INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Plataforma para recolha e identificação automática de espécies de pássaros utilizando registos áudio

DAYLSON DE CARVALHO VERA CRUZ
Outubro de 2016

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Plataforma para recolha e identificação automática de espécies de pássaros utilizando registos áudio

Daylson de Carvalho Vera Cruz

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática, Área de Especialização em Sistemas Computacionais

Orientador: Elsa Maria de Carvalho Ferreira Gomes

Porto, Outubro de 2016

Resumo

A análise e modelação dos sons de animais com o fim de estudar a sua biodiversidade são possíveis graças a uma ciência multidisciplinar denominada bioacústica. Este trabalho tem como objetivo a construção de uma solução capaz de identificar automaticamente as espécies de pássaros através do som emitido e ainda construir uma base de dados etiquetada com base nos registos áudio colhidos.

Para a classificação dos sons dos pássaros, este trabalho segue duas metodologias para a extração de atributos e recorre a tarefas do *data mining*. Uma das metodologias de extração de atributos baseia-se na descoberta dos padrões mais frequentes no som, denominado (*Motifs*). A outra metodologia bastante usada em reconhecimento de sons, baseia-se em coeficientes *Mel-Frequency* (MFCCs).

Para a classificação são usados algoritmos de árvores de decisão e *Support Vector Machines*. Foram também avaliadas diferentes metodologias de pré-processamento dos sons.

Para a realização das experiências, foi utilizado o *dataset* disponibilizado pela *Neural Information Processing Scaled for Bioacustic Bird song competition* (NIPS4B).

Nas experiências realizadas, o processo de classificação utilizando técnicas de normalização do sinal original no pré-processamento, a extração de atributos usando a abordagem MFCCs e o algoritmo de classificação *Random Forest* apresentou melhores resultados.

Neste trabalho foi também desenvolvida uma aplicação móvel para o sistema operativo *Android* capaz de gravar os sons e identifica-los utilizando os recursos da aplicação servidor (*web*), com um tempo médio de resposta de 20 segundos.

Palavras-chave: Classificação de sons, *data mining*, *motifs*, MFCC.

Abstract

The analysis and modeling of animal sounds in order to study its biodiversity are possible thanks to a multidisciplinary science called bioacoustics. This work aims to build a client / server platform able to automatically identify the species of birds through the emitted sound and still build a database labeled based on the collected audio recordings. For the classification of the sounds of the birds, this work follows two methods for the extraction of attributes and uses of data mining tasks.

One feature extraction methodology, is based on the discovery of the most frequent patterns in sound, called (Motifs). The other method, often used for sound recognition is based on Mel-Frequency coefficients (MFCCs). For the classification are used decision trees algorithms and Support Vector Machines. They were also evaluated different methods for pre-processing of sounds.

To carry out the experiments, we used the dataset provided by the Neural Information Processing Scaled for Bioacustic Bird song competition (NIPS4B).

In the experiments conducted, the classification process using normalization techniques of the original signal in the pre-processing, extracting attributes using the MFCCs approach and Random Forest classification algorithm showed the best results.

This work also developed a mobile application for the operating system Android able to record sounds and identify them using the resources of the application server, with a response average time of 20 seconds.

Keywords: Sound Classification, data mining, motifs, MFCC

Agradecimentos

Agradeço a todos os meus familiares e amigos que de uma forma ou de outra contribuíram para a realização desta tese de mestrado.

Gostaria deixar um especial obrigado a Prof. Elsa Gomes, que sempre se mostrou disponível para ajudar no desenvolvimento deste trabalho.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos e Símbolos

Lista de Acrónimos

ADT	*Android Develop Tools*
ARBIMON	*Automated Remote Biodiversity Monitoring Network*
AHA	*Area of High Amplitude*
ASR	*Automatic Speech Recognition*
AUC	*Area Under the Curve*
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
CSV	Comma-Separeted Values
FNJV	Fonoteca Neotropical Jacques Vielliard
GMM	*Gaussian Mixtures Models*
GNU	*General Public Licence*
GPS	*Global Positioning System*
HMM	*Hidden Markov Models*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
HZ	Hertz
IDE	*Integrated Development Environment*
KNN	K-*Nearest Neighbors*
LDA	*Linear Discriminant Analyses*

