gravação de um carro movido a gasolina, como demonstra a Figura 4.6.

Figura 4.6 Recolha de som de um carro movido a gasolina

O *dataset* atualmente possui mais de mil sons, compostos por gravações realizadas pelos elementos do grupo e por *datasets* online. Os *datasets* online foram essenciais para a obtenção de sons de anomalias, apesar de muitos destes não representarem falhas de motores, devido à sua escassez.

4.4 Plataforma

Por fim, este projeto tem como objetivo apresentar os dados recolhidos ao longo do tempo numa plataforma. Para tal foi utilizado o Firebase, que permite guardar ficheiros sem utilização de lógica SQL. Este possui um plano gratuito com o qual se pode armazenar 1 GiB, realizar diariamente 50000 leituras e 20000 gravações de documento [30].

Sempre que o classificador considera um evento uma anomalia, este envia o áudio do evento, com o nome da data e hora atual para um *folder* da *storage* do Firebase. Assim os dados podem ser acedidos por qualquer equipamento com acesso à internet. O *folder* no qual são guardados os ficheiros de áudio dependem do id de cada utilizador, assim é possível guardar na mesma *storage* os dados de múltiplos utilizadores sem que estes tenham acesso aos dados de outro utilizador.

Com o intuito de demonstrar os dados recolhidos, criou-se uma aplicação Web simples através de *javascript* e HTML na qual é possível criar uma conta para um novo utilizador, fazer login na mesma, observar o comportamento do motor ao longo do tempo e ouvir o som das anomalias que possam ser produzidas pelo mesmo (Apêndice A). A Figura 4.7 Mostra a página inicial desta aplicação Web, enquanto o utilizador não se encontra autenticado.

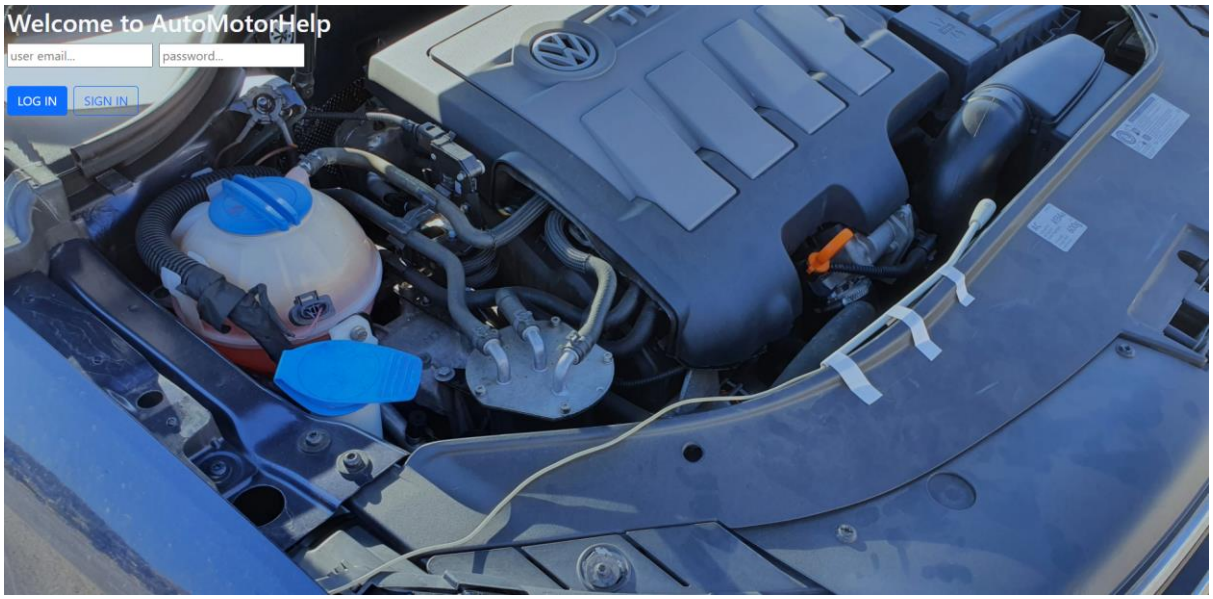


Figura 4.7 Página inicial da aplicação Web

Nos gráficos, realizados através de *Chartjs*, o utilizador pode não só observar o número de anomalias que ocorreram ao longo do tempo, como também visualizar a duração destas. Como a Firebase *storage* fornece o tamanho dos ficheiros, mas não permite obter a duração dos ficheiros de áudio, foi necessário o uso da seguinte expressão [31]:

$$fileSize = \frac{bitsPerSample * samplesPerSecond * channels * duration}{8} \quad (4)$$

para apresentar a duração dos ficheiros, sem se fazer download dos mesmos.

Na autenticação do Firebase escolheu-se a opção com email (o email não tem de ser real, mas tem de possuir formato de email) e para separar os dados de diferentes usuários utilizaram-se as regras da *storage* do Firebase, permitindo assim que cada utilizador apenas tenha acesso ao seu *folder*.

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    match /{userId}/{allPaths=**} {
      allow read, write: if request.auth.uid==userId;
    }
  }
}
```

Figura 4.8 Regras implementadas na *storage* do Firebase

Na Figura 4.8 é possível observar, que o utilizador tem de se encontrar autenticado, pois se não estiver não é possível obter o seu *uid*. Também se constata que o *path* ao qual o utilizador pode aceder tem de ter o mesmo valor que o seu *uid*, impossibilitando assim o usuário de realizar *read* ou *write* noutras pastas, que não a sua.

Para gerir a autenticação foram usadas as bibliotecas do Firebase. No caso do *python* foi necessário o uso da biblioteca *pyrebase*, que permite trabalhar com o Firebase em *python*, de uma forma similar a outras linguagens de programação.

Capítulo 5
Validação e Testes

Ao longo do trabalho foram necessários realizar muitos testes, começando com as *features* que se pretendiam utilizar neste. Estes testes foram baseados na observação do comportamento das *features* quando aplicadas a diferentes eventos.

O separador de eventos tem como principais objetivos detetar, quando o motor se encontra a funcionar, isolar sons não estacionários e separar eventos com valores de frequência muito díspares. Para tal, foram testadas as *features* estudadas com o objetivo de observar o seu comportamento em casos práticos.

Para o primeiro objetivo, todas as características estudadas têm, teoricamente, o potencial de serem úteis, tal como a Figura 5.1 demonstra.

