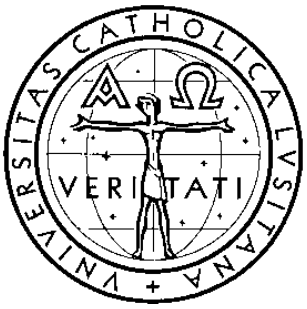
Escola das Artes da Universidade Católica Portuguesa

Mestrado em Som e Imagem



Caracterização de Sons Confortáveis e Stressantes Através da Aprendizagem Máquina

Mestrado em Design de Som 2015/2016

Manuel Fernando Batista Oliveira da Silva

Professor Orientador: Professor Doutor Pedro Duarte Pestana

Abril de 2016

Dedicatória

Quero dedicar este trabalho aos meus familiares e namorada que sempre me apoiaram durante este percurso. Dedico também a todos os músicos e investigadores da área, pois foram estes que me empurraram a seguir o caminho em que me encontro.

Agradecimentos

Agradecimento principal para o orientador desta dissertação e para todas as 30 pessoas que participaram nos questionários, que contribuíram para o sucesso desta.

Agradecimento a todas as pessoas da UCP que estiveram diretamente envolvidas no trabalho realizado, ou que contribuíram para o seu sucesso.

Agradecimento também a todas as pessoas que de algum modo apoiaram a realização do trabalho, ou que contribuíram para o seu sucesso.

Resumo

Nesta investigação, procurámos fazer uma caracterização dicotómica de sons Confortáveis ou Stressantes, através do uso de aprendizagem máquina. Para além de trazer à luz informações sobre características subjacentes a sons simples que criem uma valência subjectiva para o ouvinte médio, imaginamos que os resultados deste tipo de classificação podem contribuir para a criação de um sistemas de aconselhamento na criação de *design* sonoro para interfaces com o utilizador em produtos ou aplicações. Para o desenvolvimento do sistema foi necessário criar um *dataset* temático. Foram depois extraídos descritores áudio de baixo nível, de cada exemplo do *dataset.* Finalmente, utilizámos estes dados para alimentar algoritmos de aprendizagem máquina. Os resultados foram avaliados à luz das estratégias comuns em sistemas de *Music Information Retrieval* (MIR) e indicaram a possibilidade da criação de um sistema automático de caracterização sonora.

Palavras-chave: Aprendizagem Máquina, Categorização Sonora, Valência Sonora, Computação Afetiva, *Music Information Retrieval*, *Datasets*.

Abstract

In this research, we have looked into a dichotomous characterization of sounds as either "Comfortable" or "Stressful", through the use of machine learning. In addition to bringing light to information about underlying features the simple sounds that create a subjective medium for the listener, we envision that the results of this type of classification can contribute to the creation of an advisory system for the creation of sound design in user interfaces for products or applications. For the development of the system it was necessary to create a dataset. Low-level audio descriptors were then extracted for each instance of the dataset. Finally, we have used this data to feed machine learning algorithms. The results were evaluated in light of the common strategies in Music Information Retrieval (MIR) and indicated the viability of setting up an automatic sound characterization system.

Key Words: Machine Learning, Sound Categorization, Valencia, Affective Computing, Music Information Retrieval, Datasets.

Índice de Conteúdos

Lista de Figuras

Lista de Tabelas

Glossário

Dataset – é uma coleção de dados normalmente tabulados. Por cada elemento (ou indivíduo) são indicadas várias características.

Transformada de Fourier – É usada para decompor um sinal do domínio temporal em frequências.

Escala de Mel – é uma escala percetual de tonalidades que converte frequências Hz em mels *m*. O ponto de referência entre esta escala e uma escala normal de medida de frequências é definido ao assumirmos uma tonalidade percetual de 1000 mels para um tom de 1000 Hz. Quando superior a 500 Hz, intervalos maiores são assumidos como incrementos tonais iguais. Com isto, quatro oitavas na escala de *hertz* acima de 500Hz são assumidas como duas oitavas na escala de mel. Esta foi criada em 1937 por *Stevens*, *Volkmann* e *Newman*.

SVM - *Suport Vector Machines* (Máquina de Vetores de Suporte) são dos algoritmos mais populares em aprendizagem máquina para classificação, regressão entre outras tarefas de aprendizagem. Foram criados por *Vladimir Vapnik* em 1979, na sua forma mais linear, o SVM é uma linha que separa um grupo de exemplos positivos de um grupo de exemplos negativos.

Cluster – é um conjunto de objetos de um *dataset* agrupados de maneira a que estes sejam mais próximos a eles mesmos do que a outros conjuntos. Estes têm como objetivo explorar a mineração de dados feita de um *dataset*.

Classificador binário – é um classificador que consegue aprender e distinguir apenas duas classes ou duas opções, e.g. Confortante ou Stressante; Masculino ou Feminino; Sim ou Não; 1 ou 0.

Kernel – é um "truque" matemático usado para evitar o mapeamento específico que é necessário para que algoritmos lineares consigam aprender funções ou decisões não lineares.

1. Introdução

A utilização de aprendizagem máquina para a caracterização e classificação de padrões tem crescido bastante nos últimos anos, pelo facto de ajudar a resolver problemas complexos no mundo real que podem envolver um número elevado de variáveis. Entre estes estão a análise de expressões genéticas, o reconhecimento de imagens ou objetos em três dimensões, e, evidentemente, a classificação de sons ou objetos musicais. O assunto, no seu âmbito mais geral, está necessariamente ligado com a multiplicidade de acesso a dados e a expansão de armazenamento dos mesmos ("*big data*"). A intratabilidade de conjuntos de dados tão vastos tem levado a um desenvolvimento de sistemas que permitam a descoberta de informações úteis dentro de coleções excessivamente grandes e desorganizadas (Brynjolfsson & Mcafee, 2014).

A presente dissertação enquadra-se nesta temática de inovação, pretendendo um contributo ao universo da computação afetiva (a interpretação de emoção e características humanas por parte do computador) direcionada para o contexto sonoro, com aplicações a *sound design.* Nesta secção, vamos apresentar o tema da investigação e o projeto final que lhe diz respeito. Pretendemos igualmente fazer a descrição da metodologia utilizada e uma descrição da estrutura da restante dissertação.

1.1 Apresentação do tema de investigação e do projeto final

Ao longo da presente dissertação descreveremos uma tentativa de caracterização bipolar de sons em categorias dicotómicas Confortável-Stressante, através de aprendizagem máquina. Temos como objetivo secundário perceber o quão fiável é o processo de aprendizagem, partindo do uso de um *dataset* sonoro (contendo 100 exemplos sonoros relacionados com a área dos dispositivos móveis), criado no contexto da Unidade Curricular de projeto. Criámos um objeto de interface com o utilizador, ao qual chamámos "Boxi", que nos permitiu a simplificação da caracterização do *dataset*, assistindo o processo de classificação manual, em que 30 pessoas fizeram a sua aferição subjetiva. Posteriormente, alimentámos o *software* WEKA (*Waikato Environment for Knowledge Analysis*) (Hall et al., 2009) com os dados, o que nos permitiu ensinar os modelos de aprendizagem, e comparar os resultados obtidos da classificação humana, e da classificação através da máquina.

Desde o aparecimento das máquinas que o homem as tenta explorar ao máximo, em busca do limite na possibilidade de transposição de tarefas humanas. Um dos princípios mais

emblemáticos é o de aprender, um fundamento principal do da aprendizagem máquina, um subcampo da inteligência artificial que se dedica ao desenvolvimento de algoritmos e técnicas que permitem ao computador aprender, isto é, permitem a este aperfeiçoar o seu desempenho em alguma tarefa. Na inteligência artificial existem dois tipos de raciocínio:

- Indutivo: extrai regras e padrões de grandes conjuntos de dados que usa para aprender e confirmar algo que lhe é pedido.

- Dedutivo: é um processo de raciocínio que partindo de uma ou mais afirmações, chega a uma conclusão lógica.

Mas, na aprendizagem máquina apenas se usa o processo indutivo, pois este é o raciocínio que vai permitir à máquina aprender e responder a problemas com argumentos (Monard & Baranauskas, 2003).

No nosso caso, ao falarmos em som, direcionamo-nos mais para os sons afastados da indústria musical, não querendo com isto dizer, que tenhamos posto os classificadores musicais à parte. Estes tiveram grande peso na nossa investigação, sendo que é dentro da interpretação de características musicais que a aprendizagem máquina se tem desenvolvido grandemente, como pode ser observado no estado de arte.

1.2 Metodologia e Calendarização utilizada para a Investigação

A metodologia referente à categorização de sinais áudio em classes dicotómicas está bastante estandardizada, tendo evoluído ao longo das últimas duas décadas em paralelo com o desenvolvimento da *International Society for Music Information Retrieval* (ISMIR) que tem todos os anos uma conferência e um desafio aberto à comunidade (MirEx) para avaliação de algoritmos que completem tarefas relevantes dentro da classificação sonora. É de notar que, como o próprio nome do organismo indica, o conceito de sonoro é muitas vezes afunilado para o conteúdo musical. No nosso caso é importante testar também a adequação de uma metodologia que bebe bastante de conceitos musicais ao ser aplicada a sons simples, desprovidos de musicalidade, pelo menos como característica principal.

Toda a investigação foi pensada para ser dividida em cinco etapas principais, que depois se vieram a distribuir num total de catorze etapas, como podemos verificar na Tabela 1 “Cronograma de Tarefas”.

A primeira seria uma pesquisa aprofundada sobre Aprendizagem Máquina em Áudio. Nesta pesquisa tínhamos de perceber se estaria ao nosso alcance a criação de um *software*. De raiz, ou se poderíamos depender de *software* comercial/académico. Estava ainda em aberto neste ponto o foco específico do nosso trabalho, havendo ainda abertura para, dentro da temática geral, mudar o enquadramento específico.

A segunda etapa seria a construção ou seleção do *Dataset*. Num contexto de aprendizagem artificial, é tipicamente necessário ter um conjunto de dados (neste caso, sons) realista, e que possa ser classificado por utilizadores. Poderíamos depender de *datasets* disponíveis na comunidade académica, mas uma pesquisa preliminar permitiu aferir que o que nos interessava não estava disponível. Tivemos então de explorar questões como: que sons utilizar para serem caracterizados pelo *software*? Quantos sons seriam suficientes para ter uma boa margem de acerto? E qual a área temática dentro da qual os selecionaríamos. Estas questões foram respondidas na Pré-produção, como pode ser observado na parte de Desenvolvimento, subcapítulo 3.1.1: Pré-Produção.

Na fase seguinte estudamos aspetos relacionados com interface com o utilizador e construímos o objeto "Boxi", com o propósito de simplificar o processo da classificação manual, e, ao mesmo tempo, tornar esta classificação numa experiência interativa para os utilizadores, visto que se trata de uma tarefa potencialmente complexa e aborrecida, em que a capacidade de manter um nível elevado de concentração é essencial para o sucesso do resultado final.

A quarta etapa passou pela extração dos descritores de áudio de baixo nível, partindo do uso do *software* MARSYAS (*Music Analysis, Retrieval and Synthesis for Audio Signals*). Foram extraídos os seguintes descritores: MFCC (*Mel Frequency Cepstral Coefficients*), *Spectral* Centroid, Spectral *Flux*, *Spectral Roll-off*, *Zero Crossing Rate* e *Chroma*. Os mesmos são explicados no Estado de Arte, subcapítulo 2.2: Descritores Áudio de Baixo Nível.

A quinta e última etapa consistiu na passagem destes dados extraídos com o MARSYAS para o WEKA e a análise dos resultados obtidos a partir da classificação dos algoritmos do WEKA como: C4.5; Naive *Bayes*; SMO (*Sequential Minimal Optimization*); LibSVM (*Library for Support Vector Machines)*; e por último IBk (*K-nearest neighbours*). Estes algoritmos são explicados com mais detalhe no Estado de Arte, subcapítulo 2.3: Algoritmos de Aprendizagem Máquina.

Na revisão da literatura procurámos aferir a medição objetiva e quantificada, de resultados obtidos através do uso da aprendizagem máquina em áudio. Precisávamos também de uma análise em profundidade dos processos requeridos de como aplicar o nosso *dataset* ao processo. Com isto, foi realizada uma investigação dividida em duas etapas: a primeira, uma leitura de livros e artigos relacionados com todos os temas abordados nesta investigação; a segunda, a leitura de manuais e tutoriais dos *softwares* usados para a investigação. A primeira teve bastante peso na investigação, visto que nos permitiu conhecer o estado atual da aprendizagem máquina aplicada ao áudio, e a sua história, oferecendo-nos também diferentes pontos de vista e inspiração, mas a segunda etapa foi sem dúvida indispensável para a conclusão desta, pois sem esta, iria ser impossível recolher os resultados obtidos.

1.3 Descrição da estrutura da dissertação

Esta dissertação está dividida em quatro capítulos: Introdução, Estado de Arte, Desenvolvimento e Conclusão. No segundo capítulo iremos descrever o que já existe na literatura sobre os temas que abordamos na investigação e introduzir os conceitos básicos subjacentes a alguns deles. No terceiro exploramos com detalhe todo o trabalho produzido durante a investigação, apresentando resultados e oferecendo uma análise exploratória dos mesmos. Por último, iremos concluir a nossa investigação, apontando também para direções adicionais.

Cronograma em Meses/Semanas																																			
	Out.				Nov.				Dez.				Jan.				Fev.				Mar.				Abr.											
Atividades e Objectivos	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª	1ª	2ª	3ª	4ª							
Investigação Liturgica	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+								
Proposta de Investigação	+	+	+	+	+	+	+	+	+	+	+																								
Criação de Software de Aprendizagem Máquina	+	+	+	+	+	+	+	+																											
Criação do *Dataset*												+	+	+	+			+	+																
Criação do objeto Boxi																+	+	+	+	+															
Testes de Classificação com o objeto boxi																	+	+																	
Classificação Manual do *Dataset*																						+	+												
Testes de Extração dos Descritores de Áudio de Baixo Nivel																							+												
Extração de Descritores de Áudio de Baixo Nivel																					+														
Analise dos resultados obtidos da Extração.																						+													
Testes com o *Software WEKA*																					+														
Analise dos resultados obtidos do *Software Weka*																									+	+									
Documentação de toda a Investigação em formato de Dissertação													+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+							
Finalização e entrega de toda a investigação																												+							

Tabela 1.1 - Cronograma De Tarefas

2 Enquadramento, conceitos, e Estado da Arte

Neste capítulo, vamos explorar quatro subtemas diferentes. O primeiro refere-se aos conceitos básicos de Aprendizagem Máquina, o segundo que explorará descritores de áudio de baixo nível. No terceiro iremos detalhar sistemas automáticos e algoritmos de aprendizagem máquina aplicados especificamente ao som, e, por último, exploramos com detalhe os *softwares* usados nesta investigação.

2.1 Conceitos Básicos de Aprendizagem Máquina

As diferentes técnicas de aprendizagem de máquina utilizam o princípio de raciocínio indutivo. A partir deste, é possível obter conclusões genéricas partindo de dados fornecidos. O raciocínio indutivo pode ser de dois tipos: com supervisão ou sem supervisão.

Numa estratégia sem supervisão não existe a presença de um "professor externo", isto é, não é oferecido à máquina um *dataset* previamente definido e manualmente classificado, para que esta possa aprender sobre a questão em causa. Sem este exemplo, o algoritmo vai aprender com a ajuda de uma medida de qualidade. Normalmente, esta técnica é usada quando queremos que o algoritmo encontre padrões e/ou tendências num conjunto de dados.

Numa estratégia com supervisão existe a presença do "professor externo", que fornece um *dataset* que foi previamente classificado e ensina à máquina as hipóteses que tem de classificação. O algoritmo, partindo deste *dataset*, extrai o conhecimento necessário, para poder gerar futuras entradas e saídas (Andrieu, de Freitas, Doucet, & Jordan, 2003).

Na nossa investigação seguimos a opção de um sistema com supervisão, porque para além de ser o mais favorável para caracterização e classificação, tínhamos também como objetivo acessório do nosso projeto final a criação de um *dataset* e a sua classificação manual Um dos aspetos importantes é o estudo das classes do *dataset* que vai ser analisado pela aprendizagem máquina. Isto é, se os valores das classes assumirem valores discretos 1,2,3,…, K, diz-se que é um problema de classificação, caso os valores das classes sejam contínuos, é um problema de regressão. Num problema de classificação, se o valor de $K = 2$, então o problema passa a ser binário, se o $K > 2$, passa a ser um problema de classes múltiplas. Cada exemplo do *dataset* é composto por um vetor de características, e cada uma dessas

características (atributos) pode ser do tipo nominal (categórico) ou contínuo. Os atributos categóricos só podem assumir uma quantidade finita de valores, e.g. género, um atributo categórico que só pode assumir dois valores: Feminino ou Masculino. Quando um atributo é contínuo, podemos definir uma ordem linear sobre os seus valores. No entanto, estes podem ser transformados eventualmente em atributos categóricos, e.g. o peso pode ser um atributo contínuo enquanto ele pertencer aos números reais, mas o mesmo pode se transformar num atributo categorizado sendo caracterizado como: leve ou pesado (Lorena & de Carvalho, 2007).

Outro aspeto importante que se deseja ao utilizar aprendizagem máquina, é a capacidade do algoritmo que vai classificar minimizar os *outliers*, estes são exemplos isolados que são, demasiado distintos ficando assim afastados dos restantes da série, ou que são inconsistentes. Estes trazem implicações como prejuízos na interpretação dos resultados. Os mesmos podem ser identificados por vários métodos e.g. desvio-padrão. Neste método, um exemplo será considerado um *outlier*, se este encontrar uma determinada quantidade de desvios-padrão da média no exemplo. A quantidade de desvios-padrão varia conforme o tamanho do *dataset* (Gladwell, 2008).