LOLAS	*Lenght of Low Amplitude Sequence*
LPC	*Linear Predictive Coding*
LPCC	*Linear Prediction Cepstral Coefficients*
LVQ	*Linear Vector Quantization*
MFCC	*Mel-Frequency Cepstral Coefficient*
MLP	*Multilayer Perceptron*
mRMR	*minimal Redundancy-Maximal Relevance*
NIPS4B	*Neural Information Processing Scaled for Bioacustic*
PCC	*Progressive Constructive Clustering*
PCM	*Pulse Code Modulation*
PHP	*Personal Home Page*
ROI	*Region of Interest*
SMO	*Sequencial Minimal Optimization*
SOAP	*Simple Object Access Protocol*
STFT	*Short-Term Fourier Transform*
SVM	*Support Vector Machines*
UNICAMP	Universidade Estadual de Campinas
URL	*Uniform Resource Locator*
UUID	*Universally Unique Identifier*
WASIS	*Wildlife Animal Soud Identification System*
XML	*eXtensible Markup Language*

Lista de Símbolos

∑	Sigma (Somatório)
α	Alfa

$\in$	Pertence ao conjunto
μ	Micro
σ	Sigma (minúsculo)

1 Introdução

Neste capítulo apresentamos o problema, a sua contextualização e os objetivos pretendidos. Apresentamos ainda, de forma resumida a análise de valor e a abordagem preconizada.

1.1 Contexto

A bioacústica pretende analisar e modelar os sons de animais para fins de estudo da biodiversidade. Para esse estudo é útil ter ferramentas de recolha de sons, metadados associados e respetivo processamento, incluindo classificação automática. Dada a complexidade destes dados e as diferentes taxonomias das espécies, bem como o ambiente envolvente, este estudo requer abordagens originais.

1.2 Definição do problema

O objetivo deste trabalho é produzir uma plataforma de apoio à recolha de registos áudio que permita a construção de uma base de dados etiquetada. A plataforma deverá incluir um servidor, onde são armazenados os dados, e um cliente que deverá ser uma aplicação móvel para recolha de sons e de anotações feitas por biólogos ou população em geral. O servidor deverá também ser capaz de identificar espécies de pássaros que existam na base de dados, quando é recolhido um novo som pela aplicação.

1.3 Análise de valor

Pensamos que o trabalho descrito nesta tese tem valor de negócio dado que tem interesse de utilização para captar, etiquetar e classificar os sons dos pássaros existentes em S.Tomé e Príncipe. Nestas ilhas, que fazem parte dos países em via desenvolvimento, estão ser tomadas várias medidas para um crescimento sustentável. Uma delas é a criação de parques e reservas naturais, com o objetivo de manter a continuação das espécies. Neste país encontra-se várias espécies endémicas de pássaros que se encontram em via de extinção. Dada a pureza e a biodiversidade das ilhas, esta proposta se enquadra de uma forma perfeita a realidade deste país.

Este assunto será tratado de forma mais detalhada na secção 2.3.

1.4 Abordagem

Neste trabalho, pretende-se aplicar a metodologia de classificação de sons de pássaros usando os *motifs*. Esta metodologia já foi aplicada com sucesso noutro tipo de sons, tais como sons urbanos e cardíacos [Batista, 2015] [Oliveira,2014]. É também objetivo deste trabalho comparar os resultados da aplicação desta metodologia com outra tecnologia, frequentemente utilizada neste tipo de problemas, baseada em *Mel-frequency Cepstrum Coefficients* (MFCCs) [Davis and Mermelstein, 1980].

A principal contribuição e resultados deste trabalho serão uma solução, respetivo modelo de dados e uma metodologia capaz de captar, gravar e classificar registos áudio.

1.5 Estrutura da dissertação

Esta dissertação está composta do seguinte modo:

No capítulo 2, é feita a contextualização e o estudo do estado da arte.

No capítulo 3, é tratado os aspetos relacionados com a classificação de sons.

No capítulo 4, é feita a descrição das ferramentas e tecnologias usadas para este trabalho.

No capítulo 5, é apresentado a construção da solução.

No capítulo 6, é feita a descrição e demonstração dos testes realizados e dos resultados obtidos.

No capítulo 7, apresenta-se a conclusão e trabalho futuro.

2 Contexto e Estado da Arte

Neste capítulo, para além de uma contextualização do problema, faz-se um levantamento do estado da arte relacionada com a classificação no ramo da bioacústica.

2.1 A Biodiversidade das Aves

Nesta seção, pretende-se fazer uma breve introdução sobre diversidade dos pássaros e as várias formas de os identificar, dado o grande número de espécies existentes e a sua taxonomia.