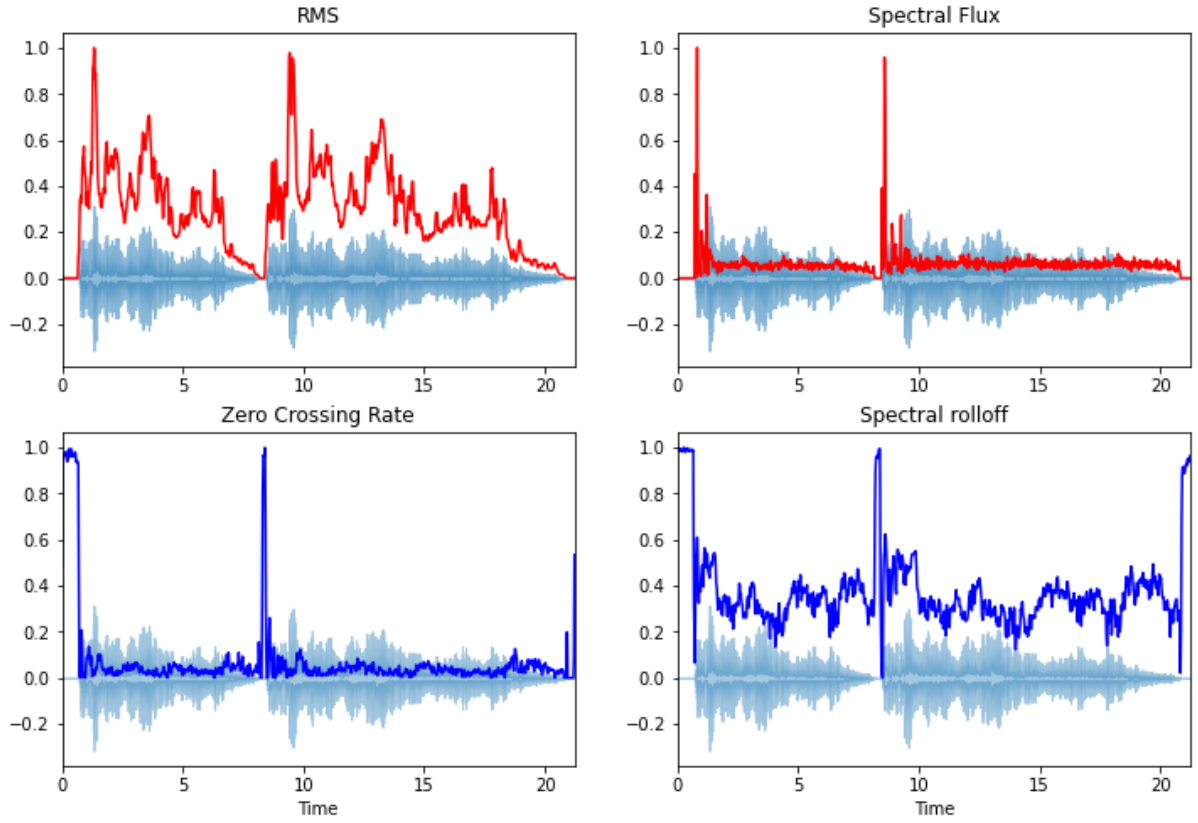


Figura 5.1 *Features* aplicadas a um sinal de um carro a iniciar duas vezes

Observa-se que, todas as *features* apresentadas detetam de forma correta o carro a iniciar. A RMS exibe valores quase nulos quando o carro não se encontra a funcionar e o *Spectral Flux* apresenta dois picos no início dos eventos. A *Zero Crossing Rate* e a *Spectral Rolloff* manifestam valores muitos elevados quando o carro está desligado (devido à presença de frequências elevadas do ruído). Apesar disto, a *Spectral Flux* demonstra não conseguir representar quando o motor é desligado e a *Spectral Rolloff* apresenta uma grande variação

quando o carro se encontra ligado.

A Figura 5.2 tem o objetivo de mostrar o comportamento das mesmas *features* focalizando num carro a desligar.

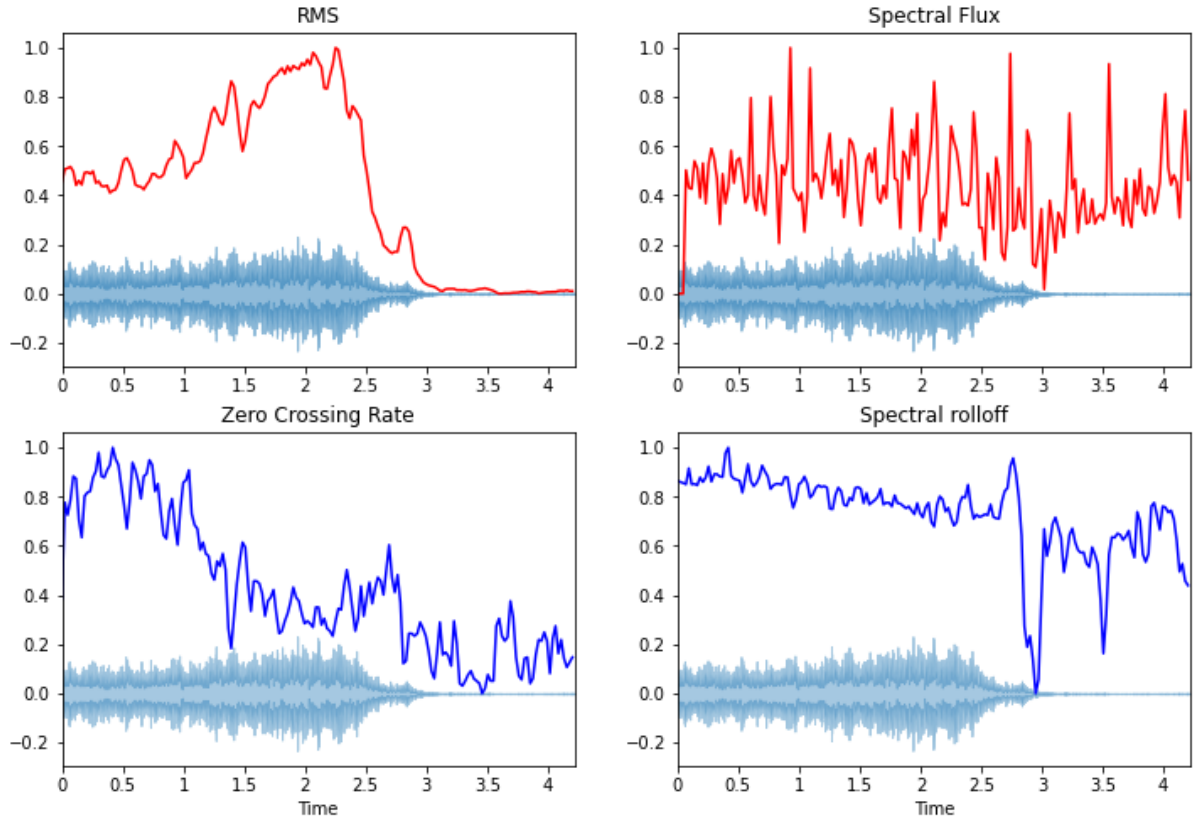


Figura 5.2 *Features* aplicadas a um carro a desligar

Na Figura 5.2, apenas a RMS descreve de forma correta o carro a desligar, pois todas as outras características não apresentam variações que possam ser justificadas a partir da conclusão retirada da Figura 5.1. Com base nestes e noutros testes, definiu-se que a RMS seria a característica a ser utilizada para identificar se o motor está a funcionar, embora esta *feature* também possa ser afetada por sons exteriores ao carro.

Para a identificação de sons não estacionários, como por exemplo o som da buzina, a característica que é teoricamente superior é a *Spectral Flux*, pois esta tem como propósito distinguir novos sons. Ainda assim, para validar este pressuposto esta foi posta à prova contra as outras características.

A Figura 5.3 representa um carro a funcionar. Durante este segmento são observados 4 sons de buzina: aos 4, 9, 27 e 30 segundos.