Na figura seguinte, representamos de forma simplificada e visual os conceitos do funcionamento da aprendizagem máquina. Nessa figura temos um grupo de *n* dados. Cada *x* dado possui *m* atributos, isto é, $x = (x1,\ldots,xm)$. As variáveis *y* representam as classes e o algoritmo de aprendizagem máquina vai extrair e gerar um classificador $f(x)$.

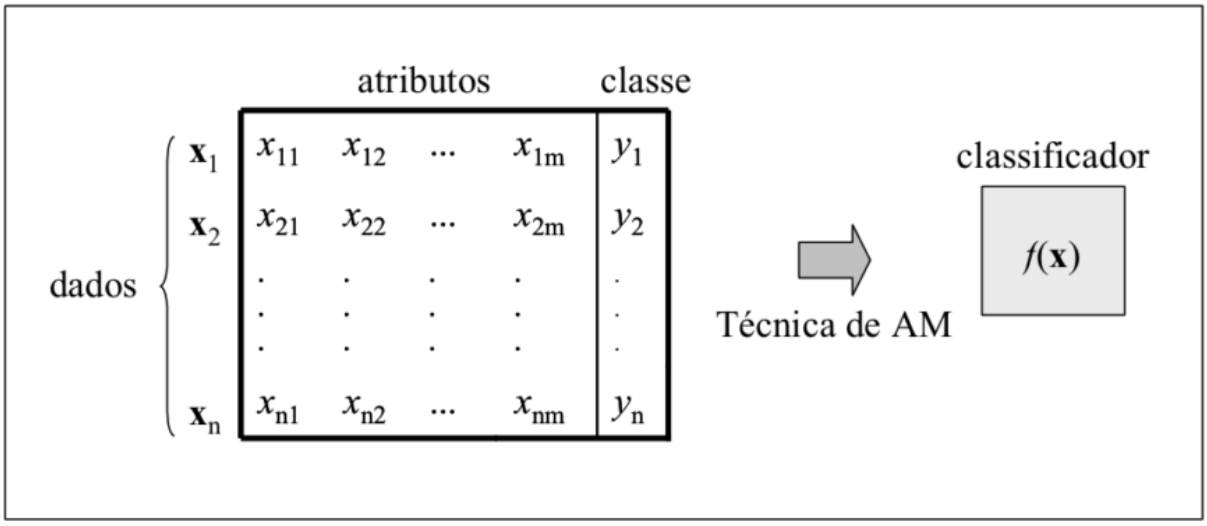


Figura 2.1 - Raciocínio Indutivo de um classificador em Aprendizagem Máquina. Retirado de (Lorena & de Carvalho, 2007)

2.2 Descritores Áudio de Baixo Nível

Descritores áudio de baixo nível são aqueles que caracterizam o som pelos seus atributos básicos, extraindo informações diretas do um sinal sonoro (Baume et. al., 2014). Não são termos que possam ser aplicados diretamente a conceitos de perceção, embora alguns tenham correlações próximas com mecanismos percetuais.

Um sinal áudio é bipolar, e no universo digital costumamos determiná-lo como oscilando no intervalo [-1,1], refletindo uma evolução análoga às variações de pressão que constituem o fenómeno acústico. Cada som é assim representado digitalmente como um longo vetor de números, sobre os quais podemos fazer operações. O desvio padrão deste vetor, por exemplo, estará necessariamente relacionado com a amplitude do som, e, embora seja um exemplo simples de um descritor de baixo nível, é normalmente irrelevante na explicação de fenómenos complexos. Utilizamos maioritariamente descritores que funcionam no domínio temporal (diretamente sobre o sinal) ou no domínio espectral ou das frequências (sobre a transformada de Fourier). Particularmente devido à forma específica como funciona a transformada de Fourier em termos práticos em áudio, é usual que os descritores de baixo nível sejam analisados não no sinal inteiro mas em subdivisões (*frames*) deste. Assim, cada descritor é ele próprio um vetor que descreve o seu comportamento e variação ao longo de várias *frames*. Para simplificar ao máximo o descritor utilizamos momentos estatísticos como a média do vetor ou o desvio-padrão do vetor ao longo de todas as *frames*, de forma a podermos apresentar apenas um número como resultado.

Existem diversos *softwares* de extração de descritores áudio. Desde bibliotecas para o MATLAB, (MIR *Toolbox,* MA *Tollbox* e *PsySound*) a códigos fonte abertos como YAAFE, CLAM, *LibXtract*, *Aubio*, *Sonic Annotator/Vamp Plugins* e MARSYAS. Contudo, ainda há dificuldade em saber se existem diferenças entre os mesmos descritores do mesmo *dataset*, extraídos de *softwares* diferentes (Baume et. al., 2014). Partindo do estudo de *Baume* et. al. (2014) onde os autores fazem uma análise dos descritores mais úteis em contexto sonoro, decidimos extrair a coleção referida na listagem abaixo. Explicamos o seu funcionamento geral,

sem entrarmos em detalhe das equações matemáticas por detrás destes, pois não são nosso ponto de foco. O leitor interessado poderá consultar Lerch (2012).

- *Mel Frequency Cepstral Coefficients.* Este é um conjunto de descritores muito relevantes em sistemas de reconhecimento de voz em áudio, devido a sua eficácia na representação das variâncias do espectro dum sinal de áudio. Este descritor é exposto normalmente em quatro etapas (Logan, 2000):

1 – Faça-se a *Transformada de Fourier* duma janela de tempo definido, do sinal de áudio.

2 – Mapeiem-se as amplitudes logarítmicas do espectro obtido a partir do sinal áudio, na escala de *mel,* usando janelas triangulares sobrepostas.

3 – Faça-se a *Transformada Discreta de Cosseno* da lista de amplitudes logarítmicas da escala de *mel*, como se estas fossem um sinal.

4 – As amplitudes resultantes do espectro deste sinal são os MFCCs.

No fundo, os MFCC são uma descrição da evolução do espectro de potência visto através de vários filtros cada vez mais finos. Normalmente resultam bastante em vários sistemas de classificação áudio, mas não é claro como se correlacionam com fenómenos do foro da perceção e da cognição.

- *Spectral Centroid*. É o ponto de balanço da distribuição de potência de um espectro, medindo o centro de massa desse espectro (Peeters, 2004). Embora descreva em traços largos se um som é predominantemente grave ou agudo, sofre de potenciais problemas e inconsistências não devendo ser interpretado diretamente como um reflexo de mecanismos preceptivos.

- *Spectral Flux*, mede a alteração do valor do espectro entre dois *frames* sequenciais (Peeters, 2004). É no fundo uma análise da quantidade de variação que está a acontecer num dado momento, discriminando e adicionando a variação que está a acontecer em cada zona de

frequências. Tem tão boa relação com a perceção humana de mudança que costuma ser utilizado como base em sistemas de segmentação (separação temporal de sons em componentes discretas coerentes).

- *Spectral Roll-off* é uma medida que determina um ponto na escala de frequências que marca os 85% da potência do espectro do som. Mede a obliquidade da forma do espectro. Distingue com grande facilidade a diferença entre um som com voz e um som sem voz, devido ao facto de som com voz ter uma potência maior nas frequências altas ao contrário do sem voz que se encontra nas frequências baixas (Peeters, 2004).

- *Zero Crossing Rate* é uma medida acústica que é definida pelo número de vezes que o sinal áudio atravessa o eixo de 0 dentro de um *frame* (Peeters, 2004).

- *Chroma* é um conjunto de atributos ligado à tonalidade onde este partindo do uso das oitavas cria um vetor de doze elementos representando os doze semitons de uma oitava seja ela qual for (Peeters, 2004). Acreditamos que para exemplos que não são necessariamente musicais, este descritor poderá ser irrelevante, mas como ele é tão utilizado, resolvemos incluí-lo para verificar a sua importância.

2.3 Algoritmos de Aprendizagem Máquina

No presente projeto, aplicámos vários algoritmos, de forma a permitir uma comparação. Partimos do uso dos cinco algoritmos mais usuais e que ficaram no "*Top 10 de melhores algoritmos de data mining*" (Wu et al., 2008). Note-se que tanto o SMO como o LibSVM são implementações alternativas de *Support Vector Machines* (SVM). Seguimos com uma caracterização dos algoritmos utilizados:

- **C4.5** Algoritmos de árvores de decisão estão entre os mais conhecidos e usados em todos os métodos de aprendizagem máquina (Quinlan, 2014). O C4.5 descende de outro algoritmo de árvores de decisão também criado por *Quinlan* que é o ID3. Este algoritmo constrói árvores de decisão a partir de um *dataset* que lhe é oferecido. Vamos supor um *dataset* $\boldsymbol{x = x_1,x_2,\ldots}$ de exemplos já classificados. Cada exemplo $\boldsymbol{x_i}$ constitui um vetor *p*-dimensional $\boldsymbol{(y_{1,i}, y_{2,i}, \ldots, y_{p,i})}$ onde o $\boldsymbol{y_j}$ representa valores de atributos ou caraterísticas dos exemplos, assim como a categoria ou classe a qual $\boldsymbol{x_i}$ pertence (Quinlan, 2014). Em cada nó da árvore, o algoritmo C4.5 escolhe o atributo dos exemplos que particiona o seu conjunto de amostras em subconjuntos com mais eficácia, tendendo a uma categoria ou outra. O critério deste particionamento é o ganho de informação normalizado. Assim sendo, o atributo com maior

ganho de informação é escolhido para tomar a decisão. O C4.5 parte depois disto para a repetição desta etapa anterior nas partições menores. Este veio trazer melhoramentos sobre o seu antecessor ID3, tais como:

- Consegue lidar com atributos contínuos e discretos, onde C4.5 cria um valor limiar e então particiona o conjunto de dados em dois subconjuntos, dos quais, um contém valores de atributos maiores do que aquele valor limiar, e, outro conjunto contém valores menores ou igual aquele valor limiar.
- Consegue lidar com *datasets* de treino com atributos incompletos, permitindo que atributos sejam rotulados como *?* para casos onde não existam valores. Quando os mesmos são rotulados como *?*, não são usados para os cálculos de entropia ou ganho de informação.
- Dá uma "poda" final a árvore que criou. Este retrocede pela árvore quando esta é criada e tenta remover ramificações que não são necessárias para o processo de decisão e substitui estas por nós "folha".

Este é implementado sob o nome de J48, como código aberto em Java no WEKA.

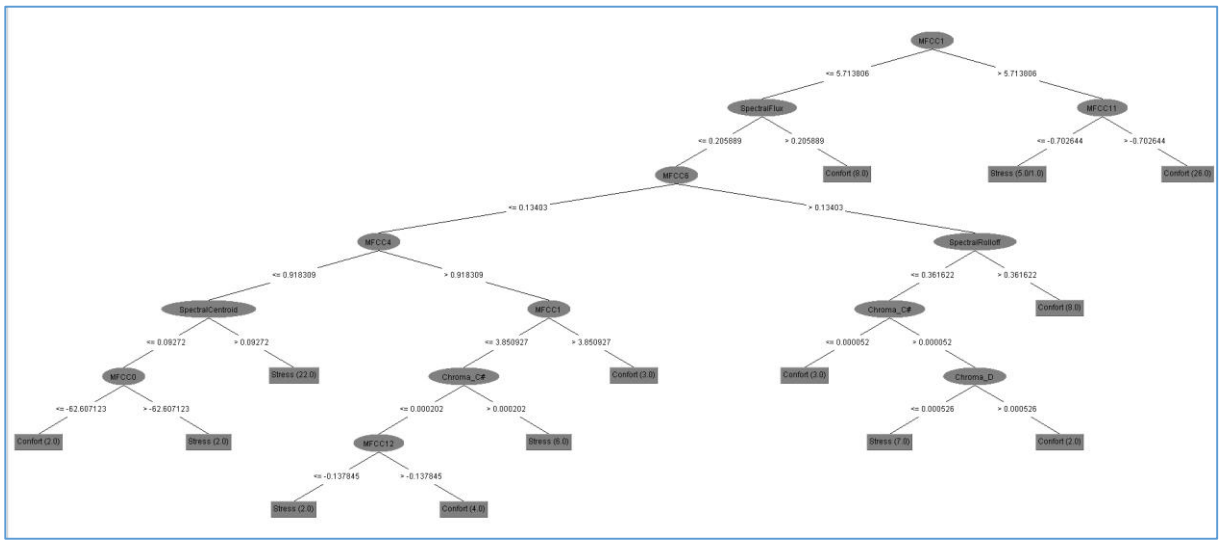


Figura 2.2 - Exemplo Gráfico da Árvore de Classificação do algoritmo J48 no WEKA

- **Naive *Bayes.*** Face a um *dataset* previamente classificado, em que cada exemplo pertence a uma classe e tem um vetor conhecido de atributos, tenta-se construir uma regra que permita a caracterização de futuras classes, partindo do uso único dos vetores de atributos para descrever futuros objetos (Wu et al., 2008). Este algoritmo é um bom ponto de partida por ser fácil de construir sem precisar de esquemas de parâmetros de estimativa complicada. Isto permite que o mesmo seja facilmente aplicado a *datasets* de grandes proporções. É tão fácil de interpretar que até utilizadores inexperientes em algoritmos de classificação conseguem perceber o porque de este fazer a classificação que faz. Finalmente, costuma obter bons resultados de classificação, não sendo o melhor para certas aplicações particulares, continua a ser de confiança e robusto para a maioria dos casos.

Vamos assumir um *dataset* $\boldsymbol{u}$ com duas classes $\boldsymbol{q}$ e $\boldsymbol{w}$, $\boldsymbol{u} = \boldsymbol{q}, \boldsymbol{w}$. Aqui vamos ter como objetivo o uso dos exemplos iniciais do *dataset* como membros de treino, para o algoritmo construir resultados superiores que vão ser associados a classe $\boldsymbol{q}$ e resultados inferiores que estarão ligados a classe $\boldsymbol{w}$**.** A classificação pode ser assim obtida a partir da comparação de estes resultados com um limite $\boldsymbol{t}$ definido. Se definirmos $\boldsymbol{P}(\boldsymbol{u} \mid \boldsymbol{x})$ para ser a nossa probabilidade condicional de um objeto $\boldsymbol{x}$ com um vetor de atributos $\boldsymbol{x} = (x_w, \ldots, x_p)$ pertencer a uma classe de $\boldsymbol{u}$, então, qualquer função monótona de $\boldsymbol{P}(\boldsymbol{u} \mid \boldsymbol{x})$, daria um resultado aceitável. Em particular a proporção de $\boldsymbol{P}(\boldsymbol{q} \mid \boldsymbol{x}) / \boldsymbol{P}(\boldsymbol{w} \mid \boldsymbol{x})$ seria também aceitável. A teoria elementar da probabilidade permite-nos decompor $\boldsymbol{P}(\boldsymbol{u} \mid \boldsymbol{x})$ como proporção a $\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{u})\,\boldsymbol{P}(\boldsymbol{u})$, onde $\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{u})$ é a distribuição condicional de $\boldsymbol{x}$ por exemplos do *dataset* $\boldsymbol{u}$, e, $\boldsymbol{P}(\boldsymbol{u})$ é a probabilidade de um exemplo pertencer a uma classe de $\boldsymbol{u}$ (Wu et al., 2008).

Isto significa que a proporção de $\boldsymbol{P}(\boldsymbol{q} \mid \boldsymbol{x}) / \boldsymbol{P}(\boldsymbol{w} \mid \boldsymbol{x})$ vai passar a ser:

$$\frac{\boldsymbol{P}(\boldsymbol{w} \mid \boldsymbol{x})}{\boldsymbol{P}(\boldsymbol{q} \mid \boldsymbol{x})} = \frac{\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{w})\,\boldsymbol{P}(\boldsymbol{w})}{\boldsymbol{f}(\boldsymbol{x} \mid \boldsymbol{q})\,\boldsymbol{P}(\boldsymbol{q})}$$

- **LibSVM** tem vindo a ser desenvolvido desde o ano 2000 por *Chang & Lin*, (2011) partindo da base do SVM, o mesmo tem sido usado mundialmente em diversas áreas, tendo sido baixado do *Source-Forge*[1] mais de 250,000 de vezes. Este divide-se em três opções:

[1] Source-Forge é um domínio de partilha de informação, *softwares*, algoritmos e outras coisas entre os investigadores de todo o mundo.

- SVC (*Suport Vector Classification*), que tem como função de decisão:

$$\text{sgn}\,(\boldsymbol{w}^T\,\emptyset(\boldsymbol{x}) + b) = \text{sgn}\left(\sum_{i=1}^{l} y_i\,\alpha_i K(x_i, x) + b\right)$$

Onde o método recebe $x_i \in R^n$, $i = 1,\ldots, l$, em duas classes e um indicador de vetor $y \in R^l$ tal que $y_i \in \{1,-1\}$. Nesta função o método partindo de $\emptyset$, mapeia um vetor de treino x num grande espaço dimensional, o algoritmo retira e armazena as variáveis y_i, α_i, e b de nomes de vetores de suporte e outros parâmetros como K (*kernel*) para construir o modelo de treino.

- *One-class* SVM, que tem como função de decisão:

$$\text{sgn}\left(\sum_{i=1}^{l} \alpha_i\,K(x_i, x) - p\right)$$

Onde o método recebe $x_i \in R^n$, $i = 1,\ldots, l$ vetores de treino sem qualquer informação sobre a classe que tem. Retirando também as variáveis de nomes de vetor e *kernel* para definir o modelo de teste sem classes.