Desde os tempos mais antigos, esta espécie animal tem sido objeto de pesquisa e observação, sendo por vezes mantidas em cativeiro para ser possível observar a sua beleza e o seu canto [Marquez,2009], [Helmut Sick, 1997]. A Sociedade para a Conservação das Aves do Brasil (SAVE Brazil) defende que a presença das aves pode servir de indicador do bem-estar do homem e do ecossistema, de tal forma que é refletido no aumento ou diminuição da presença das aves quando há alguma alteração ou mudança no ecossistema [SAVE, 2015].

São muitas as espécies existentes na face da terra e tem-se vindo a verificar um aumento devido ao fato de muitas ainda não terem sido identificadas e classificadas [Conceição, 2012].

No entanto, a identificação das espécies não é uma tarefa fácil, dada a sua taxonomia. Segundo [Antas e Cavalcanti,1988] estes animais possuem um grau de parentesco com características muito comuns dentro de cada grupo. Essas características são utilizadas pelos ornitólogos para classifica-los e identifica-los.

São várias as metodologias que permitem identificar as aves, como a observação do seu tamanho, das cores predominantes, do formato de cada parte do seu corpo, como a cabeça, as asas, o dorso, a cauda, as pernas e os olhos. É também possível identifica-las através do seu habitat, do seu comportamento, da sua alimentação e do seu voo. No entanto, pelo fato de haver casos em que as espécies são morfologicamente muito semelhantes, Helmut Sick defende que apenas o canto as pode diferenciar, tomando então este critério como base de classificação [Sick,1984].

As aves, pertencendo à classe dos animais vertebrados e caracterizadas principalmente por possuírem penas e pela sua taxonomia, são classificadas em categorias. Conta-se aproximadamente com cinquenta e nove ordens e subordens e duzentas e trinta famílias.

Os pássaros, um grupo de seres vivos pertencentes à classe Aves da ordem Passeriformes, representam mais de metade de espécies de aves existentes, com mais de 5400 espécies em todo o mundo com a exceção da Antártica [Biomania, 2015]. Estes seres vivos destacam-se pela sua natureza e beleza, geralmente caracterizados pelo colorido das penas e a melodia do seu som.

Com este projeto pretende-se desenvolver uma solução tecnológica que permita, através da vocalização, identificar especificamente as espécies dos pássaros.

2.2 A Bioacústica

O som é um meio de comunicação muito valioso devido às suas propriedades, como a capacidade de poder atingir longo alcance e de poder ultrapassar os obstáculos [Pereira,2011].

Desde dos tempos mais antigos, o som é utilizado como meio de comunicação entre os animais, tanto no meio terrestre como aéreo e aquático. O som emitido pelos animais, sempre despertou alguma curiosidade. Nos anos 1600, o padre Athanasius Kircher fazia a transcrição musical do canto de pássaros. Esta curiosidade torna-se mais relevante com o aparecimento da bioacústica, que consiste no estudo dos sons emitidos por animais [Vielliard, 2009] submetidos às leis da acústica.

Apesar dos avanços tecnológicos na área da bioacústica, tornando cada vez mais fácil o estudo de sons produzidos pelos animais, esta ciência multidisciplinar é ainda pouco conhecida. Atualmente, já existem alguns estudos e tecnologias que permitem a identificação automática dos animais pela sua vocalização [Aide, Corrada-Bravo,Campos-Cerqueira,Milan,Vegae Alvarez, 2013] [Conceição ,2015] [GunaseKaram and Revathy, 2011] [Huang, Yang, Yang e Chen, 2009] [Lopes, Kaestner,Silla e Koerich, 2011].

Como referimos acima, o som emitido pelos pássaros, indivíduos pertencentes a classe Aves, é uma das formas mais comuns de identificação e classificação, pois cada espécie tem o seu som característico e é, em muitos casos, a única forma de contato, pois devido os seus costumes e habitats nem sempre é possível manter o contato visual [Wikiaves].

Segundo [Marquez, 2008], as aves podem produzir os sons com diversas finalidades como a marcação de território, o acasalamento ou até mesmo para envio de sinais de alerta. A Siringe, um órgão que faz parte do aparelho respiratório das aves é responsável pela produção e controle de som [Wikiaves, 2015] como se pode observar na Figura 1. Estas emitem o som na faixa dos 20 HZ a 20.000 Hz, muito aproximado aos limites de audição dos seres humanos.