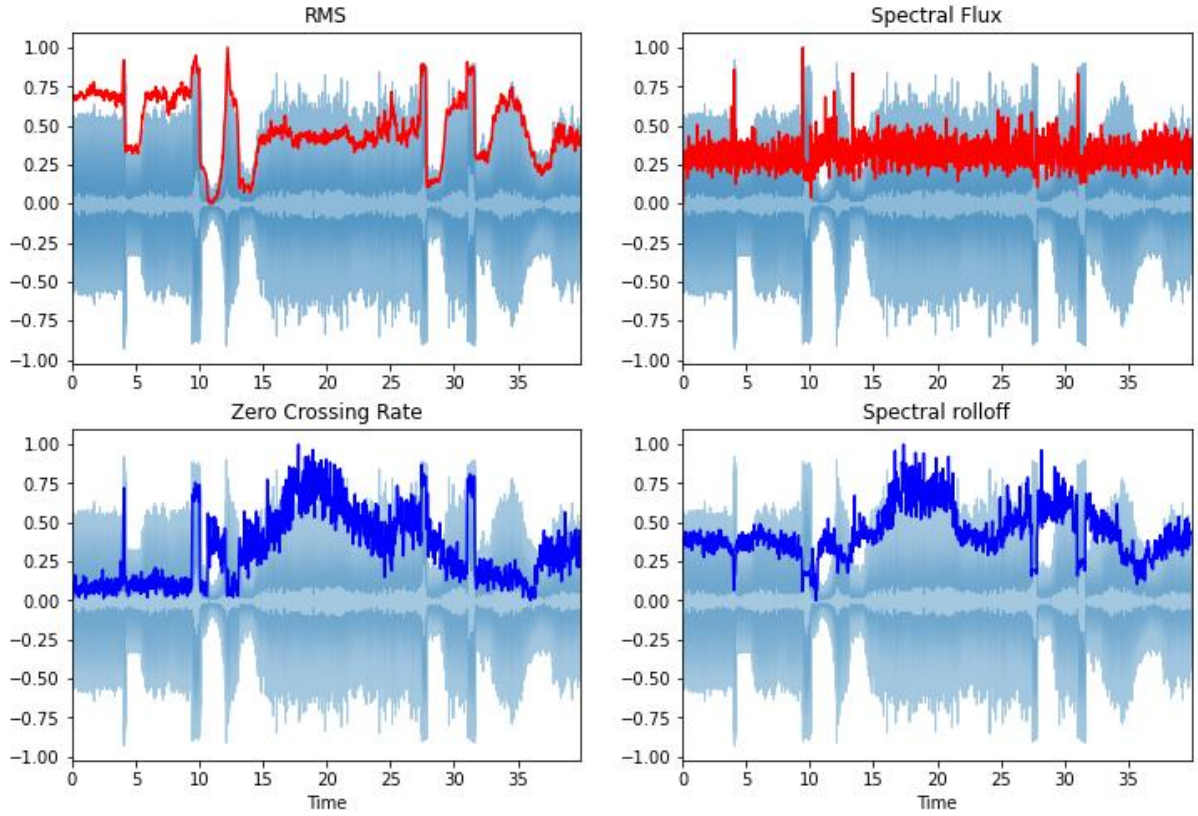


Figura 5.3 *Features* aplicadas a um som de motor com buzinas

Constata-se, que apesar da *Spectral Flux* não identificar todas as buzinas, quando comparada as outras *features*, é a que apresenta melhores resultados. É importante referir que a RMS, apesar da grande variação, aparenta apresentar ótimos resultados por as *features* se encontrarem normalizadas.

Devido à forma como a *Spectral Flux* é calculada, no conjunto das *features* estudadas, é a que fornece maior facilidade na escolha de *threshold*.

No sentido de avaliar a frequência produzida pelo motor, foram testadas todas as *features* baseadas em frequência, isto é, todas menos a RMS. A fim de se observar as diferenças de variação e dos valores obtidos das características, nas próximas figuras não é aplicada normalização aos resultados.

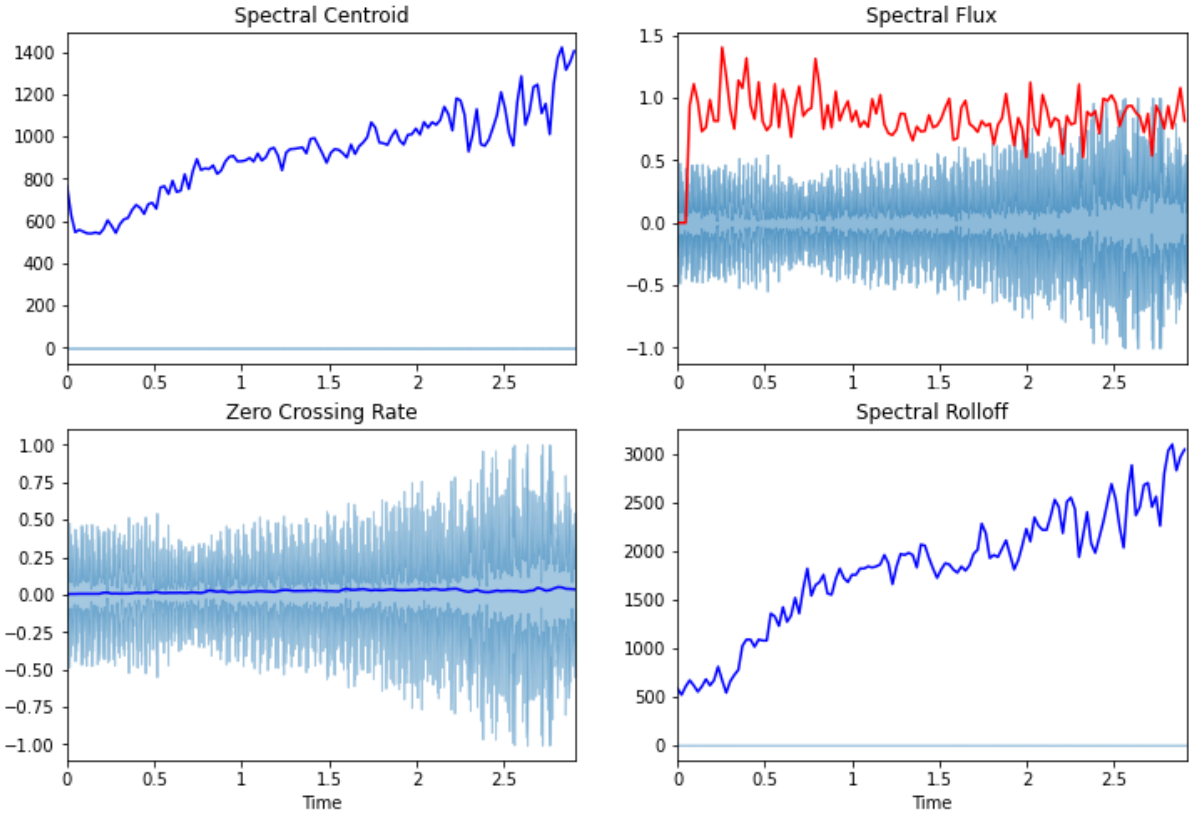


Figura 5.4 *Features* aplicadas a um carro a acelerar

A Figura 5.4 demonstra, que não só as melhores características para avaliar o comportamento do motor são as *Spectral Centroid* e a *Spectral Rolloff*, como também confirma, que embora os valores de ambas as características sejam muito diferentes, estas possuem variações similares. Tendo em conta, que ambas as características apresentam o mesmo propósito e para não haver redundância, é necessária a escolha de uma delas. A Figura 5.5 mostra o resultado da aplicação destas *features* ao som de um carro a acelerar e a trocar de mudança.