- SVR (*Suport Vector Regression*), consideremos dar um grupo de pontos de treino, $\{(x_1,z_1),\ldots,(x_l,z_l)\}$, onde $x_i \in R^n$ é um vetor de atributos e $z_i \in R^n$ é o alvo do resultado. Este vai nos dar um função aproximada de:

$$\sum_{i=1}^{l}(-\alpha_i + \alpha_i^*)\,K(x_i, x) + b$$

Desta função vai retirar variáveis e o valor de *kernel*, mais as classes para criar o modelo de teste.

A utilização típica do LibSVM envolve dois passos: primeiro, a criação de um modelo a partir do treino de um *dataset,* e segundo, o teste desse modelo para prever informação noutro

dataset. No caso de SVC e SVR, o algoritmo consegue ainda estimar probabilidades de resultados.

- *Sequential Minimal Optimization* (**SMO)** é um algoritmo baseado em SVM. Veio resolver o problema da programação quadrática (PQ)[2] que existe nos algoritmos SVM. Faz isto sem precisar de matrizes extra para armazenamento, nem passos numéricos de otimização quadrática (Platt, 1998). Decompõe o problema PQ em pequenos problemas PQ, a partir do teorema de *Osuna* para assumir a sua convergência. Ao contrário dos outros SVMs, o SMO tem como preferência resolver os mais pequenos problemas de otimização em todos os passos. Para o usual problema PQ nas SVM é necessário o uso de dois multiplicadores de *Lagrange* para se conseguir otimizar o problema da forma mais reduzida. Isto porque, os multiplicadores de *Lagrange* obedecem a uma igualdade linear constrangida. A cada passo, o SMO escolhe dois multiplicadores para otimizar juntamente, encontra os melhores valores para estes multiplicadores e melhora a base anterior dos algoritmos SVM.

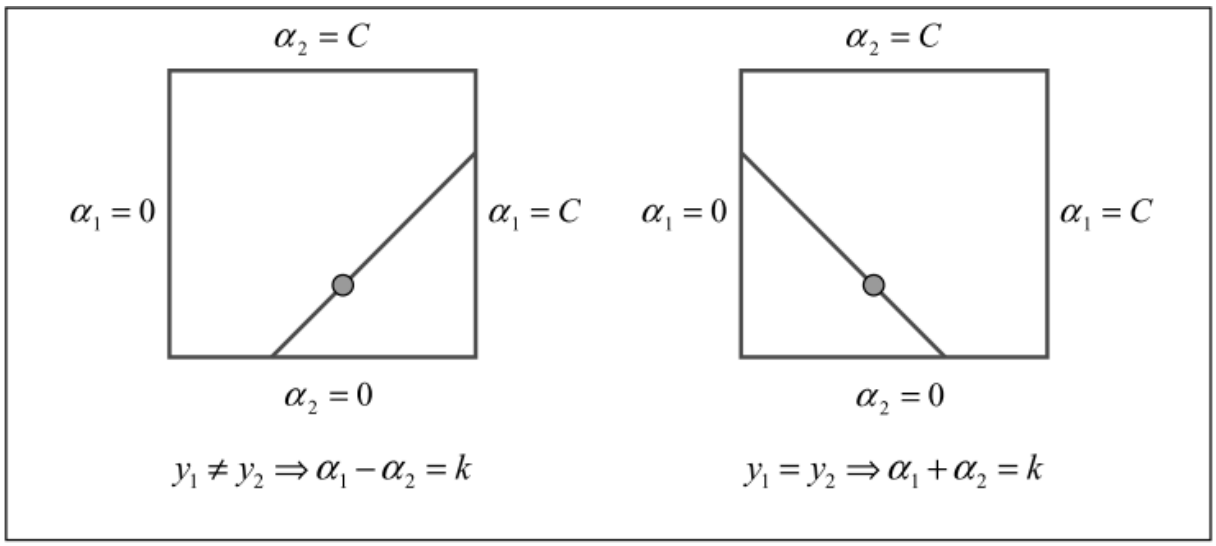


Figura 2.3 - Dois multiplicadores de Lagrange. Retirada de (Platt, 1998, p-6)

Com isto, o SMO deve ser usado, quando o utilizador não tem acesso a um pacote de otimização do problema PQ, ou quando não tem interesse no uso deste. Pode também ser usado

[2] Programação Quadrática é um problema matemático de otimização, onde o objetivo é otimizar a função quadrática de várias variáveis ao mesmo tempo.

para classificações lineares que é onde o algoritmo obtém melhores resultados. Por último, deve ser usado para a classificação de largos *datasets* devido aos seus resultados de escalamento comparados com outros algoritmos (Platt, 1998).

- **IBk** é uma variação da estratégia de k-Nearest Neighbours (kNN), uma abordagem relativamente simples à classificação. Na fase de treino, o algoritmo encontra um grupo de *k* exemplos que estão perto da área do exemplo que este a ser analisado, e que vão ser considerados "vizinhos". Consequentemente, o algoritmo cria grupos de vizinhos por cada exemplo analisado. Há três pontos fundamentais para esta abordagem que podem ser observados resumidamente na Figura 2.4.

"Input: *D*, o conjunto de *k* objetos de treino, e o objeto de teste $z = (\mathbf{x'}, \mathbf{y'})$

Processamento:

Computar $d(x', x)$ a distância entre *z* e qualquer objeto $(\mathbf{x}, \mathbf{y}) \in D$.

Seleciona $D_z \subseteq D$, o conjunto de *k* que está mais perto dos objetos de treino *z*.

Output: $y' = argmax_v \sum_{(x_i, y_i) \in D_z} I(v = y_i)$" (Wu et al., 2008, p-23).

Existem vários problemas com a performance deste algoritmo, tais como: a escolha do grupo *k* de vizinhos - se for pequeno demais, então o resultado torna-se sensível a pontos de ruído, caso seja grande demais, o algoritmo poderá começar a assumir pontos de outros grupos de vizinhos de outras classes. A abordagem de combinar os nomes dos grupos dos vizinhos também se pode tornar num problema, sendo que o método mais fácil para o resolver é através do voto maioritário, mas mesmo esta solução pode-se tornar num problema caso os vizinhos variem muito em distâncias, e apenas os vizinhos próximos tenham voto na indicação da classe do exemplo. Para o resolver, é feita uma abordagem que mede e combina não só os grupos de vizinhos como também a sua distância (Wu et al., 2008).

2.4 Sistemas de Classificação Sonora

Com a explosão nesta última década, das vastas bibliotecas sonoras digitais e do seu fácil acesso, tem havido uma rápida expansão da investigação sobre sistemas automáticos de caracterização de sons e retiro de informação destes. Estes sistemas têm como objetivos, para além da caracterização automática, procurar e organizar dados musicais e também dados sonoros que não estejam diretamente ligados à música, e.g. sons que nos rodeiam no quotidiano (Piczak, 2015). Não podemos falar de sistemas automáticos de caracterização sonora, sem falar da MIREX (*Music Information Research Evaluation eXchange*), uma estrutura com base na comunidade dedicada à avaliação de sistemas e algoritmos ligados á música (Downie, 2008), que inclui desde 2007, uma classificação de emoção do áudio na música (Downie et. al., 2007). Os melhores sistemas nesta categoria, tem vindo a desenvolver-se a cada ano que passa, usando unicamente descritores de áudio acústicos, (note-se que a maioria destes sistemas, foram desenhados para classificação de género musical e depois, adaptados para classificação de emoções). Com isto, tentámos contribuir para este desenvolvimento com uma nova abordagem, concentrando-nos numa problemática que ainda não foi explorada, que é a, caracterização de sons confortáveis e stressantes através da aprendizagem máquina.

Um dos sistemas mais polémicos é criado em *Dhanaraj* et. al. (2005). Aqui os autores exploram a ideia de um sistema que consiga prever automaticamente "*Hit Songs*[3]". Já existiam sistemas que classificavam música por, artistas ou género (Li, Ogihara, & Li, 2003), mas os autores tentam contribuir com uma nova abordagem, uma classificação por popularidade. Para isto eles usam um *dataset* de 1700 músicas, das quais extraem descritores acústicos e descritores vocais. Depois classificam-nos usando o algoritmo de máquina de vetores de suporte e "*boosting classifiers*" que são a junção de todos os algoritmos mais fracos restantes (Dhanaraj et al., 2005). Com isto, obtêm resultados positivos, o que faz os autores concluir que a ideia de que um sistema consiga prever automaticamente se uma certa música conseguirá estar no "Top 10"(Dhanaraj et. al., 2005) não é totalmente impraticável. Mas três anos depois, dois autores vêm contrariar esta ideia - "nós mostramos que não é possível a classificação da popularidade de uma música através de técnicas anteriores de aprendizagem máquina utilizando apenas dois grupos de descritores áudio razoáveis" (Pachet & Roy, 2008). Demonstram-no isto aplicando

[3] - Estas são música que estiveram no topo de uma lista de 10 melhores músicas.

as técnicas anteriores a um *dataset* de 32,000 músicas, e obtendo resultados negativos. Aplicam ainda as melhores técnicas que têm ao seu dispor e com as quais também não conseguem obter resultados positivos. Os autores não querem com isto demonstrar que é impossível classificar a popularidade de uma música partindo da análise de sinais sonoros ou da extração de descritores áudio, mas sim, que as técnicas disponíveis não lhes permitiam fazer tais (Pachet & Roy, 2008). O mesmo continua a ser explorado e em *Ni* et. al. (2011) onde os autores fazem uma nova abordagem sobre o trabalho de *Pachet* e *Roy* (2008) do qual obtém resultados positivos, mas não afirmando que com isto já conseguiriam criar o sistema que classificasse as músicas automaticamente como "*Hit Songs*" antes de estas serem lançadas para o mercado. Os mesmos autores anunciam que irão investigar o tema no futuro, com novas abordagens (Ni & Santos-Rodriguez, 2011). Embora em alguns aspetos a procura de classificação por popularidade possa não estar relacionada com a nossa pesquisa (sinais muito mais longos e complexos, aferições manuais objetivas), consideramos que a procura de padrões que expliquem um fenómeno tão absolutamente subjetivo em termos de categorias é extremamente útil para informar a nossa metodologia.

Deixando a indústria da música um pouco de parte, e partindo para a caracterização de sons que não pertençam a esta, temos em *Hantke* et.al. (2012), um sistema de reconhecimento automático de emoções, provocadas por sons do quotidiano através de aprendizagem máquina, onde eles caracterizam esta emoção em duas classes, "*Valence"* e "*Arousal*[4]". Eles fazem isto extraindo descritores áudio de baixo nível como os *Mel Frequency Cepstral Coefficients* e aplicam-nos a um algoritmo de máquina de vetores de suporte.

Temos de nos mencionar também estudos mais recentes, como *Fan* et. al. (2015), onde os autores exploram um sistema de reconhecimento automático de acontecimentos e prazeres obtidos de paisagens sonoras. Partindo também do uso das classes "*Valence"* e *"Arousal*", os autores pedem a 20 participantes para classificarem o seu *dataset* entre as duas classes. A inovação neste caso é a criação de um modelo de avaliação de ouro. Treinam o sistema com os "dados de ouro", que são as respostas mas usuais entre os participantes. Concluem assim que este modelo lhes trouxe melhorias na proximidade da classificação pela aprendizagem máquina com a avaliação dos participantes.

[4] - *Valence* representa o prazer provocado por um estímulo e *Arousal* indica a intensidade de emoção provocada por um estímulo (Fan et al., 2015).

Dentro da diversidade que é o universo sonoro, podemos avançar como exemplo um sistema de aprendizagem proposto por Elaiyaraja & Meenakshi (2012), e que, partindo do uso de um algoritmo de suporte de vetores, mostrou uma margem de acerto de 91.7%, na classificação de sons em várias classes de produção, tais como filme, desporto, publicidade, música, entre outras. Esta margem de acerto é calculada a partir da média de vezes que o algoritmo classifica exemplos corretamente (sendo estes considerados positivos verdadeiros) e.g. num *dataset* de 100 exemplos, o algoritmo consegue classificar corretamente 70 exemplos e incorretamente 30. Com isto, este algoritmo vai ter uma margem de acerto de 70%. Também podemos observar exemplos na literatura que, usando o mesmo algoritmo, tenham margens de acerto na classificação automática mais baixas. Neste sentido Steidl (2009), caracterizando emoção em discurso de crianças encontra resultados de apenas 57.9%. Podemos encontrar uma discrepância notável de 33.8 pontos percentuais entre estes dois casos. Uma das razões principais para esta variabilidade é o tamanho do *dataset* usado, sendo que um maior *dataset* vai oferecer mais exemplos de regras e padrões sobre os quais o algoritmo pode aprender.

Outra das possíveis razões seria a diferença qualitativa e quantitativa de atributos retirados do *dataset* e oferecidos ao algoritmo, notando que atributos específicos disponibilizam melhores padrões e curvas de aprendizagem para o sistema (Elaiyaraja & Meenakshi, 2012).

2.5 *Softwares* utilizados ao longo da Investigação

Para a investigação foram experimentados e utilizados vários *softwares*, tais como, MATLAB e WEKA, para a parte de aprendizagem máquina. MARSYAS, *Cmake* e *Microsoft Visual Studio* (MVS) para a extração de descritores áudio.

O MATLAB foi o primeiro a ser explorado. Este é um *software* extremamente utilizado por engenheiros e cientistas de todo o mundo, para analisar e desenhar sistemas e produtos transformativos (Mathworks, 2014). O MATLAB é usado para diversas tarefas como, aprendizagem máquina, processamento de sinal, processamento de imagem, visão computacional, comunicação, desenho de controlos, robótica e muitas mais. Esta plataforma para além de oferecer otimizações para resolver questões de investigação científicas, dá-nos também um nível alto de linguagem de programação, gráficos para o apoio à visualização de resultados, ferramentas para a criação de aplicações com interfaces customizadas entre outras (Mathworks, 2014). No entanto é frequentemente apontada como desvantagem o ser pouco amigável para novos utilizadores com pouca experiência em programação. Podemos observar o ambiente do mesmo na figura seguinte.

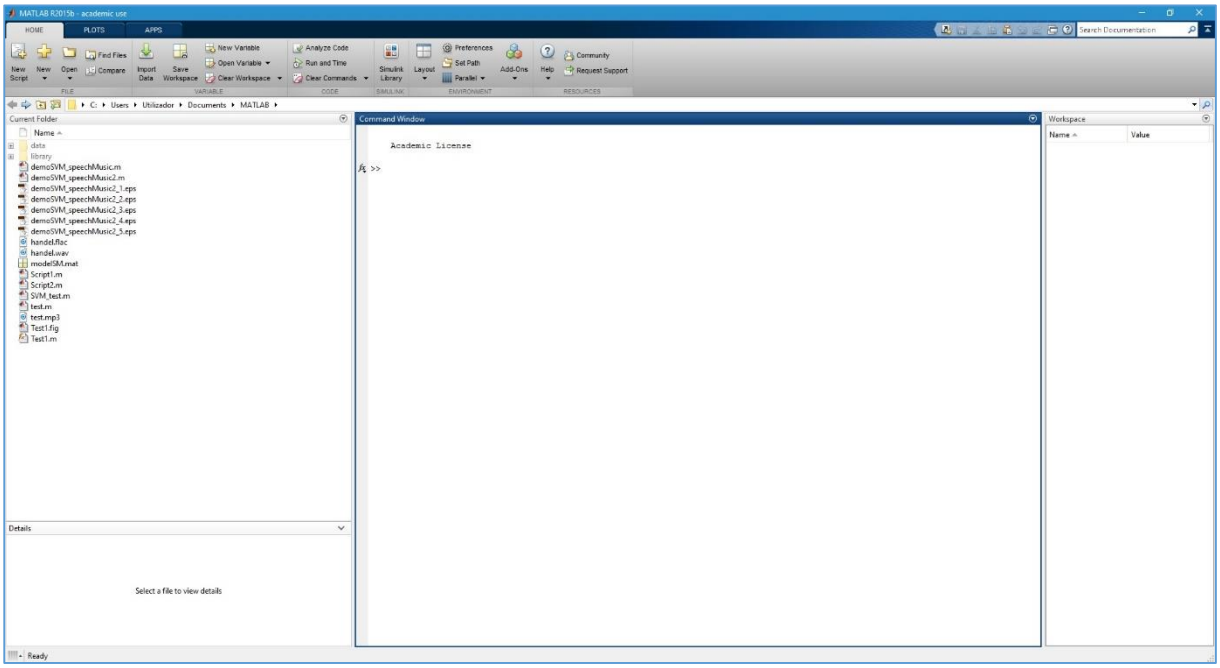


Figura 2.4 - Ambiente do Software MATLAB

O WEKA é um *software* que foi criado com a ideia de unificar todo o estado da arte feito em aprendizagem máquina, facilitando assim o seu acesso e uso a todos os investigadores. O programa foi baixado da internet mais de 1.4 milhões de vezes desde que foi colocado em Abril de 2000 (Hall et al., 2009) no *Source-Forge,* que é um domínio de partilha de informação, *softwares*, algoritmos e outras coisas entre os investigadores de todo o mundo. O *software* oferece aos seus utilizadores várias opções, desde classificar um *dataset*, a treinar um algoritmo de aprendizagem máquina, comparar algoritmos entre si para ver quais os mais adequados, entre outras. O WEKA tem variadas interfaces visuais, sendo a mais conhecida e principal o *Explorer*. Este tem os paneis mais importantes do WEKA, desde classificadores, visualização de resultados ou atributos, *clusters* entre outras.