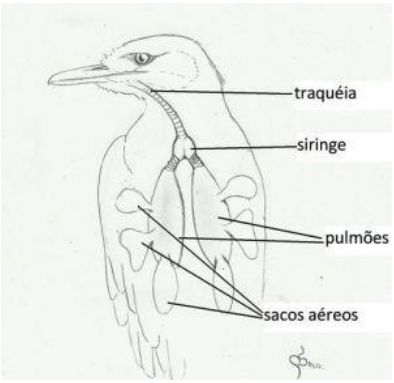


Figura 1 - Anatomia da Vocalização [Wikiaves, a].

Com este trabalho, pretendemos analisar e aplicar várias técnicas de processamento e classificação de sinais áudio no domínio da bioacústica, em particular nos sons emitidos pelos pássaros. Pretende-se que sejam aplicadas técnicas de extração de atributos como os *motifs* e MFCCs utilizadas em trabalhos desenvolvidos anteriormente em reconhecimento de sons urbanos e na detenção de patologia cardíaca [Batista, 2015] [Oliveira,2014].

2.3 Análise de valor

Em qualquer modelo de negócio é imprescindível definir bem uma Proposta de Valor (VP), pelo fato de antes de se começar um negócio ou projeto, haver a necessidade de ter bem definido o propósito do mesmo de forma a se atingir o sucesso, ou seja, importa definir que necessidade se pretende colmatar, qual será o produto/serviço a ser prestado, qual será o meu público-alvo, que utilidade terá o produto ou o serviço prestado, que custos e benefícios estarão envolvidos e porquê escolheram este produto e não outro semelhante. A viabilidade do produto/serviço terá que ser calculada, porque existem muitos fatores que poderão condicionar o êxito do negócio. A perceção que o produtor ou fornecedor terão do produto poderá ser diferente do consumidor devido a vários fatores como: o preço do produto, a qualidade, a relação entre o consumidor e o fornecedor. Estas condicionantes farão com que o consumidor tenha ou não a perceção do produto/serviço desejada pelo fornecedor ou produtor. Conclui-se então, que existe sim a necessidade de se definir bem a proposta de valor num negócio para que este seja viável.

O trabalho descrito nesta tese tem como o objetivo a produção de uma plataforma capaz de identificar as espécies de pássaros através do som emitido. Este projeto pretende atingir o mercado das ilhas de S.Tomé e Príncipe, por ser um país em via de desenvolvimento, que tem tomado várias medidas para a preservação do meio ambiente e por existir nestas ilhas várias

espécies endémicas de pássaros que se encontram em via de extinção [publico, 2016]. Esta plataforma irá permitir que o órgão competente (Direção de Recursos Naturais e Meio Ambiente) faça o estudo sobre a distribuição das espécies de pássaros pelo país, de forma a serem tomadas medidas de preservação, como também irá permitir aos biólogos interessados realizar estudos sobre a biodiversidade destas ilhas. A plataforma a ser desenvolvida neste trabalho irá ainda permitir que os amantes da natureza identifiquem as espécies de pássaros de uma forma rápida e automática.

O público-alvo é toda a comunidade científica e o público em geral amante da natureza tropical oferecida pelas maravilhosas ilhas S.Tomé e Príncipe. A plataforma permitirá à Direção de Recursos Naturais e do Meio Ambiente de S.Tomé e Príncipe, ter um maior controlo sobre os impactos ambientais causadores da extinção das espécies, para assim ser possível tomar medidas, como também permitirá aos amantes da natureza identificar as espécies de pássaros.

Numa perspetiva longitudinal de valor, esta plataforma apresenta os seguintes benefícios/sacrifícios ilustrados na Tabela 1:

Tabela 1 - Benefícios/Sacrifícios na perspetiva Longitudinal.

	Benefícios	**Sacrifícios**
Pré-Compra	• Facilidade na aquisição do produto via *PlayStory* do *Google.* • Produto Gratuito.	• Disponível apenas para Sistema Operativo *Android.*
Transação	• Ponto de Aquisição disponível em todos os dispositivos *Android.* • Segurança e Fidedignidade garantida pela PlayStory do Google.	• Compatibilidade apenas com alguns dispositivos *Android.*
Pós-Compra	• Identificação de espécies de pássaros a qualquer momento e a qualquer lugar. • Sem necessidade de contato visual com os pássaros. • Identificação rápida e fácil de espécies de pássaros.	• Gastos com 3G. • Ocupação de volume de memória do dispositivo móvel. • Gasto rápido da bateria do dispositivo móvel.
Após/Uso experiência	• Produto fácil de ser eliminado/desinstalado do dispositivo móvel. • Custo zero na eliminação ou desinstalação da aplicação.	