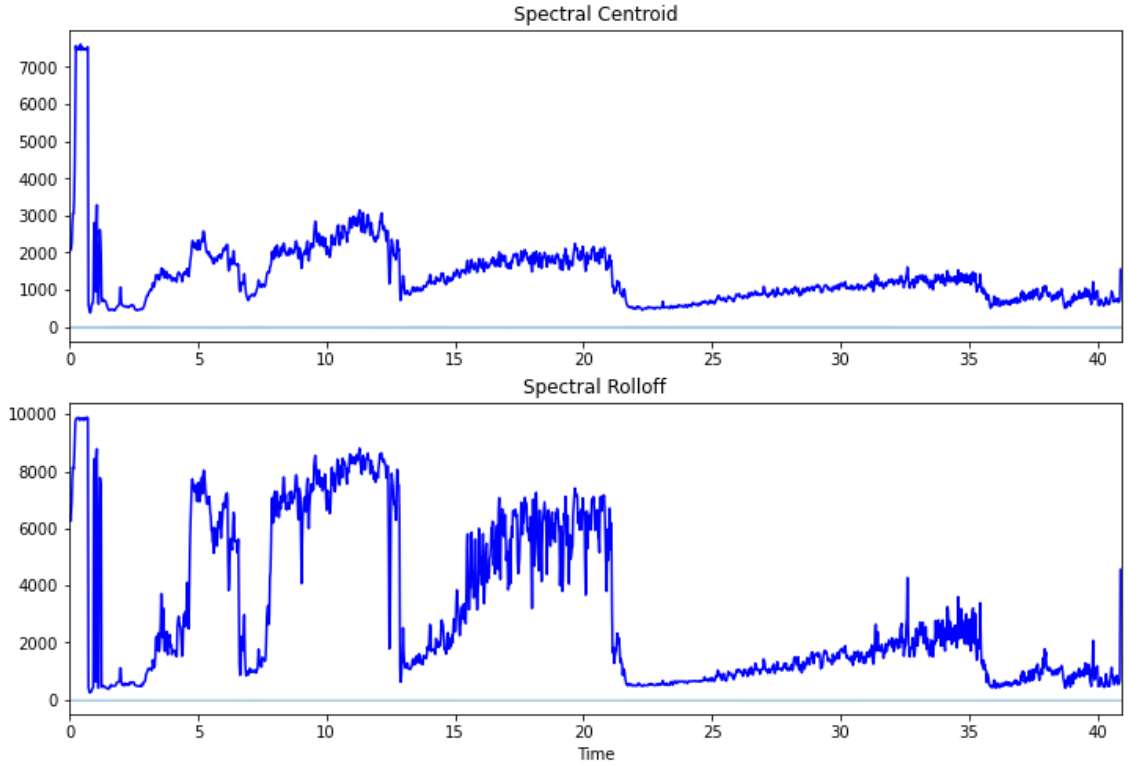


Figura 5.5 *Spectral Centroid* e *Spectral Rolloff* aplicadas a um carro a mudar de mudanças

Apesar de na maior parte das vezes a *Spectral Centroid* e a *Spectral Rolloff* apresentarem as mesmas variações, testes como representados na Figura 5.5, fizeram com que se escolhesse a *Spectral Rolloff*, para a avaliação da frequência, por esta ser mais previsível.

A *Spectral Rolloff* permite definir a percentagem de energia do espetro que se pretende obter (*roll percent*). A percentagem predefinida na biblioteca *librosa* é de 85%. Esta foi mantida, pois pretende-se ter a maior percentagem de energia possível, para descrever melhor o som produzido pelo motor. A partir de um certo valor deixa de ser útil aumentar a percentagem de energia.

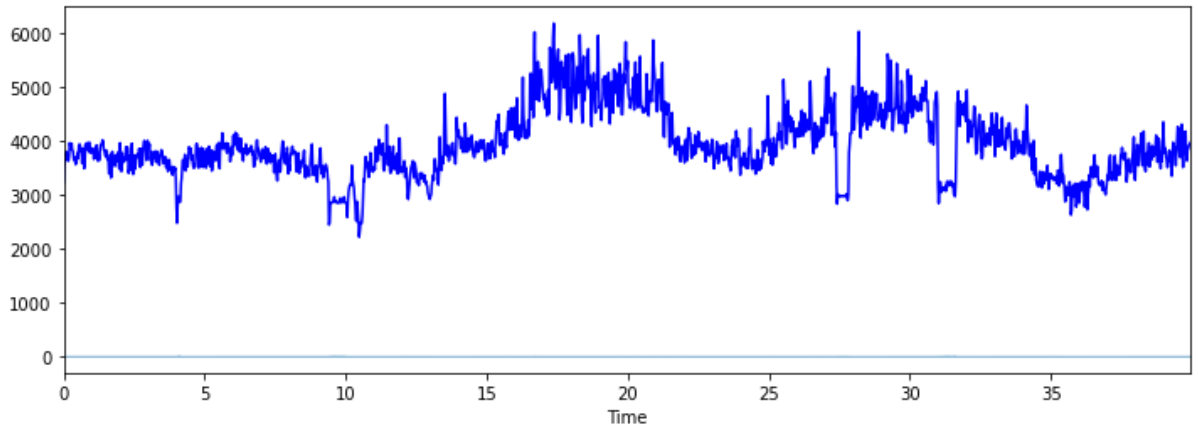


Figura 5.6 *Spectral Rolloff* com *roll* de 85% aplicada a um som de motor com buzinas

A Figura 5.6 representa a *Spectral Rolloff* com percentagem *default* do *librosa*, para o som das buzinas da Figura 5.3.

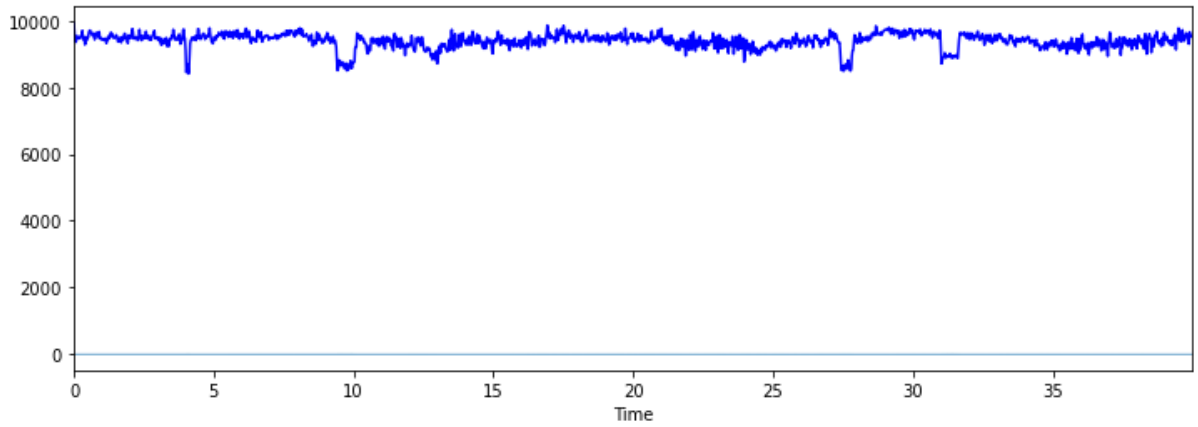


Figura 5.7 *Spectral Rolloff* com *roll* de 99% aplicada a um som de motor com buzinas

A Figura 5.7 representa a *Spectral Rolloff* com percentagem de *roll* de 99%, observa-se que a partir de uma certa percentagem a energia obtida começa a ser composta por sons que não são produzidos pelo motor. Isto é, uma parte da percentagem já representa a energia de ruído.

Para a *Spectral Rolloff*, ao contrário da RMS e da *Spectral Flux*, não se conseguiu determinar um ou mais *threshold*, por esta possuir uma variação muito díspar, dependendo dos sons testados e do tipo de motor onde se testa a *feature*. Nas Figuras 5.8 e 5.9 apresentam-se os valores de *Spectral Rolloff*, para um som de um carro movido a gasolina e para um som de um carro movido a gasóleo respetivamente, nos quais foi utilizado o mesmo percurso e tentou-se manter o mesmo regime de funcionamento.