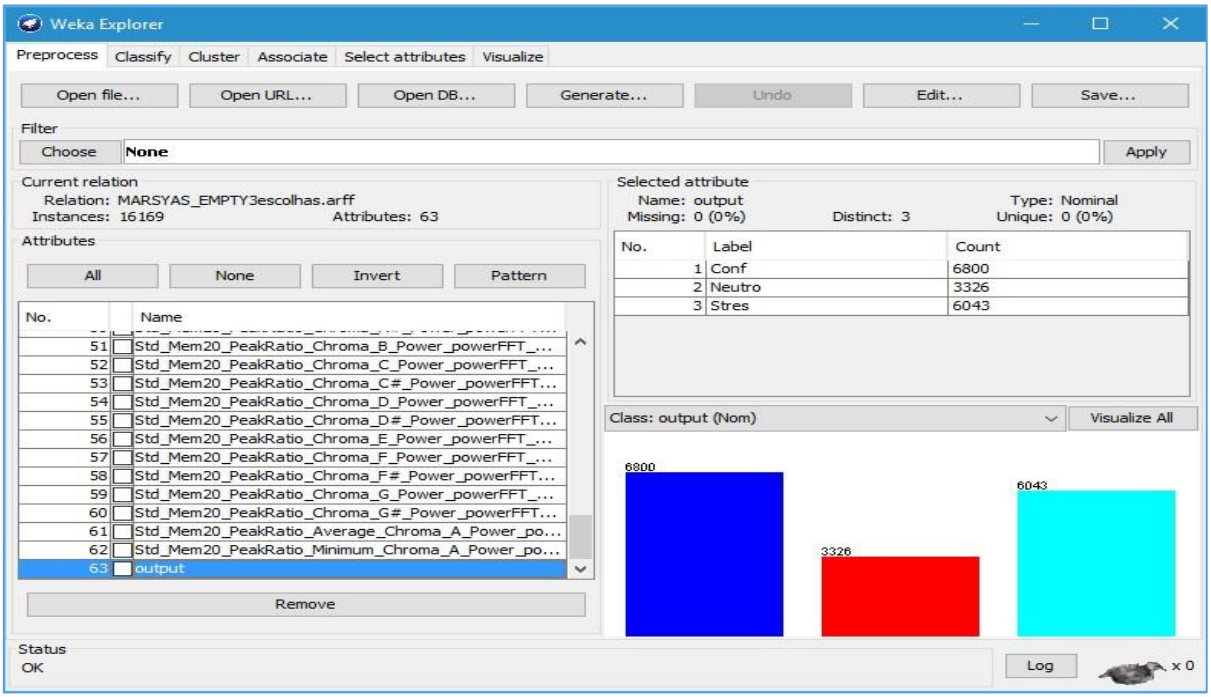


Figura 2.5 - Ambiente do WEKA Explorer

Este *software* proporciona também um interface visual, o "*KnowledgeFlow*". Este permite ao utilizador executar todas as funções disponíveis no *Explorer* de maneira mais visual e interativa (Hall et al., 2009). e. g. Enquanto no *Explorer* apenas podemos utilizar um classificador de cada vez, no *KnowledgeFlow* podemos utilizar vários ao mesmo tempo, como pode ser observado no esquema da Figura 2.7.

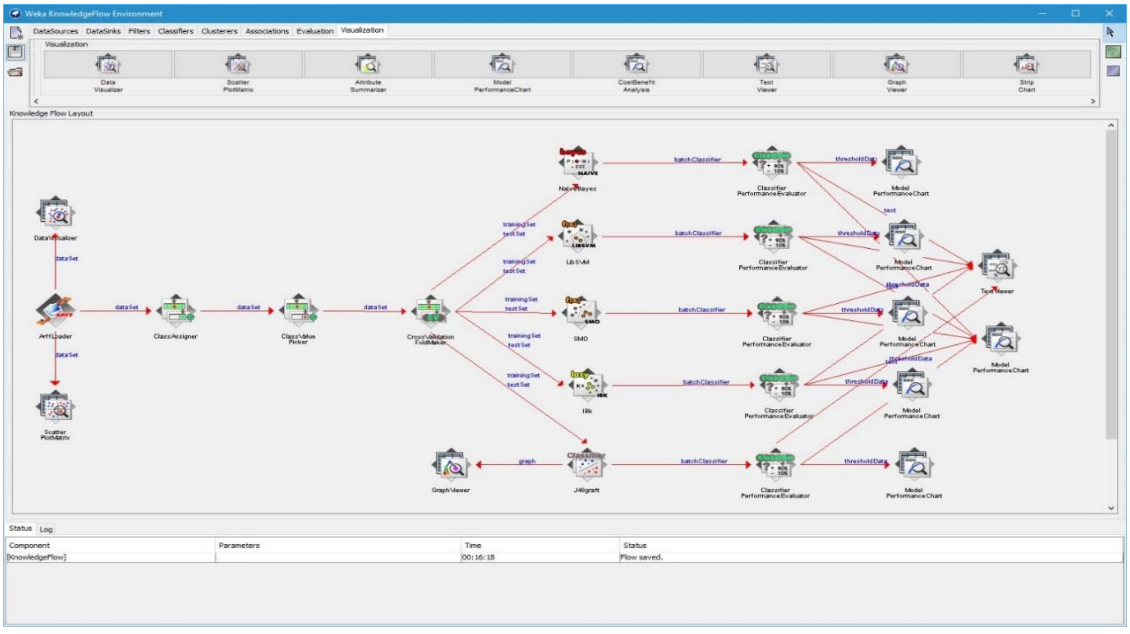


Figura 2.6 - Ambiente do WEKA KnowledgeFlow

Para a extração dos descritores áudio, vários *softwares* utilizados tiveram de ser integrados. Foi necessário usar o *Cmake* e o MVS para compilar o código fonte do MARSYAS. O *Cmake* é uma plataforma "aberta ao público", e tem como intuito construir e testar aplicações distribuídas em formato fonte, como o MARSYAS. Este último começa a ser desenhado em 1998 e foi evoluindo até ao ano 2000 onde George Tzanetakis, um investigador da Universidade de Vitória e interveniente muito activo na comunidade MIR, decide que o código deveria ser alterado e a utilização redesenhada (Percival & Tzanetakis, 2008). Após algum tempo de distribuição, o programa começou a perder a popularidade na comunidade científica. Com isto Tzanetakis, com a ajuda de Roger Dannenberg, começou a modificar a forma de como o *software* gerava as matrizes de vetores dos atributos e a implementar novos algoritmos que estavam a aparecer na comunidade (Percival & Tzanetakis, 2008). O MARSYAS não oferece uma interface gráfica, o que torna a experiência pouco apelativa, funcionando apenas através de linhas de código.

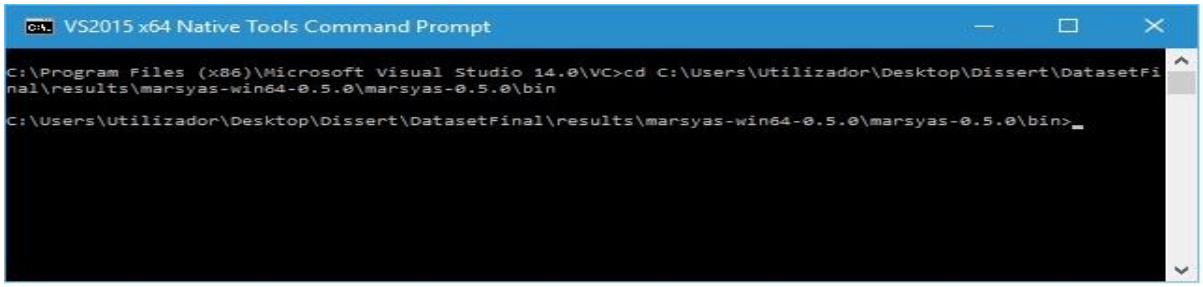


Figura 2.7 - Ambiente do MARSYAS

3 Desenvolvimento e Resultados.

Neste capítulo, vamos aprofundar no que consistiu a parte empírica desta dissertação e quais as fases ocorridas na mesma. Este estará dividido em três subcapítulos: Desenvolvimento que irá abordar toda a parte de criação do *dataset* e objeto Boxi; Análise de Resultados da Aprendizagem Máquina onde exploramos o funcionamento do nosso sistema partindo dos resultados obtidos; e por ultimo uma Conclusão de capítulo;

3.1 Desenvolvimento

O desenvolvimento prático deste trabalho foi previamente dividido em duas partes:

Pré-Produção, em que planeamos todo o desenvolvimento e construímos ou exploramos as ferramentas que permitem iniciarmos a **Produção**, onde aprofundamos a criação do *Dataset* e do objeto Boxi, utilizado para simplificar o processo de classificação manual. Ao longo deste processo colecionámos todos os dados necessários para alimentar o sistema de aprendizagem máquina.

3.1.1 Pré-Produção

Nesta fase foi desenvolvida toda a parte metodológica que iria ser usada nas etapas seguintes, e que nos veio chamar a atenção para, e obrigar a resolver, questões como:

- Quais os *softwares* necessários para a criação do nosso sistema de aprendizagem máquina? Há capacidades técnicas suficientes para a criação dos mesmos de raiz? Que opções pré-construídas existem, quão costumizáveis são, e como se comparam com a opção de ter um sistema autónomo e totalmente construído no âmbito do projeto?

- Quantos exemplos auditivos teriam de existir no *dataset* para este ser aceitável e aplicável ao mundo real *a posteriori*?

- Quais os melhores sons a utilizar, de forma a serem caracterizados como Confortáveis ou Stressantes? Qual o conceito subjacente à sua coleção? (e.g. áreas dos transportes públicos, área do cinema, área da medicina, etc.);

- Quantas pessoas seriam necessárias para classificar o *dataset* manualmente?

- Quais os descritores áudio de baixo nível adequados com vista à classificação pretendida? Como extraí-los? Valeria a pena reduzir o seu número?

Começámos a responder às questões iniciando a procura de soluções para construir o nosso *software* de aprendizagem máquina. Experimentámos o *MATLAB* (*Matrix Laboratory*) e algumas das suas bibliotecas, o que mais tarde na produção veio a criar alguns obstáculos, os quais irão ser expostos na parte de Produção. Verificamos a inexistência de um sistema fechado que conseguisse autonomamente responder a todas as necessidades do nosso projeto. Embora tenhamos entendido que vários trabalhos na área eram construídos totalmente sobre um sistema como MATLAB ou R, encontramos também vários casos de aproveitamento de sistemas independentes de forma integrada para chegar a uma análise completa, e optámos por esta última abordagem.

De seguida estipulámos que seriam no mínimo necessários 100 exemplos auditivos para a construção do *dataset*, para este ser aceitável *a posteriori* para análise de dados. Escolhemos este número porque, com um *dataset* inferior a 100 exemplos sonoros, não é oferecida informação suficiente para o algoritmo que vá caracterizar e aprender deste. Naturalmente que há que atingir um compromisso entre a dificuldade de recolha de sons com qualidade, diversidade e âmbito apropriados, e os requisitos mínimos para que um sistema possa ser fidedigno. Como quisemos apertar com os critérios de qualidade e âmbito, tivemos dificuldade em ir para além dos 100 exemplos.

Decidimos que estes deveriam estar inseridos em três áreas: sons de interface público em sistemas de transporte coletivo; sons de máquinas de suporte à saúde em contexto médico; e *sonic logos*. O ser humano é constantemente afetado por este tipo de estímulos, mesmo sem consciência direta, o que considerámos justificar a sua inclusão. Estes exemplos seriam na nossa perspetiva ideais para serem sujeitos a caracterização como Confortáveis ou Stressantes.

Logo depois, concluímos por análise de literatura em casos comparáveis que este *dataset* deveria ser classificado manualmente auscultando a opinião de 30 pessoas de idade adulta.

Após estarmos convencidos, através de análise de literatura e métodos heurísticos, que tínhamos as respostas apropriadas às questões que nos pusemos, e dada a impossibilidade de confirmar a sua validade sem pôr as respostas especulativas em prática, foi iniciado o processo de produção.

3.1.2 Produção

Ao passar para a etapa de produção, deparámo-nos com o primeiro na criação do *software* que corresse os algoritmos de aprendizagem máquina. A programação de raiz envolveria conhecimento avançado de programação e de matemática, algo que implicaria uma curva de aprendizagem e investimento temporal excessivo para o âmbito do mestrado. Encontrámos a solução de usar um *software* existente, já anteriormente enunciado neste documento, que é o WEKA. Este é bastante conhecido e usado no meio científico, o que nos garantiu um grau elevado de confiança na sua utilização, de forma a alcançar os objetivos traçados. Esta estratégia é muito frequentemente utilizada em trabalhos da área.

O sistema WEKA, no entanto, espera que o utilizador o alimente com um *dataset* já previamente populado com os resultados da classificação manual e com os descritores de baixo nível. Há uma grande variedade de descritores e de sistemas já preparados para a sua extração e, como previamente referido, há que escolher o conjunto mais apropriado, por vezes utilizando várias fontes para criar uma listagem exaustiva. Encontramos várias opções de código disponível para a extração destes, tais como YAAFE (*Yet Another Audio Feature Extractor*), MARSYAS, *Sonic Annotator/Vamp Plugins*, entre outros. Optámos por usar o MARSYAS que era o que oferecia maior compatibilidade com o WEKA.

Ultrapassados estes dois obstáculos, deu-se início à criação do *dataset*. Começámos por colecionar sons dentro das três áreas escolhidas na pré-produção, mas à medida que apareciam mais exemplos, tornava-se percetível que iria ser quase impossível recolher todos os 100 sons com qualidade máxima e direitos de uso a tempo. Conseguiu-se ainda agrupar um primeiro *dataset* com 88 sons, mas alguns deles de fontes pouco fidedignas, o que veio a obrigar a sua exclusão automática, reduzindo para 40 os sons utilizáveis. Com isto, veio a ideia de alterar a área e passar para uma área de sons icónicos que afetam constantemente o ser humano em contextos modernos. Selecionamos a área dos telemóveis usando desde sons de chamadas, alarmes, mensagens, alertas e sons de sistema. Foram então escolhidas as marcas alvo: *Apple*, *Microsoft*, *Oneplus*, *LG e Samsung*. A utilização destes sons trouxe também a vantagem de que todos os exemplos já tinha sido previamente aprovados por grandes marcas.

Numa primeira escolha terminámos com um total de 615 exemplos sonoros, No entanto, estava fora de questão utilizar todos estes exemplos, e assim começámos a restringir o critério de elegibilidade, escolhendo apenas:

- sons com pequenas melodias, com um máximo de 8 notas diferentes,
- sons repetitivos como os alarmes habituais,
- sons sem voz humana ou animal,
- sons que não envolvessem grandes acordes musicais nem muitos instrumentos diferentes,
- sons com um máximo de 10 segundos, não contando aqueles que ultrapassavam os 10 segundos através de repetição literal, e.g. alarmes e chamadas.