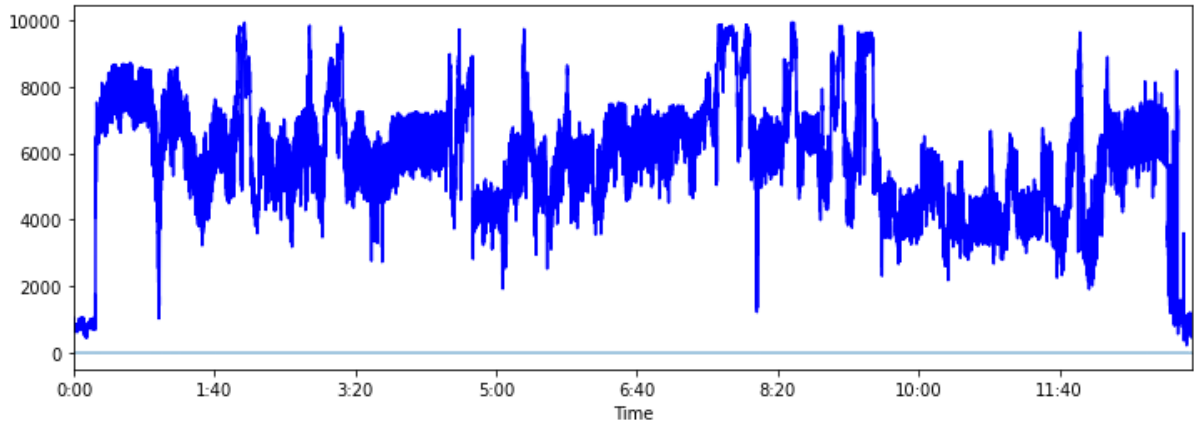


Figura 5.8 *Spectral Rolloff* aplicada a um carro movido a gasolina

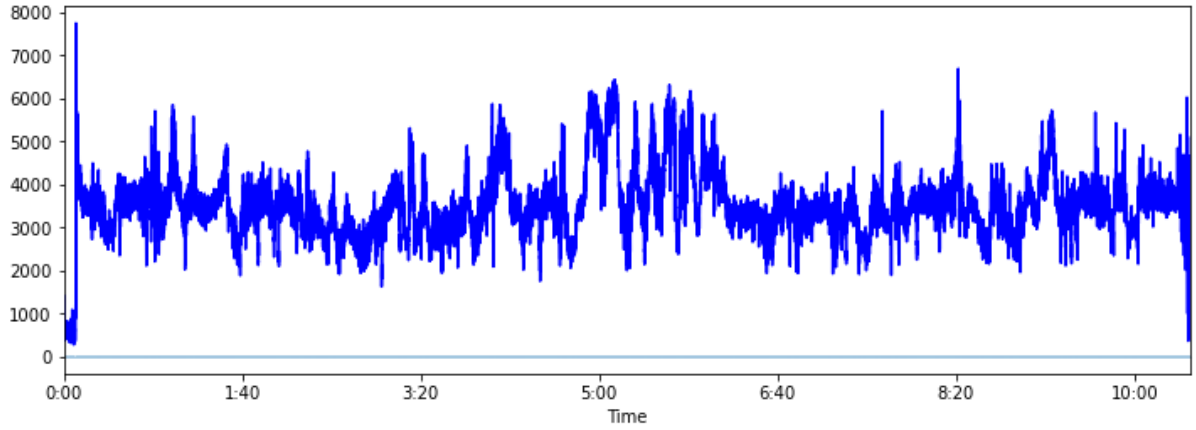


Figura 5.9 *Spectral Rolloff* aplicada a um carro movido a gasóleo

Como se pode observar os valores de *Spectral Rolloff* do carro movido a gasolina são muito diferentes do carro movido a gasóleo, por isso não se decidiu usar um *threshold* fixo, como nas outras características.

O classificador tem como tarefa classificar o evento passado a este. Para obter os melhores resultados possíveis é necessário escolher a característica que distingue o maior número de vezes, corretamente, as anomalias dos sons normais.

São comparadas três *features* a *Chromagram*, os MFCCs e a *Mel-Spectrogram*. Estas são avaliadas com base na sua *precision*, *recall* e *f1-score*. Como apresentado na Figura 5.10, a *precision* representa a proporção de identificações positivas que são corretas, a *recall* reflete a proporção de número de casos positivos que são identificados corretamente e a *f1-score* é obtida através da média ponderada da *precision* e da *recall*.

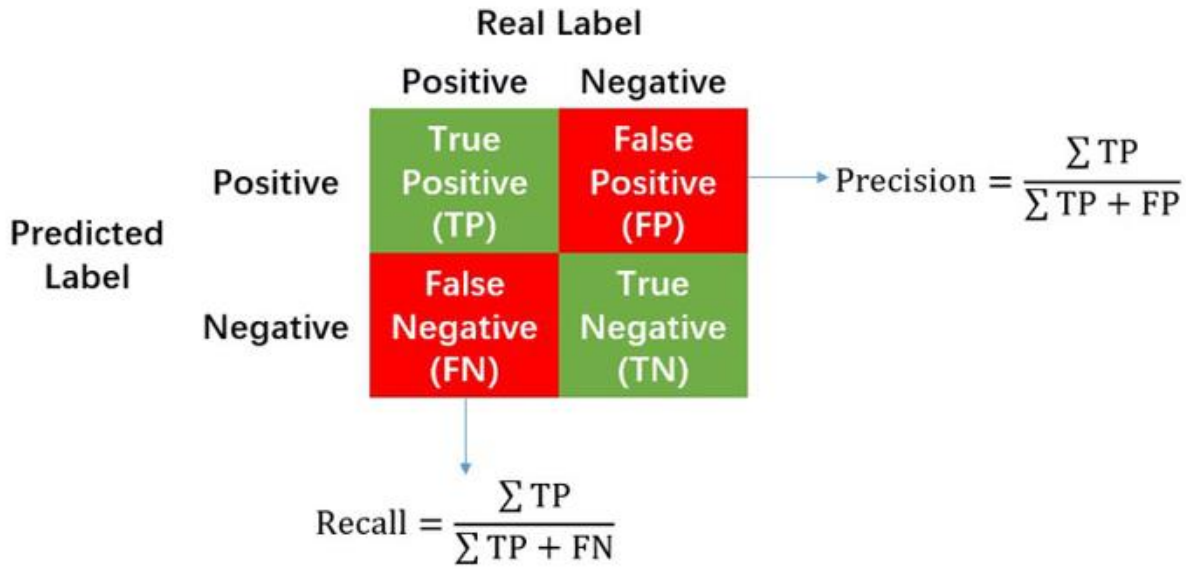


Figura 5.10 Cálculo da *precision* e da *recall*

Segue-se a Tabela 5.1 com os resultados das *features* testadas.

Tabela 5.1 Percentagem de acertos dos Modelos testados

Model MFCCs	*precision*	*recall*	*f1-score*
anomaly	0.99	1.00	0.99
engine	0.99	0.98	0.99
Model Mel-Spectrogram			
anomaly	1.00	0.99	0.99
engine	0.99	1.00	0.99
Model Chromagram			
anomaly	0.93	0.96	0.95
engine	0.95	0.92	0.94

Observa-se que as *features* com melhores resultados são a *Mel-Spectrogram* e os MFCCs. Mas é importante salientar que, todos os resultados finais encontram-se inflacionados, pois estes testes foram realizados a sons que são muito similares aos do *dataset* utilizado para o treino dos modelos.

Para além da percentagem de acerto do modelo, também é preciso ter em conta o seu tempo de execução. A Tabela 5.2 demonstra o tempo que os modelos demoraram a classificar um som de dez segundos, a escolha de um som com este tempo advém do facto de que os eventos separados terem uma duração máxima de dez segundos. Para estes testes foi utilizado um computador com uma memoria de 7,6 GiB e com um processador Intel Core i5-1035G1 CPU @ 1.00GHz x 8.

Tabela 5.2 Tempo demorado pelos modelos a classificar um som de dez segundos

Models	*Run Time*
Mel-Spectrogram	0.09 *seconds*
MFCCs	0.09 *seconds*
Chromagram	0.10 *seconds*

Constata-se que as analises realizadas pelas *features* não possuem encargos de tempo que possam afetar, de forma negativa, o funcionamento em tempo real da aplicação.

As Tabelas 5.1 e 5.2 apenas permitem confirmar que a *Chromagram* é a pior *feature* entre as estudadas. Para estabelecer a *feature* que seria utilizada pelo classificador foi necessária realização de mais testes.

Tabela 5.3 Classificação de anomalias pelos modelos

Sons	MFCCs		*Mel-Spectrogram*		*Chromagram*	
	Engine	*Anomaly*	*Engine*	*Anomaly*	*Engine*	*Anomaly*
Motor a funcionar	0.01	0.99	0.99	0.01	0.99	0.01
Motor inundado	0.01	1.00	0.00	1.00	0.01	0.99
Motor com a biela partida	0.00	1.00	0.00	1.00	0.51	0.49
Motor gripado	0.00	1.00	0.00	1.00	0.01	0.99
Motor com junta de velocidade constante partida	0.00	1.00	0.00	1.00	0.06	0.94
Motor com transmissão partida	0.99	0.01	0.00	1.00	0.22	0.78

Na Tabela 5.3 apresentam-se os resultados da aplicação dos modelos baseados nas diferentes *features* a sons de anomalias. Como se pode observar, o *Mel-Spectrogram* obtém os melhores resultados entre os modelos estudados. Com base nestes e noutros testes concluiu-se que a *feature Mel-Spectrogram* seria a mais indicada para classificação dos eventos.