Conseguimos com isto isolar 318 exemplos sonoros dos 615. Este aumento de número implicaria aumentar o número de colaboradores para a classificação manual do *dataset*. Isto veio a provar-se temporalmente e logisticamente impraticável, pelo que regressámos ao pano original de ter 100 exemplos auditivos. Com isto veio a questão do como extrair 100 sons do universo de 318, sem impor a nossa influência direta e subjetividade no processo de escolha. A primeira opção foi a criação de um gerador de números aleatório. Implementou-se um pequeno código em C++ no *software Microsoft Visual Studio* que em 318 números, gerou 100 números aleatórios.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
	int c, n;
	printf("100 random numbers in [1,318]\n");
	for (c = 1; c <= 100; c++) {
		n = rand() % 317 + 1;
		printf("%d\n", n);
	}
	return 0;
}
```

Inicialmente este código funcionou sem problemas para o fim desejado, até ao momento em que foram escolhidos os 100 exemplos, altura em que nos deparámos com o facto de alguns destes números aleatórios serem repetidos. Posto isto, foi necessário optar por outra abordagem, e partimos então para o uso do *Windows PowerShell* implementando apenas uma linha de

comando, o que automatizou e facilitou ainda mais o processo da escolha dos 100 sons aleatórios, como se pode observar na seguinte figura.

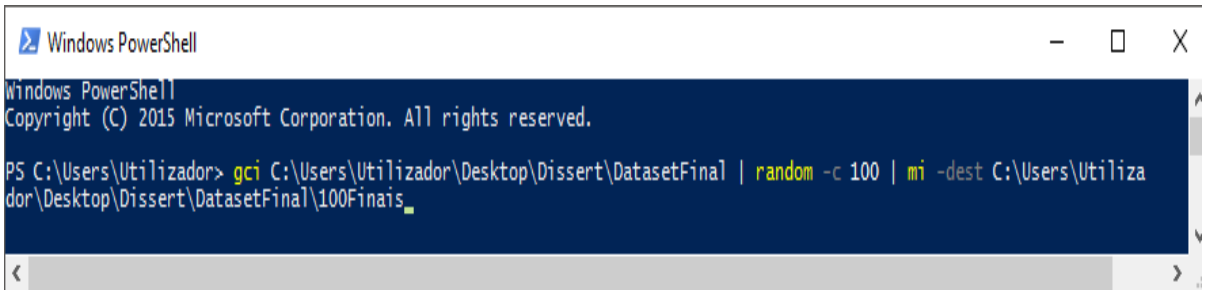


Figura 3.1 - Linha de Comandos Para escolha Aleatória

A linha de comandos da Figura 3.2, faz o seguinte: partindo do comando *gci* (*Get-ChildItem*) diz ao computador qual a pasta em que se tem de focar, depois usa o *random -c* para dizer a quantidade de objetos aleatórios que vai escolher e termina com o *mi* (*Move-Item*) *-dest* (*destination*) dizendo que é para aquela pasta específica, que vai mover esses 100 exemplos.

Já que estávamos a explorar as possibilidades de simplificação de tarefas do *Windows PowerShell*, aproveitámos para automatizar igualmente a parte de renomear todos estes exemplos para o nome de Dataset e o seu devido número entre 1 e 100 à frente, tanto por um princípio organizativo, como para a melhor integração nos *softwares* a utilizar de seguida, como MAX-MSP e *Arduino*, onde ter ficheiros com nome sequencial é bastante útil. Conseguimos esse objetivo, usando as seguintes linhas de comando:

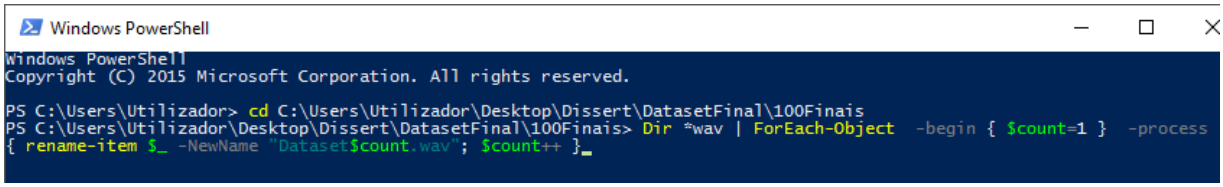


Figura 3.2 - Linha de Comandos Para Renomear todos os Ficheiros

Depois de termos os 100 sons isolados num só local, deu-se início ao processo de uniformização em termos de características gerais do ficheiro áudio, desde frequência de amostragem, taxa de quantização *(bit depth)* e em formato (.wav). Para tal, utilizámos o

software Adobe Audition utilizando o método de *batch processing*[5] dando-lhe as seguintes especificações: 24 bits; 44100 Hz e não foi alterado o número de canais áudio (Mono 1 Canal de áudio ao centro; Stereo 2 Canais de áudio, esquerda e direita) deixando os de origem de cada ficheiro. Foram necessárias fazer estas alterações porque muitos dos exemplos, sendo obtidos de marcas diferentes vinham com características diferentes. Estas especificações uniformizadas podem ser observadas na figura seguinte:

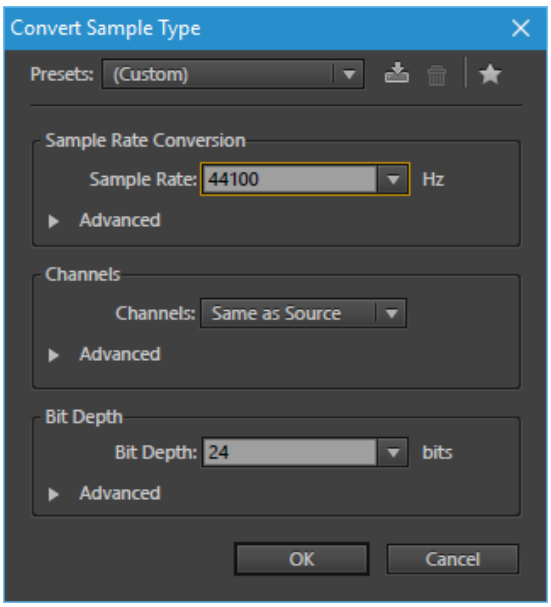


Figura 3.4 - Batch Processing das características de todos os sons

Terminada esta etapa na totalidade, só faltava uniformizar todos os sons em termos de volume aparente segundo a norma EBU R128 (*European Broadcasting Union Recommendation R128*). Esta é uma norma Europeia vinda do universo de *broadcast*, que vem interpretar o volume apercebido (*loudness*) aferindo através de um único número uma sensação subjetiva e difícil de caracterizar. Através da norma temos uma expectativa de que não existam discrepâncias de volume entre sinais. Aplicámos esta mesma norma no nosso *dataset*, para que a discrepância de volumes de uns exemplos para os outros não influenciem a escolha do utilizador. É sobejamente reportado que a utilização de sons com diferente *loudness* em testes subjetivos tem tendência a enviesar os mesmos (Bech & Zacharov, 2006). Com todos os outros

[5] - *Batch Processing* é a execução de vários programas ou funções num computador sem intervenção manual do utilizador.

fatores iguais, *loudness* superior tende a resultar na impressão subjectiva de que um dado som terá melhor qualidade.

Posteriormente, começámos a classificação manual do *dataset*. Nesta etapa questionámo-nos se não seria mais interessante torná-la numa experiência interativa para os utilizadores, criando um objeto físico pudesse ser utilizado para classificar o *dataset*. Com isto, partímos para a criação de um objeto ao qual foi dado o nome de "Boxi", o que veio automatizar todo o processo de recolha de informação do utilizador. Para tal, foi usada uma placa *Arduino Uno*, duas placas RFID (*Radio-Frequency IDentification*) e uma Placa de Ensaio onde tudo estaria ligado, depois o utilizador usaria uma etiqueta com ID (*IDentification*) que seria reconhecido pelas placas RFID. Este objeto teria de estar ligado ao computador para funcionar porque necessita de um *patch*[6] específico que foi criado em MAX-MSP e também um código específico escrito no *software Arduino* para a placa *Arduino Uno* reconhecer e receber dados das placas RFID e passar estes para o computador.

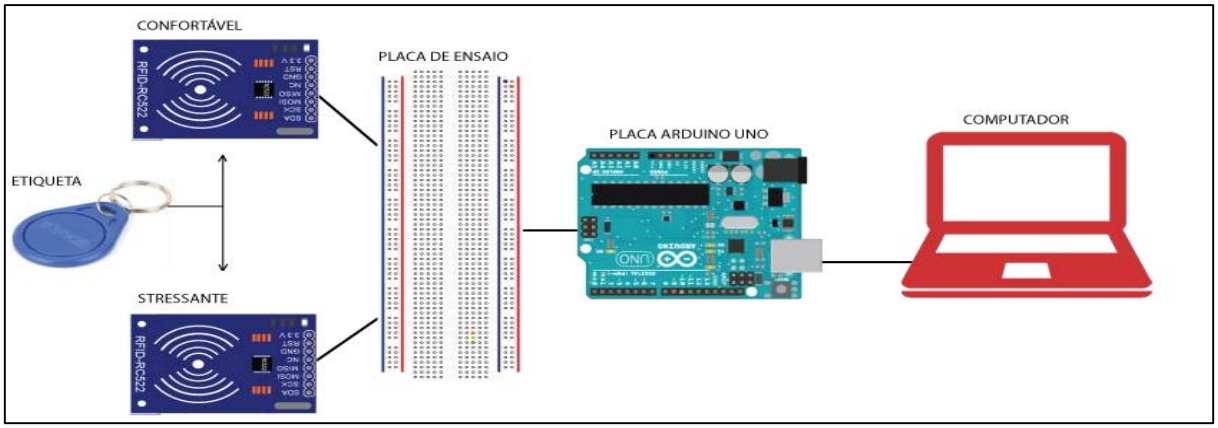


Figura 3.3 - Esquematização do funcionamento da Boxi

[6] - Pequeno programa criado no MAX-MSP para realizar funções específicas.

Com este esquema pensado, foi desenhado um *storyboard* para termos uma proximidade do objeto e do seu uso final, desenhando apenas três momentos, que achámos o suficiente para representar o uso da mesma, como pode ser observado de seguida.

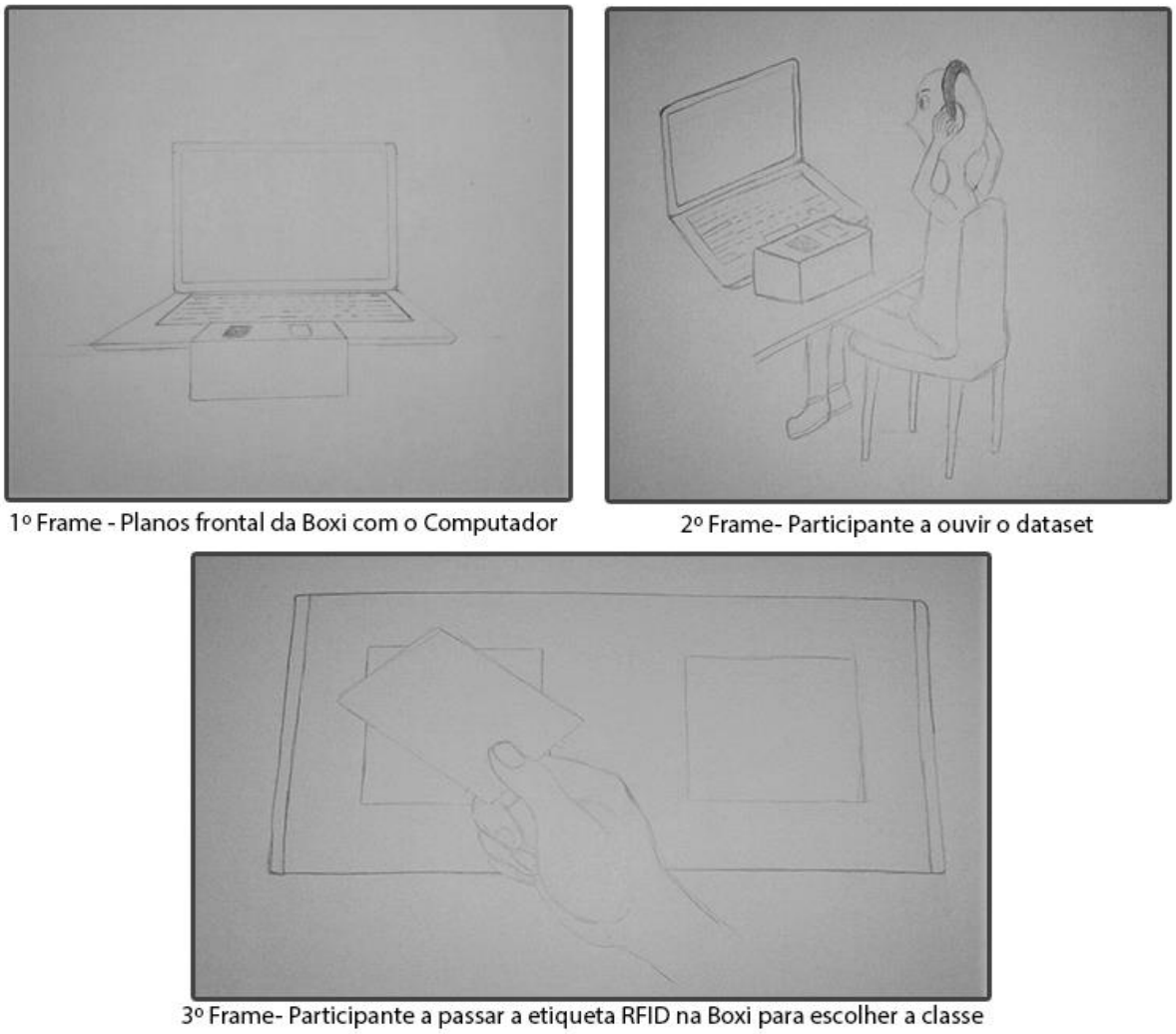


Figura 3.4 - Storyboard do objeto Boxi

Partindo do esquema e do *storyboard* pensados previamente para o funcionamento da Boxi, deu-se início a prototipagem desta. Sabíamos que iria ter a forma de uma caixa, e que o tamanho da mesma teria de ser o suficiente para que todos os componentes ficassem no seu interior.

De seguida foi construído o primeiro protótipo, sobre qual foram feitos os primeiros testes, utilizando apenas o *Arduino* e o seu monitor para verificar se conseguiríamos receber informações das placas RFID. Com o sistema base a funcionar, foi criada a Boxi, construída partindo de uma caixa, com todos os componentes no interior, como se pode observar nas seguintes figuras:

Figura 3.5 - Fotos do objeto Boxi na sua forma final

O próximo passo foi a escrita do código, permitindo a interpretação simultânea de ambas as placas de deteção RFID no mesmo *Arduino*. O código funciona da seguinte forma: recorrendo às bibliotecas necessárias para poder compilar (*SPI.h* e *MFRC522.h)*, define-se que pinos da placa *Arduino* vão ser usados pelas placas RFID. Após este passo, cria-se uma variável para cada placa RFID, que vai ser utilizada mais tarde para as identificar e distinguir. De seguida, reserva-se a porta série COM3 com um *baud rate* de 9600 e abre-se a mesma, dado que é a que vai permitir à placa *Arduino,* comunicar com o computador e posteriormente com o MAX-MSP. Depois cria-se um ciclo que está constantemente à procura da situação em que alguma das placas está a receber dados, e que se tal acontecer, então bloqueia a variável dessa mesma placa e envia uma mensagem ao computador, identificando qual das placas está a ser usada naquele momento.

```
#include <SPI.h>
#include <MFRC522.h>

#define RST_PIN          9
#define RST_PIN2         5
#define SS_PIN           10
#define SS_PIN2          4

MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance.
MFRC522 mfrc522_2(SS_PIN2, RST_PIN2);   // Create MFRC522 instance.
MFRC522::MIFARE_Key key;

void setup() {
  Serial.begin(9600); // Initialize serial communications with the PC
  SPI.begin();        // Init SPI bus
  mfrc522.PCD_Init(); // Init MFRC522 card
  mfrc522_2.PCD_Init(); // Init MFRC522 card
  for (byte i = 0; i < 6; i++) {
    key.keyByte[i] = 0xFF;
  }
}