Depois da aplicação ter sido implementada foram realizados inúmeros testes compostos por um motor a funcionar de forma correta e com sons de anomalias sobrepostos a este. No melhor dos casos a aplicação conseguiu identificar duas das cinco anomalias sobrepostas ao motor. Também se verificou que o som do motor a funcionar corretamente nunca foi considerado anomalia.

Por fim, pôs-se à prova o separador de eventos, realizando testes com a aplicação a separar o som sempre de cinco em cinco segundos e comparando o resultado destes com o resultado obtido com a máquina de estados. Nas Figuras 5.11, 5.12 e 5.13 observa-se um sinal retangular aplicado à representação de um sinal sonoro composto por cinco anomalias sobrepostas ao som de um motor a funcionar de forma correta. O sinal retangular apenas possui dois valores: zero e um. Quando este se encontra a um considera-se que existe anomalia, caso contrário não existe.

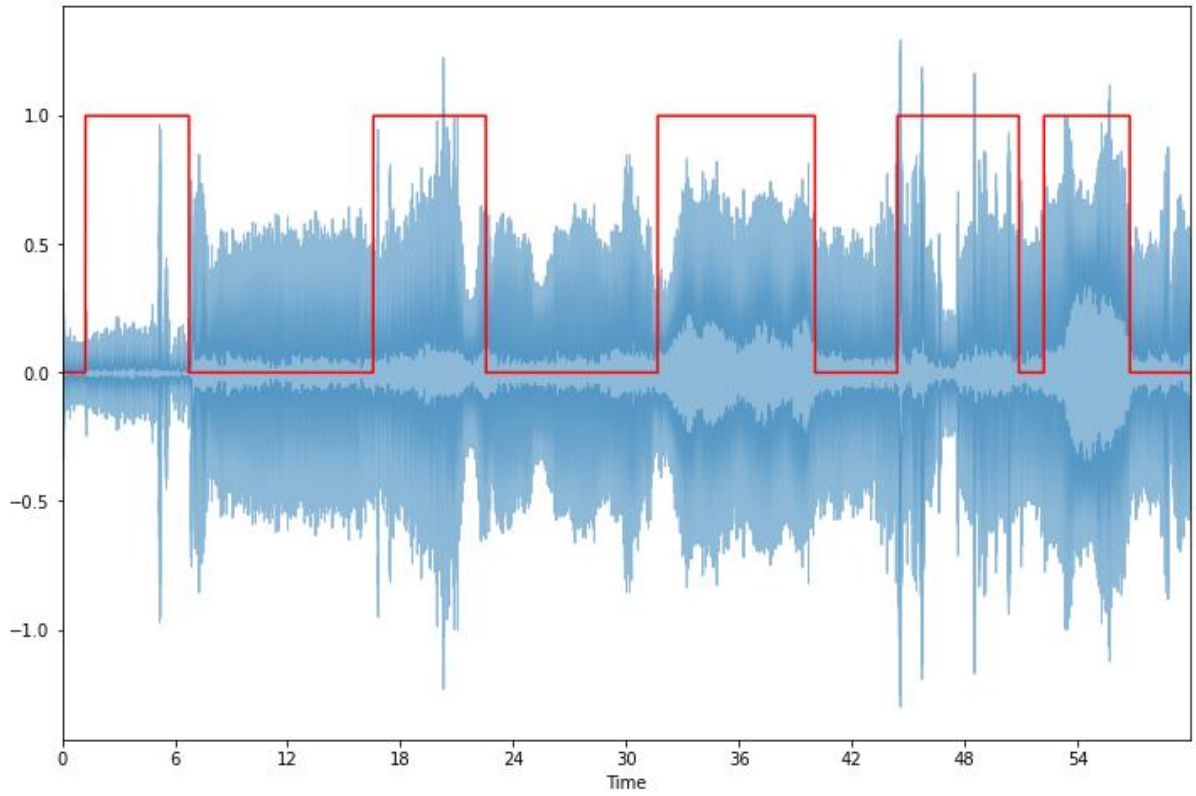


Figura 5.11 Anomalias identificadas corretamente

Na Figura 5.11 o sinal retangular representa onde realmente se encontram as anomalias sobrepostas.

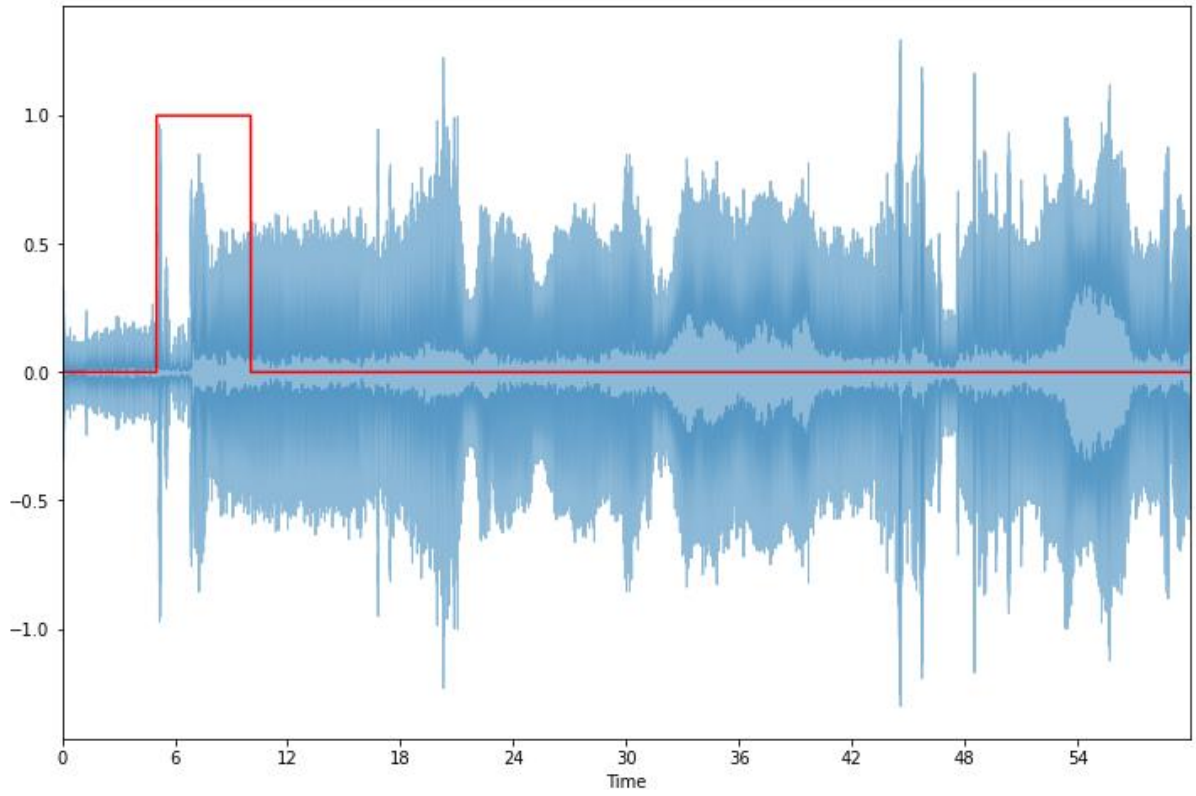


Figura 5.12 Anomalias identificadas pelo classificador sem separação de eventos

Na figura 5.12 o sinal retangular reflete a avaliação de segmentos, sempre, com cinco segundos. Constata-se, que o classificador apenas consegue identificar uma das anomalias presentes e que parte do evento identificado como anomalia inclui o som do carro a funcionar de forma correta.