void loop() {
if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
    delay(700);
  } else {
    Serial.print(F("rfid1 "));
    String uid = "";
    for (byte i = 0; i < mfrc522.uid.size; i++) {
        uid = uid + mfrc522.uid.uidByte[i];
    }
    Serial.println(uid);
  }
  if ( ! mfrc522_2.PICC_IsNewCardPresent() || ! mfrc522_2.PICC_ReadCardSerial() ) {
    delay(700);
  } else {
    Serial.print(F("rfid2 "));
    String uid = "";
    for (byte i = 0; i < mfrc522_2.uid.size; i++) {
        uid = uid + mfrc522_2.uid.uidByte[i];
    }
    Serial.println(uid);
  }
}
```

Com a comunicação entre placas de RFID, *Arduino* e computador a funcionar, demos início a criação do *patch* no *software* MAX-MSP. Começámos por reutilizar um *patch* criado num projeto anterior para fazer a ligação entre o *Arduino* e o MAX-MSP, este pequeno *patch* fica inserido no *patch* principal o que o torna num *subpatch*. Este serve de ponte utilizando novamente a porta série, que tinha sido previamente aberta para este fim, no código de *Arduino*, por onde passa a informação dos dados recebidos a partir das placas RFID, encaminhando estas através de um *output* para o *patch* principal.

O mesmo pode ser observado na figura seguinte:

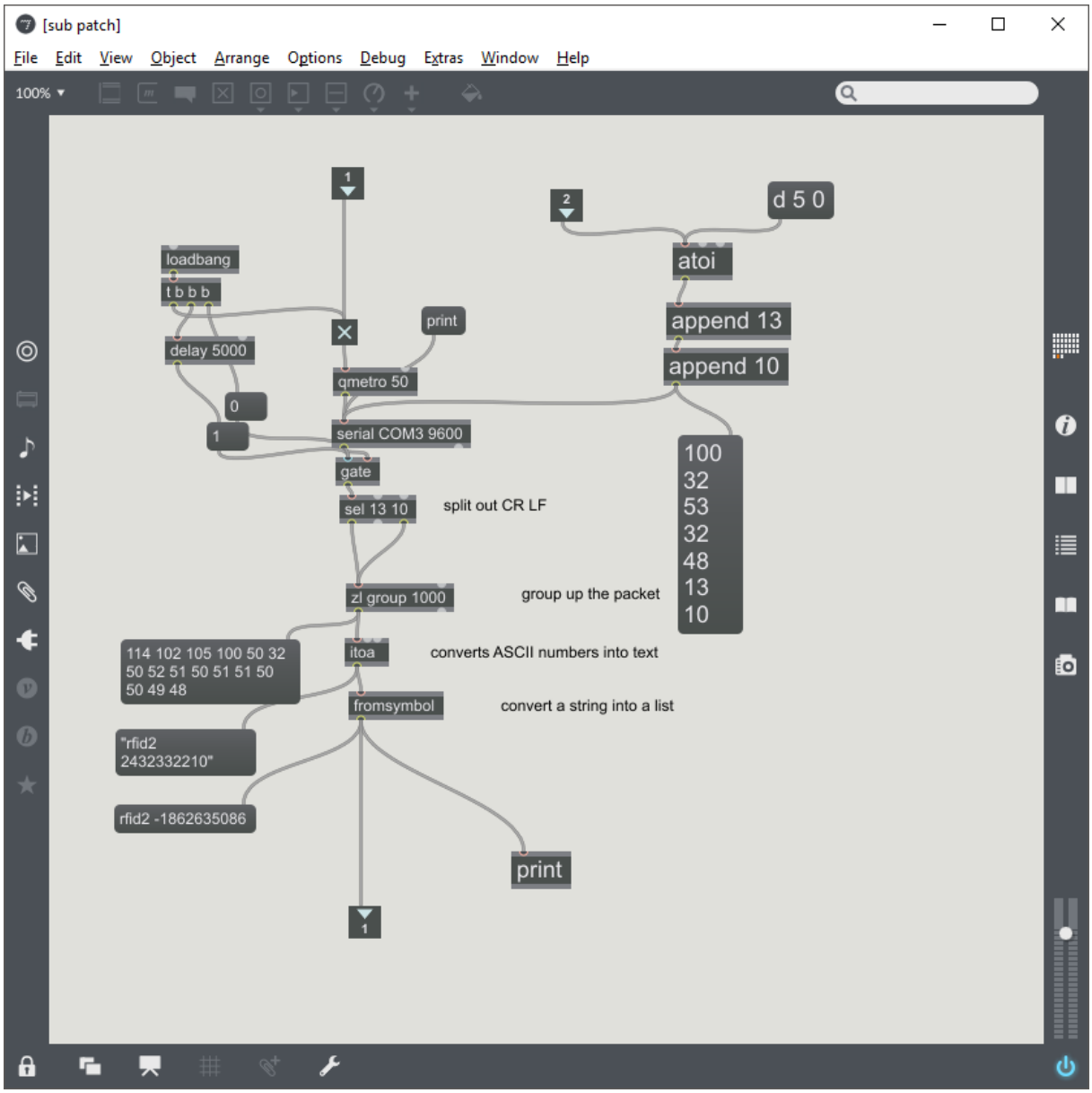


Figura 3.6 - SubPatch de Comunicação entre Arduino e MAX-MSP

De seguida, partimos para a construção do *patch* principal do funcionamento da Boxi. Para isso, precisávamos que este conseguisse, reproduzir o *dataset* posteriormente criado, mas como o mesmo contém 100 exemplos auditivos, foi necessário criar um contador para reproduzir os mesmos por ordem crescente. Para tal, utilizámos o objeto "*umenu*" que permitiu criar a lista dos 100 sons utilizando o objeto "*populate*" que partindo do uso do objeto "*opendialong fold*" obtém o caminho da pasta onde está situado o *dataset* e preenche o *umenu* com os ficheiros desta pasta. Após isto, o *patch* utiliza um contador para percorrer todos estes sons e um contador secundário ligado a um objeto "*coll*", que vai criar um bloco de notas onde automaticamente escreve as escolhas da classificação manual, utilizando *entradas* do contador de números, para saber que som o utilizador está a ouvir no momento e também as entradas *confort* para confortável e *stress* para stressante para naquele número especifico apontar, qual destas duas entradas o utilizador escolheu ficando com o resultado que se pode observar na figura seguinte.

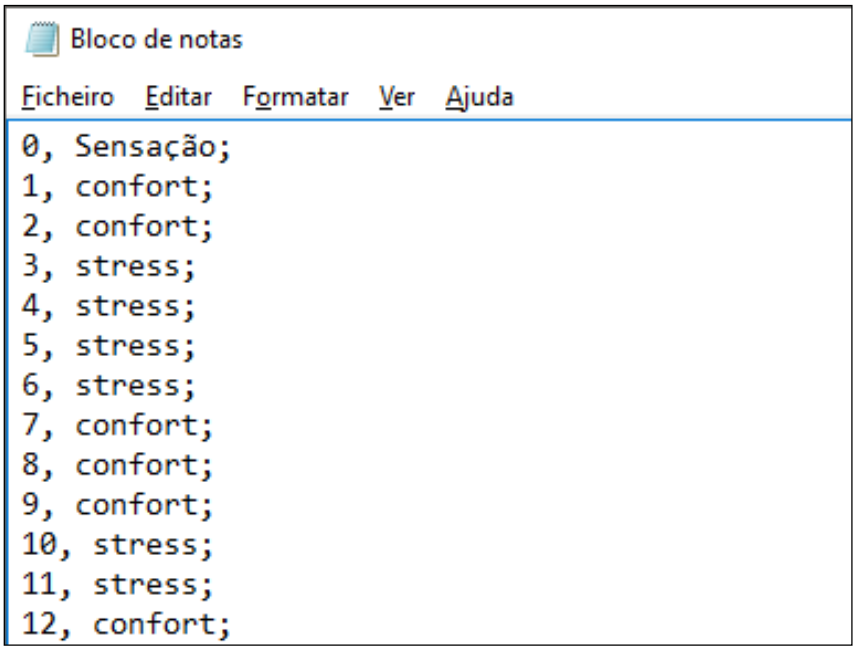


Figura 3.7 - Exemplo do resultado do objeto Coll

Caso todo este processo tivesse sido efetuado sem auxílio de um interface tangível, demoraria cerca de 15 a 20 minutos por utilizador a fazer a classificação. Utilizando este *patch,* em funcionamento com a Boxi, passou para uma média de 6 minutos e 34 segundos por utilizador.

Com o *patch* a funcionar, começámos a fazer testes com a Boxi para ver se não havia falhas antes de partir para os utilizadores e se era possível melhorar a performance ou, acrescentar novas funcionalidades ao *patch* que fossem uma mais-valia para a classificação. Logo no primeiro teste deparámo-nos com o facto de que era necessário colocar um *delay* (atraso) na captação de informação das placas RFID porque as mesmas estavam demasiado rápidas a enviar informação ao *patch*, o que fazia com que este saltasse exemplos auditivos à frente sem querer. De seguida, reparámos que era necessário um estímulo visual para avisar o utilizador que tinha mudado de exemplo, criando para esse efeito, um objeto *bang* e aumentando o tamanho o *umenu*, isolando-os no *patch* para estes se destacarem. Ligámos ainda este *bang* ao acionador do exemplo sonoro fazendo com que o *patch* tocasse automaticamente o som ao mudar de exemplo. Pode ser observado o *Patch* final na página seguinte.

Figura 3.8 - Patch Principal da Boxi

Dos 100 exemplos do *dataset* os participantes classificaram 43 deles como confortáveis e 31 como stressantes (em maioria), o que, sem ser forçado, revela uma divisão razoavelmente equitativa. Isto evita a preocupação com aspetos relacionados com *datasets* mal balanceados, que obrigaria a uma série de considerações adicionais. Deparámo-nos com o facto de 26 exemplos terem sido classificados como um empate técnico (15 participantes classificavam como confortável enquanto outros 15 classificavam como stressante), ficando sem classificação final. Para combater isto inicialmente, tomámos a decisão de escolher a classe, partindo do princípio de que um som para ser confortável teria de ter certos atributos tais, que nos pareceram consensuais ao analisar o que já tinha sido classificado, tais como:

- Reverberação e Atraso sonoro, estes criam a sensação do imaginário, desconhecido e de bem-estar;

- Timbre, graves e médios, mas principalmente graves que se tornam em sons calmos e relaxantes (e.g. vozes radiofónicas são calmas devido ao seu timbre grave);

- Harmonia, baseada em acordes maiores que tornam os sons mais felizes e simpáticos;

- Melodia, curta e memorável, sem saltos tonais bruscos.

Estando os sons confortáveis caracterizados, todos os que não se encontrassem dentro estas características, eram classificados como stressantes. Ficámos então com os resultados que podem ser observados no gráfico inferior. Partimos assim para a divisão do *dataset* nestes dois grupos colando os exemplos nas suas respetivas pastas. Ficando com um total de 57 exemplos confortáveis e 43 exemplos stressantes.

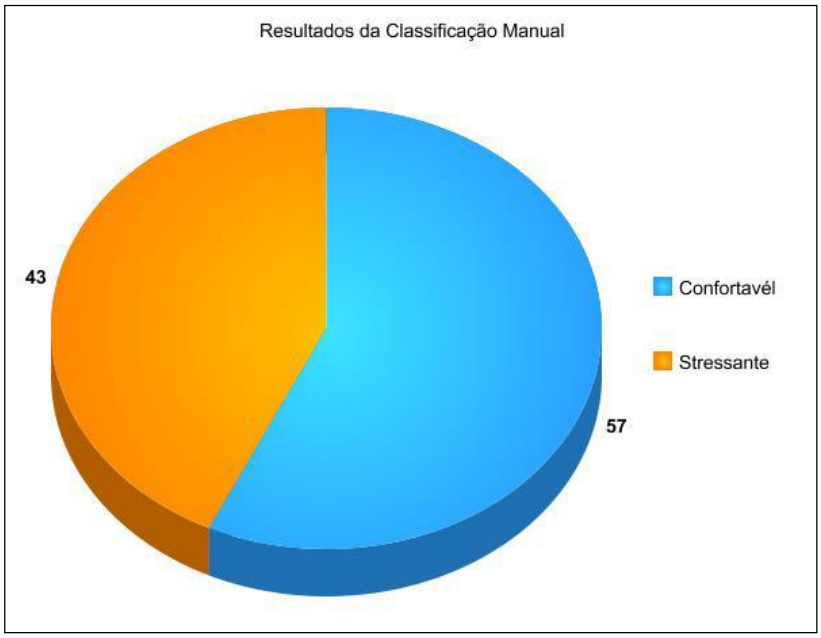


Figura 3.9 - Distribuição dos Resultados da Classificação Manual entre duas classes

A segunda abordagem para resolver o problema do empate dos exemplos foi a criação de uma terceira coleção neutra, que para além de nos resolver este problema, ainda nos proporcionou a possibilidade de comparar o funcionamento do nosso sistema na classificação do nosso *dataset* entre duas classes e três classes. Com isto, foi criada a classe "Neutro". Todos os exemplos sonoros previamente classificados pelos participantes que ficavam em empate (15/15) ou que não ultrapassavam a margem de dezoito-doze (18/12) eram considerados nesta classe, ficando assim o *dataset* dividido em 3 classes com 43 sons Confortáveis, 31 Stressantes e 26 Neutros. Reconhecemos que este sistema *post hoc* está longe de ser o ideal, e que forçamos os valores a considerar neutros de forma a construir grupos razoavelmente balanceados. No entanto, apresentamos apenas esta possibilidade como uma alternativa a um plano que mantivemos, para que haja acesso a um termo de comparação adicional durante a fase de avaliação.

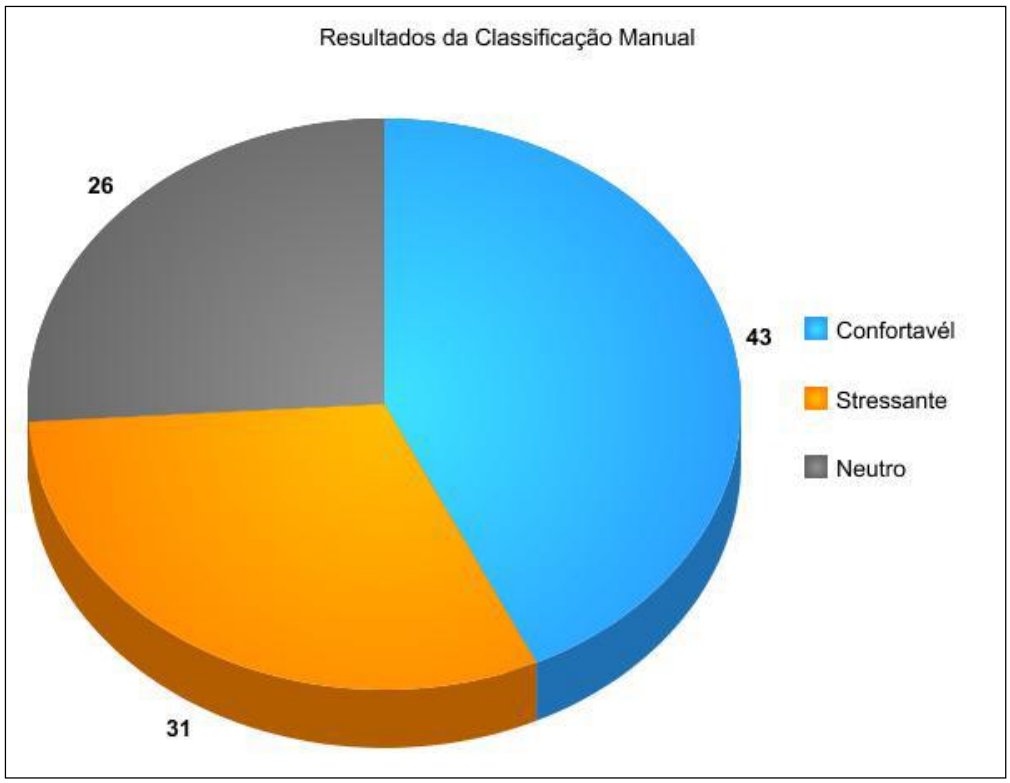


Figura 3.10 - Distribuição dos Resultados da Classificação Manual entre três classes

Com a parte de classificação manual terminada, iniciamos a etapa de extração dos descritores áudio de baixo nível. Para isso, escolhemos extrair os mais usuais, os MFCC, *Spectral* Centroid, Spectral *Flux*, *Spectral Roll-off*, ZCR e *Chroma*, dos quais já falamos

anteriormente na secção 2.2: Descritores Áudio de Baixo Nível. Para isto foi utilizado o MARSYAS e a sua operação "*bextract*". Primeiro foi necessário criar os ficheiros de teste. Esta etapa consistia na criação de dois documentos de texto, cada um correspondendo a uma classe, com a diretoria de todos os ficheiros de áudio para o MARSYAS saber onde os encontrar. Para isto usamos a operação "*mkcollection*":

```
mkcollection -c Confort.mf C:\Users\Utilizador\Desktop\Dissert\DatasetFinal\DatasetFinal\2Escolhas\Confort
-c Stress.mf C:\Users\Utilizador\Desktop\Dissert\DatasetFinal\DatasetFinal\2Escolhas\Stress
```

Com este código dissemos ao MARSYAS que teria de criar duas coleções utilizando o atributo "-c", uma delas teria de chamar-se "Confort.mf" e iria ser criada a partir da pasta dos exemplos confortáveis, e a outra, "Stress.mf" que utilizaria a pasta dos sons stressantes. Depois destas duas coleções criadas utilizámos o seguinte código:

```
bextract -fe  Confort.mf Stress.mf -w CS.arff
```

Com esta linha de comando, o *software* determinou que tinha apenas de extrair os descritores de áudio sem os classificar devido ao atributo "-fe". Sabe também que iria usar as bibliotecas previamente criadas e finalmente criar um ficheiro "CS.arff". Neste ficheiro foi gravar todos os descritores que extraiu, e o mesmo iria ser usado mais tarde no *software* WEKA para analisar e classificar os mesmos.

De seguida foi necessário criar novas coleções e extrair novos descritores de áudio para o *dataset* com a divisão em três classes utilizando os seguintes códigos:

```
mkcollection -c Conf.mf C:\Users\Utilizador\Desktop\Dissert\DatasetFinal\DatasetFinal\3Escolhas\Confort
-c Stres.mf C:\Users\Utilizador\Desktop\Dissert\DatasetFinal\DatasetFinal\3Escolhas\Stress
-c Neutro.mf C:\Users\Utilizador\Desktop\Dissert\DatasetFinal\DatasetFinal\3Escolhas\Neutro
```

A operação "*mkcollection*" novamente em uso com a adição de uma nova coleção ficando com as coleções "Conf.mf" para os Confortáveis, "Stres.mf" para os Stressantes e "Neutro.mf" para os Neutros.

bextract -fe Conf.mf Stres.mf Neutro.mf -w CSN.arff

Por fim executamos mais uma vez, a operação "*bextract*" que fez exatamente o mesmo que no exemplo anteriormente descrito, mas usando as três coleções, ao invés de só duas. Com toda esta etapa de produção do objeto finalizada e com os *datasets* prontos, procedemos para a análise dos resultados.

3.2 Análise e discussão dos Resultados da Aprendizagem Máquina

Com todas as etapas anteriores concluídas demos inicio à extração de dados finais dos algoritmos de aprendizagem máquina com o WEKA. Começamos por extrair a curva ROC e a sua área, através dos algoritmos na caracterização, isto no *dataset* de duas classes. A *Receiver Operating Characteristic*, ou curva de ROC, é um resultado gráfico que demonstra a performance de um classificador binário. É criada a partir dos valores positivos de acerto (*true positive rate*) no eixo dos yy, em comparação com os valores falsos de acerto (*false positive rate*) no eixo dos xx. Estes eixos variam entre os valores de 0 e 1, sendo que uma curva de ROC perfeita atingirá mais rapidamente o 1 no eixo dos yy. Usualmente uma curva de ROC consegue nos mostrar várias coisas como:

- Salienta a troca entre sensitividade que mede a porção de exemplos positivos que são classificados corretamente e especificidade que ao contrário da sensitividade mede os negativos que são classificados corretamente, sendo que o incremento da sensitividade é acompanhado pela diminuição da especificidade.

- Diz-nos também que quanto mais perto a curva estiver dos 90 graus, melhores serão os resultados obtidos da classificação. Caso esta esteja mais perto dos 45 graus, os resultados da classificação serão o pior possível.

- Também nos indica que a área inferior à curva mede o funcionamento do algoritmo quanto a classificação de atributos. Novamente, uma área de será perfeita quando valor for 1 e péssima quando a 0.5.

Para obter esta curva foi utilizado primeiro método de *Split Learning/Test*. Neste, o algoritmo pega numa percentagem (definida pelo utilizador) do *dataset* oferecido e aprende a partir deste. De seguida aplica o que aprendeu à restante percentagem que sobrou. E.g. na nossa investigação foi usado este método, para um divisão de 70/30, onde foi dado 70% do *dataset* ao algoritmo para este aprender e aplicar o que aprendeu nos restantes 30%. O algoritmo que teve melhores resultados partindo deste método foi o IBk, com 70% de margem de acerto na caracterização dos sons entre as duas classes Confortável ou Stressante. Neste caso, a área ROC foi de 0.629 (63%). Pode-se observar as curvas de ROC na figura seguinte, onde *Confort* é para a classe de sons Confortáveis e *Stress* para a de sons Stressantes.

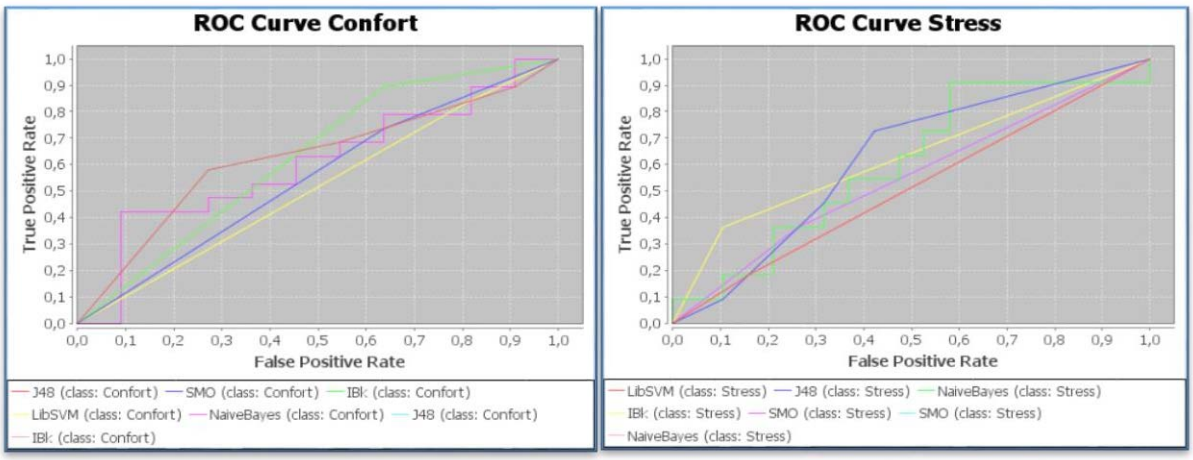


Figura 3.11 - Curvas de ROC dos algoritmos caracterizando o Dataset de Duas Classes

Algoritmo em Método Split Learning/Test	Área de ROC	% De Acerto na Caracterização
IB_K	0,629	70%
C4.5 (J48)	0,624	60%
SMO	0,550	60%
LibSVM	0,512	60%
Naive Bayes	0,603	56.66%

Tabela 3.1 - Resultados em Método de Split Learning/Test_DuasClasses

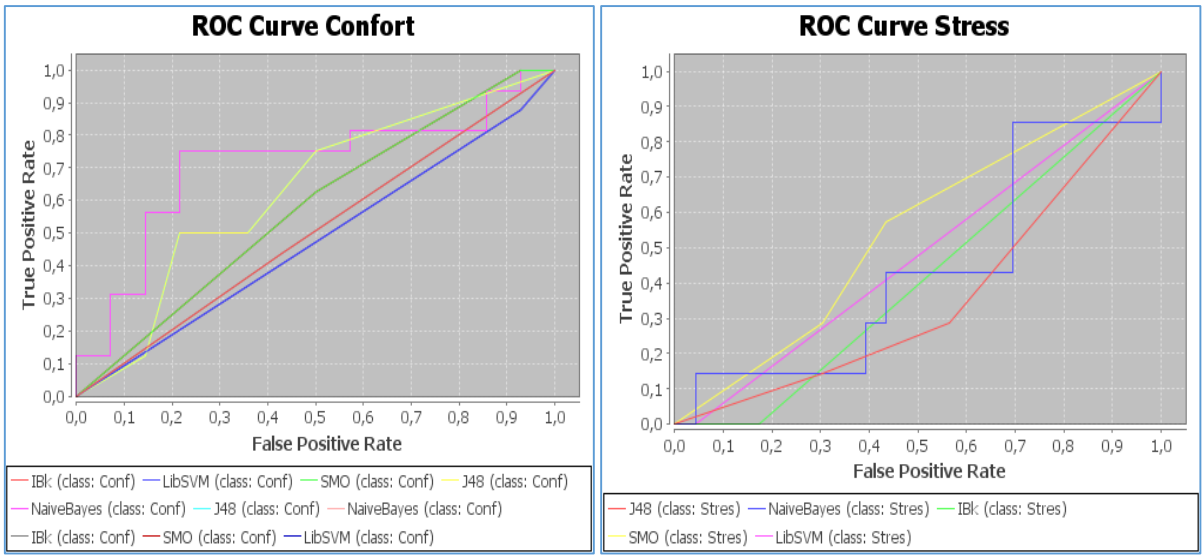
Partindo destes resultados, observamos que os algoritmos têm resultados positivos, tendo entre os cinco uma média de 61.33% de acerto na caracterização dos sons entre duas classes. Mas como o *dataset* tem apenas 100 exemplos, decidimos alterar o método para *Cross Validation* e com isto "multiplicar" o número de exemplos do *dataset*, para ver qual o comportamento dos algoritmos com este. Neste foi aplicada uma multiplicação ao número de vezes que o algoritmo vai classificar dados do *dataset*, criando virtualmente o aumento do tamanho do *dataset*. Isto é, no nosso caso, demos uma multiplicação de 10, então o algoritmo dividiu o *dataset* em dez partes e treinou com nove destas dez partes e aplica a classificação que aprendeu na última parte restante de seguida, pegou nestas partes e misturou-as e escolheu outras nove partes diferentes. O mesmo vai repetir este processo dez vezes para que todas as partes sejam classificadas.

Algoritmo em Método Cross Validation	**Área de ROC**	**% De Acerto na Caracterização**
Naive Bayes	0,694	62%
SMO	0,589	61%
IBK	0,566	59%
LibSVM	0,500	56%
C4.5 (J48)	0,592	58%

Tabela 3.2 - Resultados em Método de Cross Validation_DuasClasses

Na Tabela 3.1, observamos que o algoritmo *Naive Bayes* ficou em último lugar com pior performance. Este tem agora resultados superiores na Tabela 3.2 e encontra-se em primeiro lugar. Com isto, deparamo-nos que ao usar o método de *Cross Validation* houve uma queda na margem de acerto de todos os algoritmos à exceção do *Naive Bayes*.

Face a estes resultados, explorámos o *dataset* de três classes, usando também os dois métodos anteriores. Obtivemos primeiro as curvas de ROC para as três classes e a sua área:



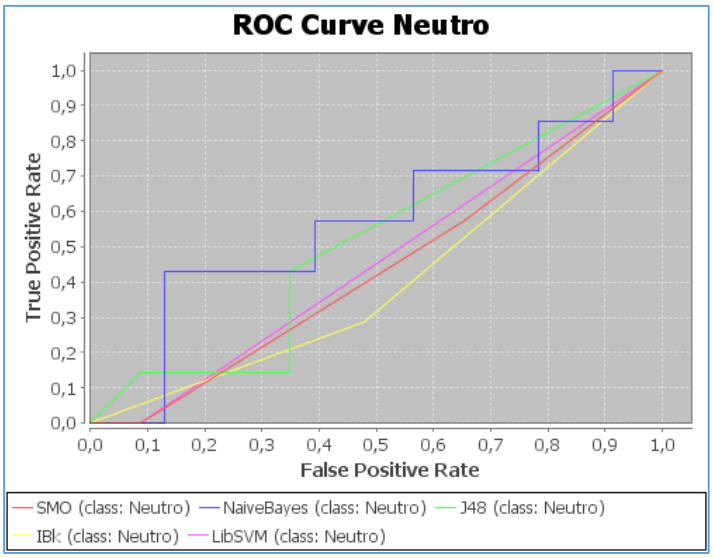


Figura 3.12 - Curvas de ROC dos algoritmos caracterizando o Dataset de Três Classes

Algoritmo em Método Split Learning/Test	Área de ROC	% De Acerto na Caracterização
Naive Bayes	0,612	53.33%
LibSVM	0,470	46.66%
SMO	0,535	43.33%
C4.5 (J48)	0,538	40%
IBK	0,460	30%

Tabela 3.3 - Resultados em Método de Split Learning/Test_Três Classess

Nos primeiros resultados obtidos usando o método de *Split Learning/Test*, deparamo-nos com uma queda bastante acentuada de 18,67 pontos percentuais, na média de acerto de todos os algoritmos na caracterização das três classes, quando comparada a média de acerto da caracterização de duas classes. Decidimos comparar estes com os do método de *Cross Validation* para tirar dúvidas.

Algoritmo em Método Cross Validation	**Área de ROC**	**% De Acerto na Caracterização**
C4.5 (J48)	0,641	51%
SMO	0,535	44%
Naive Bayes	0,575	42%
LibSVM	0,500	42%
IBK	0,537	40%

Tabela 3.4 - Resultados em Método de Split Learning/Test_Três Classess

Aqui na Tabela 3.4 deparamo-nos com uma queda de 7.4 pontos percentuais na média de acerto, quando comparados os resultados com os da Tabela 3.1. Observamos também um aumento de 1.14 pontos percentuais na média de acerto, ao comparar com o método de *Split Learning/Test* da Tabela 3.3 na caracterização de três classes.

Quando observamos a queda acentuada na margem de acerto do algoritmo IBk no caso de o *dataset* ter três classes, decidimos ir investigar o porquê da mesma. Concluímos que o algoritmo não tem exemplos suficientes dos quais possa aprender, porque os mesmos estão mais espalhados e em menor número nas três classes. Com isto fizemos novos testes só com este algoritmo no *dataset* de três classes, para perceber se seria possível melhorar a sua performance. Modificamos os atributos do algoritmo e permitimos que o mesmo aprendesse de cinco vizinhos ao invés de um como estava estipulado anteriormente. Obtivemos de imediato um aumento da percentagem de acerto deste na caracterização do *dataset* de três classes.

Algoritmo	Método de Split Learning/Test % De Acerto	Método de Cross Validation % De Acerto
IBk 5 “vizinhos”	50%	48%

Tabela 3.5 - Resultados de Acerto do Algoritmo IBk com 5 "vizinhos" no Dataset de três classes

Podemos observar que existe um aumento de 20 e 18 pontos percentuais em comparação aos testes com o algoritmo a aprender por apenas 1 vizinho. Decidimos por curiosidade ver qual o comportamento do algoritmo, aquando da caracterização do *dataset* de duas classes com cinco vizinhos.

Algoritmo	Método de Split Learning/Test % De Acerto	Método de Cross Validation % De Acerto
IBk 5 “vizinhos”	66.66%	61%

Tabela 3.6 - Resultados de Acerto do Algoritmo IBk com 5 "vizinhos" no Dataset de duas classes

De seguida, começamos a explorar os resultados relacionados com *precision*, *recall* e *F-measure*. Estas são que variam entre 0 e 1, sendo 1 uma indicação de uma boa performance de um algoritmo. Ao aferir a qualidade de um algoritmo de classificação, há que considerar a sua capacidade de reconhecer verdadeiramente membros da classe em questão (*true positives* – TP), de reconhecer verdadeiramente membros que não são da classe (*true negatives* - TN), mas também a indicação de que pertencem à classe elementos que na realidade não pertencem (*false positives* – FP) e a indicação de que não pertencem à classe elementos que na realidade deveriam ser incluídos (*false negatives* – FN). Dir-se-ia que uma medida sobre a capacidade de classificar verdadeiramente seria adequada, mas muitas vezes é incompleta. Essa medida é a acuidade (*acuity*) de um algoritmo e é dada por:

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$

Para casos em que não há um balanço de elementos adequado entre classes esta medida pode dar resultados muito pouco fidedignos. Para isso criamos a medida de *recall*, que dá uma

indicação clara da capacidade do algoritmo indicar o máximo de elementos que de facto pertencem à classe, fazendo:

$$R = \frac{TP}{TP + FN}$$

Um algoritmo com bom *recall* recupera muitos elementos que importam ao utilizador, deixando poucos dos elementos relevantes de fora, mas pode fazê-lo a troco de recuperar também muito lixo. Para isso utilizamos o conceito de *precision*:

$$P = \frac{TP}{TP + FP}$$

Como seria conveniente ter apenas um valor que nos indicasse a robustez de um algoritmo sem descriminar entre as capacidades patenteadas pelas medidas de *precision* e *recall*, introduzimos a *f-measure*, uma média harmónica ponderada entre as duas medidas, que normalmente é apresentada como *F1-score*, em que a ponderação é idêntica (um meio – um meio):

$$F1 = \frac{P + R}{2PR} = \frac{2TP}{2TP + FP + FN}$$

Começando pelo método de *Split Learning/Test* com os algoritmos a caracterizar o *dataset* de duas classes. Quando deparados com os resultados dos algoritmos nestas três medidas, notámos uma quebra na medida de *precision* em comparação com o *recall*, em todos os algoritmos exceto o C4.5. O mesmo pode ser observado na Tabela 3.9.

Algoritmo em Método Split Learning/Test	**Precision**	**Recall**	**F1-score**
C4.5 (J48)	0,600	0,600	0,600
SMO	0,585	0,600	0,590
Naive Bayes	0,518	0,567	0,528
LibSVM	0,552	0,600	0,552
IBK	0,693	0,700	0,673

Tabela 3.7 - Resultados das Três Medidas em dataset de duas classes - Split Learning/Test

Isto indicou-nos que os nossos algoritmos exceto o C4.5, estavam a devolver mais resultados relevantes que do que a classificar corretamente. O C4.5 devolvia também resultados relevantes, mas ao mesmo tempo, classificava-os com mais precisão.

Algoritmo em Método Cross Validation	Precision	Recall	F1-score
C4.5 (J48)	0,588	0,580	0,582
SMO	0,603	0,610	0,601
Naive Bayes	0,613	0,620	0,604
LibSVM	0,509	0,560	0,454
IBK	0,580	0,590	0,578

Tabela 3.8 - Resultados das Três Medidas em dataset de duas classes - CrossValidation

Quando comparados os resultados do método *split learning/test* com os do *cross validation*, observamos que em geral, existe uma melhor prestação na classificação aquando o uso do método de *split learning/test*. Faltava-nos agora comparar estes resultados com os obtidos do *dataset* de três classes.

Quando fomos confrontados com estes resultados, reparámos que houve uma quebra acentuada na devolução de resultados. Fizemos uma avaliação aos algoritmos e ao compararmos os resultados dos métodos, ficou concluído que o que estava a influenciar estes resultados era o tamanho do *dataset* para as três classes. Sendo que aquando do uso do método de *cross validation* obtivemos melhores resultados devido à "multiplicação virtual" do *dataset*.

Algoritmo em Método Split Learning/Test	Precision	Recall	F1-score
C4.5 (J48)	0,481	0,400	0,425
SMO	0,361	0,433	0,394
Naive Bayes	0,511	0,620	0,604
LibSVM	0,277	0,467	0,347
IBK	0,323	0,300	0,304

Tabela 3.9 - Resultados das Três Medidas em dataset de duas classes – Split Learning/Test

Algoritmo em Método Cross Validation	Precision	Recall	F1-score
C4.5 (J48)	0,588	0,580	0,582
SMO	0,603	0,610	0,601
Naive Bayes	0,613	0,620	0,604
LibSVM	0,509	0,560	0,454
IBK	0,580	0,590	0,578

Tabela 3.10 - Resultados das Três Medidas em dataset de duas classes - CrossValidation

Com estes resultados decidimos analisar e tentar perceber se existia diferença entre a utilização dos descritores de áudio pelos algoritmos aquando da classificação do *dataset*. Para isto utilizamos algoritmos de análise de descritores do WEKA, com isto comparamos quais destes tem mais impacto na nossa caracterização sonora.

Na Tabela 3.11 podemos observar o *ranking* dos descritores extraídos e assim perceber quais destes tem mais impacto na caracterização do *dataset* de três classes. O número apresentado é o coeficiente de determinação (o quadrado da correlação de Pearson) que nos dá uma ideia do quanto da variabilidade da variável dependente é explicada por alterações na variável independente. Os valores variam entre 0 e 1 sendo 1 o máximo atingível. Embora genericamente baixos, há que considerar que em classificação ou regressão são as ponderações entre os coeficientes utilizados para cada descritor que fornecem indicação da robustez geral do sistema. Aqui tentamos apenas indicar contributos parcelares.

RANKING DOS DESCRITORES DE ÁUDIO – TRÊS CLASSES			
1 - 0.2043	MFCC1	**17 - 0.0901**	MFCC4
2 - 0.1955	Chroma_AveragePeakRatio	**18 - 0.0856**	MFCC12
3 - 0.1792	ZeroCrossing	**19 - 0.0845**	Chroma_E
4 - 0.1564	Chroma_A	**20 - 0.083**	MFCC9
5 - 0.1491	Chroma_A#	**21 - 0.0728**	MFCC5
6 - 0.1452	MFCC8	**22 - 0.0618**	Chroma_G
7 - 0.1395	MFCC2	**23 - 0.0608**	MFCC6
8 - 0.1356	SpectralCentroid	**24 - 0.0504**	MFCC3
9 - 0.1272	Chroma_F	**25 - 0.0483**	Chroma_D#
10 - 0.1248	Chroma_MinimunPeakRatio	**26 - 0.0471**	Chroma_C
11 - 0.1152	Chroma_F#	**27 - 0.0447**	Chroma_C#
12 - 0.1146	MFCC7	**28 - 0.0438**	SpectralRolloff
13 - 0.1018	Chroma_B	**29 - 0.0408**	Chroma_D
14 - 0.096	Chroma_G#	**30 - 0.0267**	MFCC11
15 - 0.0944	MFCC10	**31 - 0.0255**	MFCC0
16 - 0.0916	Spectralflux		

Tabela 3.11 - Rank dos Descritores Áudio no Dataset de Três Classes

Nesta tabela mostramos quais os descritores com mais impacto na caracterização sonora que são: MFFC1; *Chroma_AveragePeakRatio*; *ZeroCrossing*; *Chroma_A*; e *Chroma_A#*. Estes parecem ser os descritores mais úteis para o nosso algoritmo de aprendizagem máquina. A presença de um MFCC entre os descritores mais relevantes não é surpreendente. Como previamente referido, estes descritores têm vindo a provar uma utilidade quase universal em tarefas de classificação áudio. Os descritores Chroma, no entanto, não farão demasiado sentido, pois revelam tendências tonais e não é de esperar (embora estejamos abertos a surpresas) que um indicador de tonalidade absoluta em Lá (Chroma_A) ou Lá sustenido (Chroma_A#) seja um bom previsor de valência emocional sobre sons simples. O descritor ZeroCrossing costuma estar associado com uma escala harmónico-ruidoso, pelo que é aceitável que caracterize o fenómeno que estudamos.

Como expectável, o mesmo *ranking* altera quando caracterizamos um *dataset* de duas classes, como podemos observar na Tabela 3.12:

RANKING DOS DESCRITORES DE ÁUDIO – DUAS CLASSES			
1 - 0.34185	Chroma_AveragePeakRatio	**17 - 0.07191**	MFCC11
2 - 0.31673	MFCC2	**18 - 0.0689**	Chroma_G
3 - 0.287	MFCC1	**19 - 0.06058**	Chroma_C#
4 - 0.1924	SpetralFlux	**20 - 0.05085**	Chroma_B
5 - 0.16246	Chroma_F	**21 - 0.04592**	MFCC4
6 - 0.16161	Chroma_A	**22 - 0.03663**	MFCC5
7 - 0.15872	ZeroCrossing	**23 - 0.03281**	Chroma_E
8 - 0.14755	MFCC8	**24 - 0.03103**	MFCC3
9 - 0.14248	Chroma_A#	**25 - 0.02899**	MFCC6
10 - 0.14098	Chroma_MinimunPeakRatio	**26 - 0.02536**	Chroma_D
11 - 0.12823	Chroma_F#	**27 - 0.01293**	Chroma_D#
12 - 0.11674	MFCC10	**28 - 0.01085**	Chroma_C
13 - 0.10868	MFCC7	**29 - 0.01048**	MFCC12
14 - 0.10776	SpectralCentroid	**30 - 0.01044**	MFCC0
15 - 0.09976	MFCC9	**31 - 0.00894**	SpectralRolloff
16 - 0.08064	Chroma_G#		

Tabela 3.12 - Rank dos Descritores Áudio no Dataset de Duas Classes

Nesta versão, podemos observar que para além de os cinco primeiros descritores terem alterado, estes, tem melhores resultados, atingindo 0.34 no *Chroma_Average Peak Ratio* e 0.32 no MFCC2. Enquanto na Tabela 3.11 o valor máximo foi de 0.2 com o descritor MFCC1, este está em terceiro lugar com 0.28 na caracterização de duas classes, ou seja, o mesmo parece ter mais impacto com menos classes.

3.3 Conclusão de Capítulo

Com isto concluímos a parte empírica da nossa investigação. Percorremos caminhos inicialmente inesperados, mas o percurso foi deveras interessante e estimulante de construir. Em suma, deveríamos ter usado os 318 exemplos colecionados inicialmente ao invés de retirar 100 destes. Assim poderíamos ter uma classificação mais fundamentada para o sistema. Concluímos também que o método mais fiável para a criação de um sistema automático de aprendizagem máquina para a caracterização de sons como Confortáveis ou Stressantes é o *Split Learning/Test*. Neste método foi onde obtivemos melhores resultados aquando na classificação de apenas duas classes com o algoritmo IBk. Com isto confirmámos também que para nosso sistema ter boa performance e melhores resultados na caracterização de três classes, necessita de um *dataset* maior, de forma a oferecer uma maior variedade de exemplos através dos quais os algoritmos possam aprender.

4 Conclusões e perspetivas de trabalho futuro

Conseguimos com os nossos resultados demonstrar que é possível a criação de um sistema automático para a caracterização de sons como Confortáveis ou Stressantes partindo do uso de aprendizagem máquina. Estamos conscientes de que este teria de ser exposto a outros *datasets* para otimização de desempenho, mas conseguimos uma prova de conceito da metodologia que nos deu conhecimento de causa para poder levar a investigação a um nível mais operacional. Alguns aspetos deveriam ter sido abordados de maneiras diferentes, mas tal não foi possível porque implicaria uma curva de aprendizagem e investimento temporal excessivo para o âmbito do mestrado. Entre estes contam-se:

- A criação de um *software* onde estivesse implementado o sistema de classificação de forma a que fosse possível caracterizar sons em tempo real, e.g. numa aplicação para telemóvel.

- A utilização de outros *softwares* para extrair os descritores áudio, comparando resultados entre si, e avaliando o impacto adicional que trariam aquando nos esquemas de classificação pelo WEKA.

Ambas as abordagens acima pronunciadas podiam por si só ser um tema de investigação individual, algo que ficará decerto como hipótese para trabalhos futuros. Mas como observámos no estado de arte, não há investigações que fiquem fechadas, pois podemos sempre abordar as mesmas de diferentes formas.

Encontrámos dificuldades durante o nosso percurso como de esperado, e estas trouxeram questões tais como:

- Poderíamos ter feito uma melhor abordagem na nossa investigação? Com certeza, poderíamos ter explorado aspetos como classificar os *datasets* com diferentes combinações de descritores de algoritmos ao invés de usar todos numa só classificação. Isto daria uma abordagem completamente diferente da nossa, que até nos poderia ter oferecido um sistema com melhores margens de acerto. É típico utilizar algoritmos de otimização de escolha de descritores, como o *Principal Component Analysis* (PCA), algo que complexificaria em demasia a nossa prova de conceito. Poderíamos também ter criado um grupo de *datasets* ao invés de um só e aplicar a aprendizagem máquina de um para os restantes e comparar resultados.

- Uma das dificuldades que pensamos ter tido bastante influência nos nossos resultados foi o uso do MARSYAS. Isto porque reparámos que o MARSYAS tinha influência na classificação do *dataset* devido à estratégia de extração. Aquando da criação dos ficheiros ".arff" que depois eram recebidos no WEKA, o MARSYAS criava exemplos "desnecessários", o mesmo inicialmente pegava nos 100 exemplos e dividia-os em 16169 pequenos *frames* de exemplos. Isto criou inicialmente bastante confusão nas nossas classificações porque o sistema estava a caracterizar "*frames* de sons" como confortáveis ou stressantes ao invés de caracterizar sons completos. Isto trouxe influência nos resultados, pois estes estavam com margens de acerto de classificação altíssimas por causa do WEKA reconhecer o *dataset* como 16169 exemplos ao invés de 100 apenas. Isto permitia aos algoritmos uma aprendizagem muito maior e mais variada.

- Poderíamos ter explorado novos algoritmos ao invés de usar os mais habituais na área científica? Vimos que era possível fazer uma abordagem ao facto de uso de algoritmos pouco usados e a comparação de resultados destes com os mais habituais. Isto até nos poderia oferecer um novo tema de investigação que envolveria a classificação de algoritmos oferecidos pelo WEKA, em que comparando a performance destes sobre vários *datasets* previamente criados e classificados criaríamos um *ranking* de eficácia de algoritmos.

- Deveríamos ter usado mais exemplos para o *Dataset* ao invés de 100 exemplos? Sim, esta questão ficou respondida aquando a análise dos resultados, pois os mesmos não eram os esperados que seriam com margens de acerto superiores a 85%. A melhor margem obtida foi de 70% no uso do IBk, isto indicou-nos que o nosso sistema se fosse afrontado com *datasets* de grandes proporções com o que tinha aprendido do nosso *dataset* iria ter grandes percas, e.g. num *dataset* com 1000 exemplos, o nosso sistema iria sempre falhar na classificação de 300 exemplos como Confortáveis ou Stressantes. Esta situação não seria de todo aceitável num meio real. Para isto a solução mais viável e seguindo os nossos resultados obtidos, seria a criação de um *software* que tivesse nele implementado o algoritmo de extração de descritores de áudio, o algoritmo de aprendizagem máquina IBk e que este pudesse aprender de vários *datasets* e evoluir automaticamente.

- Com isto vem a questão de ensinar manualmente aos algoritmos como classificar certos exemplos dos resultados obtidos do *dataset*, e.g. dizer-lhe que o exemplo *x* não pertencia à classe *y* como ele previu mas sim à classe *z*. Isto iria dar fruto a que o nosso *dataset* pudesse ser reaproveitado.

Em suma, verificamos que os nossos resultados estiveram aquém do esperado inicialmente. Sim é possível construir um sistema automático funcional a partir destes, mas o mesmo teria de ser melhorado substancialmente. Todos os algoritmos teriam de ser revistos e os seus atributos comparados de diferentes formas. Todos os descritores teriam de ser analisados a fundo e extraídos de diferentes *softwares* para além do MARSYAS e finalmente, devia ter sido usado todo o *dataset* inicial e classificado com mais participantes. Todas estas questões poderão ser abordadas em trabalhos futuros.

Referências e Bibliografia

Andrieu, C., de Freitas, N., Doucet, A., & Jordan, M. I. (2003). An Introduction to MCMC for Machine Learning. *Machine Learning*, 5–43.Retirado de http://doi.org/10.1023/A:1020281327116

Baume, C., Fazekas, G., & Barthet, M. (2014). Selection of Audio Features for Music Emotion Recognition Using Production Music. *AES 53rd International Conference: Semantic Audio*, 1–9. Retirado de http://www.aes.org/e-lib/browse.cfm?elib=17110

Bech, S., & Zacharov, N. (2006). *Perceptual Audio Evaluation - Theory, Method and Application*. John Wiley & Sons.

Bello, J. (n.d.). Chroma and tonality. *Nyu.Edu*, 1–45. Retirado de http://www.nyu.edu/classes/bello/MIR_files/tonality.pdf

Caetano, M., & Wiering, F. (2014). Theoretical framework of a computational model of auditory memory for music emotion recognition. *International Symposium on Music Information Retrieval*, (Ismir), 331–336. Retirado de http://www.terasoft.com.tw/conf/ismir2014/

Chang, C., & Lin, C. (2011). LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*, 1–39. Retirado de http://doi.org/10.1145/1961189.1961199

Brynjolfsson, E., & Mcafee, A. (2014). The Second Machine Age - Work, progress, and prosperity in a Time of Brilliant Technologies, 1–142.Retirado de http://doi.org/10.1177/0019793914565342

Dhanaraj, R., Logan, B. & Machines, V. (2005). Automatic Prediction of Hit Songs. *International Conference on Music Information Retrieval*, (September) HPL-2005-149, 11–15. Retirado de http://doi.org/10.1111/mono.12110

Downie, J. S. (2008). The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, *29*(4), 247–255. Retirado de http://doi.org/10.1250/ast.29.2477

Downie, X. H. J. S., Laurier, C., & Ehmann, M. B. A. F. (2008). The 2007 MIREX audio mood classification task: Lessons learned. In *Proc. 9th Int. Conf. Music Inf. Retrieval* (pp. 462-467).

Elaiyaraja, V., & Meenakshi, P. (2012). Audio Classification Using Support Vector Machines and Independent Component Analysis, *Journal of Computer Applications Volume V,* 34–38.

Fan, J., Thorogood, M., Riecke, B. E., & Pasquier, P. (2015). Automatic Recognition of Eventfulness and Pleasantness of Soundscape. *Proceedings of the Audio Mostly 2015 on Interaction With Sound*, 12:1–12:6. Retirado de http://doi.org/10.1145/2814895.2814927

Gladwell, M. (2008). *Outliers: The story of success*. Hachette UK.

Geller, J. (2002). Data Mining : Practical Machine Learning Tools and Techniques with Java Implementations, *SIGMOD Record, Vol. 31, No. 1(March)*, 76–77.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10. Retirado de http://doi.org/10.1145/1656274.1656278

Hantke, S., Weninger, F., Han, W., Zhang, Z., & Narayanan, S. (2012). Bj ¨ Institute for Human-Machine Communication, Technische Universit ¨ at M ¨ unchen, Germany School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China Signal Analysis and Interpretation Lab, University of Southern Cal. *Icassp2012*, (Section 2), 341–344.

Kim, J., & Larsen, L. (2010). Music Emotion and Genre Recognition Towards New Affective Music Taxonomy. *128th Audio Engineering Society Convention*.

Lerch, A. (2012). *An introduction to audio content analysis: Applications in signal processing and music informatics*. John Wiley & Sons.

Li, T., Ogihara, M., & Li, Q. (2003). A comparative study on content-based music genre classification. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval SIGIR 03*, *15*(5), 282. Retirado de http://doi.org/10.1109/ISSPA.2003.1224828

Liebetrau, J., & Schneider, S. (2013). Music and Emotions : A Comparison of Measurement Methods. *134th Convention Audio Engineering Society (May), 4-7.*

Liebman, E., Stone, P., & White, C. N. (2015). How music alters decision making - impact of music stimuli on emotional classification. *16th International Society for Music Information Retrieval Conference (ISMIR 2015)2*, 793–799.

Lorena, a. C., & de Carvalho, a. C. P. L. F. (2007). Uma Introdução às Support Vector Machines. *Revista de Informática Teórica E Aplicada*, *14*(2), 43–67. Retirado de http://seer.ufrgs.br/index.php/rita/article/viewArticle/rita_v14_n2_p43-67

Logan, B. (2000). Mel Frequency Cepstral Coefficients for Music Modeling. *International Symposium on Music Information Retrieval*, *28*, 11p. Retirado de http://doi.org/10.1.1.11.9216

Mathworks, C. (2014). Primer R 2014 a. Retirado de http://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf

Melamed, I. D., Green, R., & Turian, J. P. (2003, May). Precision and recall of machine translation. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003--short papers-Volume 2 (pp. 61-63). Association for Computational Linguistics.

Mierswa, I., & Morik, K. (2004). Automatic Feature Extraction for Classifying Audio Data, 1–28.

Mohino-herranz, I. (2014). Creation of New Virtual Patterns for Emotion Recognition through PSOLA, *136th Convention Audio Engineering Society (April)*, 26-29.

Monard, M. C., & Baranauskas, J. A. (2003). Conceitos sobre Aprendizado de Máquina. *Sistemas Inteligentes: Fundamentos E Aplicações*, 39–56. Retirado de http://dcm.ffclrp.usp.br/~augusto/publications/2003-sistemas-inteligentes-cap4.pdf

Ni, Y., & Santos-Rodriguez, R. (2011). Hit song science once again a science. *In 4th International Workshop on Machine Learning and Music.* 2–3. Retirado de http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.412.9732&rep=rep1&type=pdf

Özcan, E., Egmond, R. Van, & Jacobs, J. J. (2014). Product Sounds : Basic Concepts and Categories, *International Journal of Design Vol 8*(3), 97–111.

Pachet, F., & Roy, P. (2008, September). Hit Song Science Is Not Yet a Science. In ISMIR (pp. 355-360).

Pachet, F., & Sony, C. S. L. (2012). Hit song science. *Tzanetakis & Ogihara Tao, Editor, Music Data Mining*, 305–326.

Peeters, G. (2004). A large set of audio features for sound description (similarity and classification) in the CUIDADO project. *CUIDADO IST Project Report*, *54*(0), 1–25. Retirado de http://doi.org/10.1234/12345678

Percival, G., & Tzanetakis, G. (2008). Marsyas User Manual.

Piczak, K. J. (2015). ESC: Dataset for Environmental Sound Classification. *Proceedings of the 23rd ACM International Conference on Multimedia, MM 2015*, 1015–1018. Retirado de http://doi.org/10.1145/2733373.2806390

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods Support Vector Learning*, *208*, 1–21. Retirado de http://doi.org/10.1.1.43.4376

Saitta, L. (1995). Support-Vector Networks, Machine *Learning*, 20, 273–297.

Steidl, S. (2009). *Automatic Classification of Emotion-Related User States in Spontaneous Children's Speech*, Dissertação Universidade Erlangen-Nürnberg 250.

Tsuchiya, M., Asada, a, Ryo, K., Noda, K., Hashino, T., Sato, Y., … Inoue, M. (2003). Relaxing intraoperative natural sound blunts haemodynamic change at the emergence from propofol general anaesthesia and increases the acceptability of anaesthesia to the patient. *Acta Anaesthesiologica Scandinavica*, *47*(8), 939–943. Retirado de http://doi.org/10.1034/j.1399-6576.2003.00160.x

Umapathy, K., Member, S., Krishnan, S., Member, S., Rao, R. K., & Member, S. (2007). Audio Signal Feature Extraction and Classification Using Local Discriminant Bases, *15*(4), 1236–1246.

Wu, X., Kumar, V., Ross, Q. J., Ghosh, J., Yang, Q., Motoda, H., … Steinberg, D. (2008). *Top 10 algorithms in data mining*. *Knowledge and Information Systems* (Vol. 14). Retirado de http://doi.org/10.1007/s10115-007-0114-2

APÊNDICE A

Conteúdos em Formato Digital (DVD)

1. ARDUINO

1.1 – RFIDINO

2. Dataset

2.1 – 2 Classes
2.2 – 3 Classes
2.3 – DATASETTOTAL

3. Descritores Áudio de Baixo Nível

3.1 – WEKA_DATASET2CLASSES
3.2 – WEKA_DATASET3CLASSES

4. MAX-MSP

4.1 – Classificação_Dataset