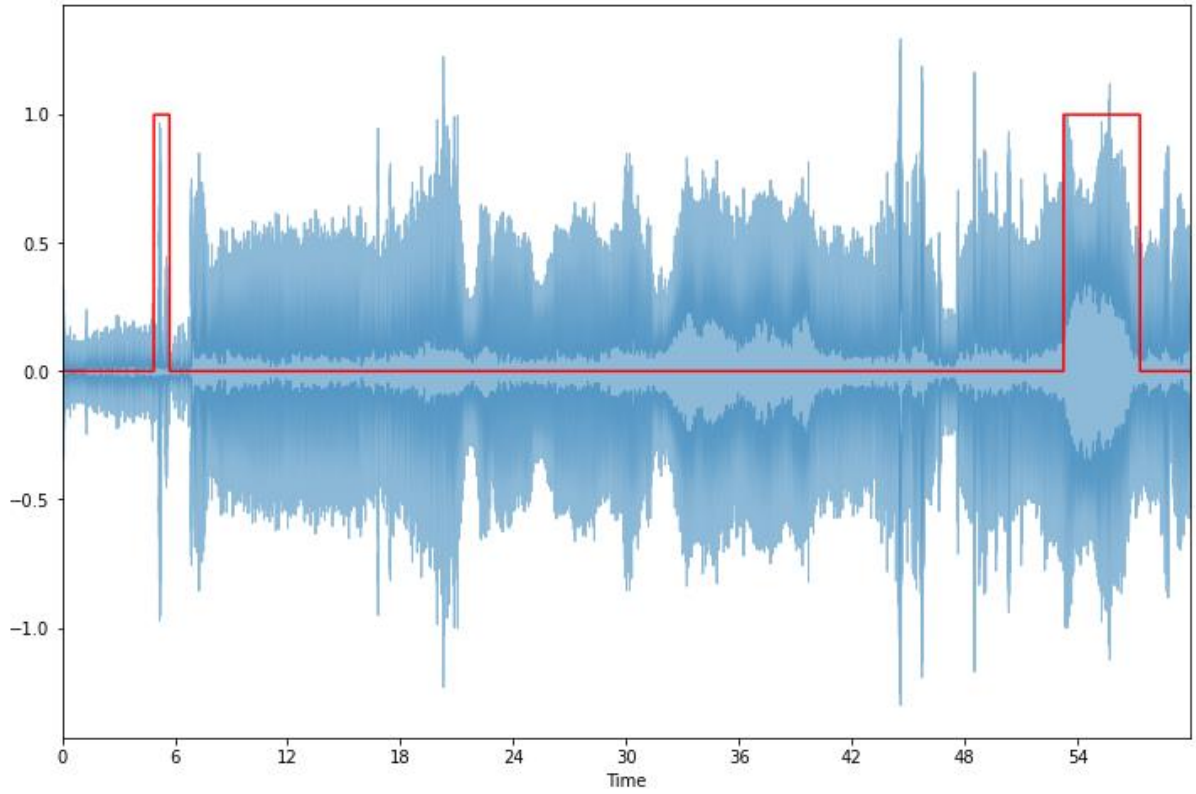


Figura 5.13 Anomalias identificadas pelo classificador com separação de eventos

Na Figura 5.13 o sinal retangular descreve a avaliação com a utilização do separador de eventos. Neste teste o classificador consegue identificar duas das cinco anomalias presentes e não classifica como anomalia o som do carro a funcionar de forma correta.

Conclui-se que quanto menor é o evento da anomalia maior é importância de utilizar o separador de eventos. E que a separação de eventos facilita a identificação de anomalias, pois sem esta o classificador teria de avaliar segmentos, muitas vezes, compostos por mais de um evento.

Capítulo 6
Conclusões e Trabalho Futuro

No âmbito deste projeto foram abordados os sons produzidos por motores de viaturas, com o objetivo de identificar se estes possuíam alguma anomalia. Esta identificação tem o propósito de fazer com que os condutores de automóveis sejam alertados para se deslocarem às oficinas apenas quando necessário, sem pôr em risco a sua segurança. No futuro, esta também poderá auxiliar a que carros autónomos "agendem" as suas próprias reparações.

Ao longo deste trabalho, foram estudadas inúmeras características, *features*, sonoras com o objetivo de se perceber como estas poderiam auxiliar na identificação de sons anómalos. Com base nas suas definições e formas de cálculo dividiram-se as *features* em dois grupos: um para separar eventos sonoros, composto por *Zero Crossing Rate*, *Spectral Centroid*, *Spectral Rolloff*, *Spectral Flux* e *Root Mean Square*; e outro para classificar os eventos separados, constituído por *Spectrogram*, *Mel-Spectrogram*, *Chromagram* e *Mel Frequency Cepstral Coefficients*.

Para este projeto, foi necessária a criação de um *dataset*. Este é composto por sons provenientes do *AudioSet* [4] e por gravações dos carros disponíveis aos elementos do grupo. Apesar de terem sido contactadas entidades externas, devido a questões de tempo e disponibilidade, não foi possível realizar a gravação dos veículos das mesmas.

No Capítulo 5 abordou-se a apresentação e discussão dos resultados. A conclusão aí retirada permitiu simplificar e melhorar a aplicação desenvolvida. Não só foram descartadas conclusões retiradas do estudo das características sonoras como foram validadas algumas das teorias propostas no início do projeto. Na parte da separação de eventos observou-se, que a RMS é a *feature* mais indicada para verificar se o motor do carro se encontra a funcionar, que a *Spectral Flux* permite a identificação de sons não estacionários e que a *Spectral Rolloff* é, entre as estudadas, a *feature* que melhor descreve a variação de frequência produzida pelo motor. Na classificação constatou-se, que a *Mel-Spectrogram* é a mais eficaz na identificação de anomalias, como se observa na Tabela 5.3, e o que o seu tempo de execução não impossibilita a aplicação de funcionar em tempo real.

Finalmente conclui-se, que a separação de eventos pode contribuir para um melhor desempenho do módulo de classificação, pois sem este o classificador teria de avaliar segmentos com mais do que um evento, como se pode observar nas Figuras 5.11, 5.12 e 5.13. O classificador poderá obter melhores resultados com um *dataset* mais diverso. Isto poderá auxiliar no desenvolvimento de novos projetos.

Como trabalho futuro propõem-se: (i) a implementação de um modelo de classificação do tipo *Autoencoder*, por este não necessitar que o *dataset* tenha sons de anomalias, (ii) a integração da aplicação, composta pelo separador de eventos e do classificador destes, num dispositivo IoT, devido à sua compacidade, que o permite colocar num local próximo do motor e protegido do ruído exterior. Também se sugere o estudo de diferentes *features* que possam ser úteis tanto para a separação de eventos como para a classificação. Igualmente, seria interessante realização de gravações sons de motores híbridos, com o objetivo de enriquecer o *dataset*, para o classificador se poder adaptar a diversos tipos de carros.

Apêndice A
Utilização de Códigos

Existem dois ficheiros de código implementado em *python* um *realTime.py* e um *fileReader.py*. O primeiro permite receber o sinal sonoro obtido pelo microfone do computador, através da biblioteca *PyAudio*, e para correr este programa é necessário utilizar o *prompt* usado para instalar as bibliotecas do *python* como o Anaconda Prompt, neste basta correr o comando: "*python realTime.py* email password". O segundo realiza a mesmas ações, que o primeiro código, mas ao invés de receber o sinal do microfone, lê um ficheiro de áudio, através do *librosa.load*, e a *thread* de *callback* do *PyAudio* é substituída por uma *thread* que lê o sinal obtido do ficheiro em segmentos e realiza *sleeps* (que demoram, mais ou menos, o mesmo que o tempo real) no meio dos segmentos para o utilizador e a *thread* consumidora terem uma sensação de *real time*. Para correr este código utiliza-se o seguinte comando: "*python fileReader.py* email password ficheiro" – sendo que o ficheiro tem de ser do tipo *wav*.

O código implementado para a aplicação Web corre num *server localhost* (habitualmente porto 80). Para correr a aplicação o utilizador tem que apenas abrir a página *html* num browser. Dependendo do sistema operativo o utilizador tem de instalar ou não programas para poder ter um *server* em *localhost*. No caso do Linux basta abrir a página *html*. No caso do Windows o utilizador deve instalar um programa como o Xampp e iniciar o Apache. Se o porto 80 já se encontrar ocupado com o serviço com PID 4, deve-se ir aos Services, encontrar a o serviço World Wide Web Publishing Service Properties, selecionar este e pôr o seu Startup type em Disable e o Service status em Stop, depois basta apenas salvar as alterações. Os ficheiros da aplicação deverão ser adicionados à pasta *htdocs*.

Quando o utilizador abrir a aplicação Web poderá criar ou entrar em uma conta, para ter acesso aos áudios guardados no Firebase. Na página My Audios o utilizador poderá ouvir todos os sons guardados para a conta em questão. Na página Duration Chart este poderá visualizar um gráfico de barras, que apresenta a duração das anomalias e pode ser clicado para alterar a unidade temporal. Por fim, o usuário, na página Number Chart, poderá ver um gráfico de barras, que representa o número de anomalias. Neste também pode ser alterada a unidade temporal, bem como ouvir o som representado nas barras em questão.

Recomenda-se o acesso a uma conta já existente, por esta já possuir alguns sons, com o seguinte email e password: "mail@mail.com" e "password".

Apêndice B
Contacto com Entidades Externas

Ao longo deste projeto verificou-se a quase inexistência de sons de anomalias de motores e de motores híbridos ou elétricos nos *datasets* disponibilizados online. Mesmo os sons de motores encontrados tinham o problema de não possuírem boa qualidade ou de serem gravados relativamente afastados do motor.

Como pretendia-se possuir um *dataset* com muitos sons e com sons de falhas de motores, foram procuradas algumas entidades do setor automóvel.

A Figura B.1 demonstra uma das tentativas de contacto com a oficina EVolution - Service & Battery. Esta entidade mostrou-se indisponível, devido à falta de existência de veículos nas imediações da mesma.

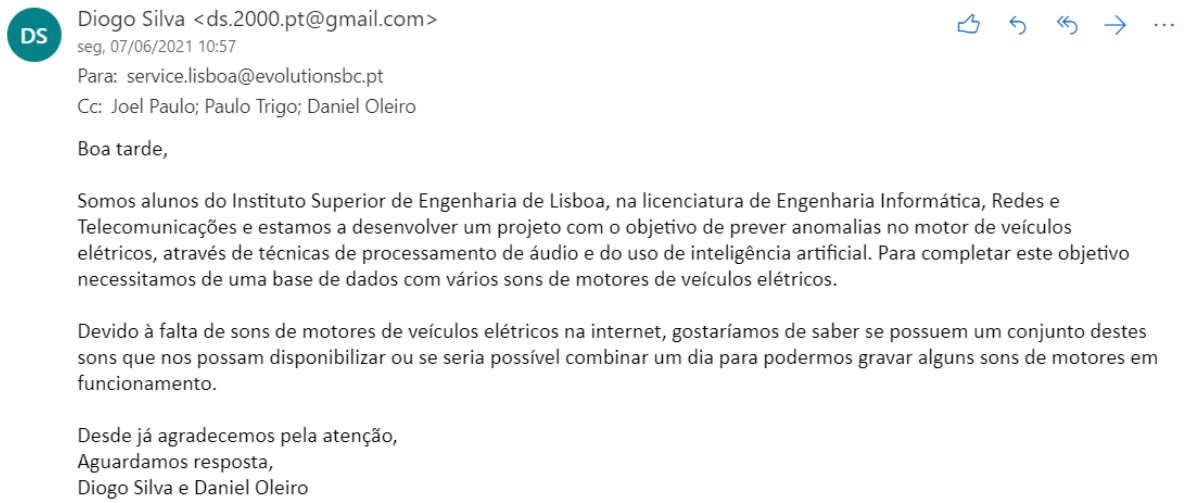


Figura B.1 E-mail enviado à oficina da EVolution - Service & Battery

Bibliografia

[1] Manutenção Preditiva vs. Preventiva: Qual É a Diferença?
https://blog.infraspeak.com/pt-pt/manutencao-preditiva-vs-preventiva/

[2] Feature (machine learning)
https://en.wikipedia.org/wiki/Feature_(machine_learning)

[3] Unsupervised Detection of Anomalous Sounds for Machine Condition Monitoring
http://dcase.community/challenge2020/task-unsupervised-detection-of-anomalous-Sounds-results

[4] AudioSet
https://research.google.com/audioset/

[5] Acoustic Anomaly Detection for Machine Sounds based on Image Transfer Learning
https://www.scitepress.org/Papers/2021/101858/101858.pdf

[6] Root Mean Square
https://en.wikipedia.org/wiki/Root_mean_square

[7] Spectral Rolloff
https://librosa.org/doc/0.8.0/geneRated/librosa.feature.Spectral_Rolloff.html

[8] Special Area Exam Part II, April 28, 2001
https://ccrma.stanford.edu/~unjung/AIR/areaExam.pdf

[9] Spectral Centroid
https://en.wikipedia.org/wiki/Spectral_Centroid

[10] Spectral Flux
https://en.wikipedia.org/wiki/Spectral_Flux

[11] Onset (audio)
https://en.wikipedia.org/wiki/Onset_(audio)

[12] Understanding the Mel Spectrogram
https://medium.com/analytics-vidhya/understanding-the-Mel-Spectrogram-fca2afa2ce53

[13] AudioSignalProcessingForML
https://github.com/musikalkemist/AudioSignalProcessingForML/blob/master/17%20-%20Mel%20Spectrogram%20Explained%20Easily/Mel%20Spectrograms%20Explained%20Easily.pdf

[14] Mel-frequency Cepstrum
https://en.wikipedia.org/wiki/Mel-frequency_Cepstrum

[15] Pitch class
https://en.wikipedia.org/wiki/Pitch_class

[16] Chroma feature
https://en.wikipedia.org/wiki/Chroma_feature

[17] Rede neural artificial
https://pt.wikipedia.org/wiki/Rede_neural_artificial

[18] Feedforward neural network
https://en.wikipedia.org/wiki/Feedforward_neural_network

[19] Keras - Dense Layer
https://www.tutorialspoint.com/keras/keras_dense_layer.htm

[20] Why Use Python for AI and Machine Learning?
https://steelkiwi.com/blog/python-for-ai-and-machine-learning/

[21] Lecture 18: Concurrency—Producer/Consumer Pattern and Thread Pools
https://www.cs.cornell.edu/courses/cs3110/2010fa/lectures/lec18.html

[22] np-rw-buffer
https://pypi.org/project/np-rw-buffer/

[23] Feature scaling
https://en.wikipedia.org/wiki/Feature_scaling

[24] The Sequential class
https://keras.io/api/models/sequential/

[25] Flatten layer
https://keras.io/api/layers/reshaping_layers/flatten/

[26] Layer activation functions
https://keras.io/api/layers/activations/

[27] Dropout layer
https://keras.io/api/layers/regularization_layers/dropout/

[28] Sobreajuste
https://pt.wikipedia.org/wiki/Sobreajuste

[29] Difference Between a Batch and an Epoch in a Neural Network
https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/

[30] Firebase: Uso e limites
https://firebase.google.com/docs/firestore/quotas

[31] Audio File Size Calculations
http://support.optiviewusa.com/dvr/archived/Analog_embedded-DVR/MANUAL/Audio%20File%20Size%20Calculations.pdf