Este projeto apresenta um cenário de negócio do tipo *win-win*. O maior cliente alvo é a Direção de Recursos Naturais e Meio Ambiente de S.Tomé, onde todos os utilizadores irão criar um maior valor para a plataforma, avaliando-a por meio de *feedback* para assim ser possível efetuar melhorias no desempenho e aumentado a base de dados com dados etiquetados.

Será da responsabilidade da Direção de Recursos Naturais e Meio Ambiente, a criação de base de dados etiquetada, bem como a recolha de novas entradas, para que seja aumentada a potencialidade da plataforma de forma a ser possível tomar medidas de preservação do meio ambiente, sendo um fator de ganho para a Direção de Recursos Naturais e Meio Ambiente como para o país. A plataforma ganhando uma maior dimensão haverá a necessidade de efetuar um melhor controlo e gestão da mesma, bem como melhorar a sua eficiência e eficácia, sendo necessário contratar pessoal competente para tal. É ai aonde o desenvolvedor/autor da plataforma também terá o seu ganho, oferecendo o seu serviço como o autor da plataforma.

Juntamente com a Direção de Recursos Humanos e Meio Ambiente de S.Tomé e Príncipe o desenvolvedor/autor da aplicação irá procurar apoios de instituições estrangeiras ligadas ao meio ambiente, certo que haverá muitas interessadas em desenvolver estudos biológicos nestas ilhas havendo uma ferramenta local que facilite estes estudos.

De forma a se descrever melhor a ideia de negócio apresenta-se na Figura 2 o modelo de negócio de Canvas:

Figura 2 - Modelo de Negócio de Canvas para descrever BirdSound Identification.

De acordo com o mercado em que o negócio se insere, existem métodos e modelos que nos permitem analisar e/ou quantificar a criação de valor para o negócio. Estes métodos com base em análises matemáticas e/ou modelos concetuais, nos permitem analisar a situação de uma

forma racional e estrutural, de forma a tomar melhores decisões estratégicas dada as circunstâncias otimizando os *outputs* desejados. Os métodos quantitativos recorrem a técnicas numéricas que nos permitem analisar o valor do negócio, bem como escolher determinadas opções/estratégias dado um conjunto discreto de alternativas que criam um valor mais acrescentada para o negócio. Os modelos concetuais nos permitem ter uma visão estruturada do problema avaliando todos os fatores que influenciam o acréscimo do valor do negócio, permitindo-nos avaliar o peso destes fatores, de forma a se tomar melhores decisões estratégicas.

2.4 Abordagens anteriores

Existem inúmeros trabalhos realizados para a identificação e classificação de registos áudio no domínio da bioacústica. Estes trabalhos aparecem com diversos objetivos, desde a realização de pesquisas biológicas na evolução das espécies, aos impactos ambientais causados pelas atividades humanas.

A identificação automática de espécies de animais através de registos áudio permite-nos obter a informação sobre a densidade populacional de uma espécie numa determinada área, permitindo a tomada de medidas eficazes para a preservação e conservação do meio ambiente evitando a extinção destas espécies [Marquez at al.2011] [Mellinger et ai, 2007].

Diferentes trabalhos, empregam as diversas técnicas para o problema de análise e classificação de sons. O som, como um dos meios de identificação e comunicação dos animais, permite-nos obter muitas informações acerca das espécies. Na Tabela 2, são apresentadas algumas das aplicações disponíveis no mercado para identificação automática de espécies de animais através do som emitido. Na Tabela 3, referem-se alguns trabalhos de pesquisa realizados no ramo da bioacústica.

Tabela 2 - Lista de aplicações para identificação automática de espécies de animais através do som emitido.

Nome	Técnicas usadas	Descrição
Automated Remote Biodiversity Monitoring Network (ARBIMON)	• *Short-Term Fourier Transform* (STFT), para criar o espectrograma. • *Hidden Markov Models* (HMMs) • *Region of Inteerest* (ROI). • Bau-Welch algorithm	Plataforma baseada em modelo Cliente/Servidor. Este projeto disponibiliza *hardware* e *software* para identificação automática das espécies de animais, com base nos registos áudio. A plataforma é composta por uma estação móvel, alimentada pelo sistema solar, que capta o som dos animais e envia para uma estação base, que por sua vez envia a gravação em tempo real para o servidor. O servidor é

		depois responsável pelo processamento áudio e a sua identificação. Esta plataforma ainda apresenta outras características, como permitir que o utilizador crie o seu modelo num ambiente *web*, para a identificação dos registos gravados, entre outras. Esta plataforma esta disponível *online* e é gratuita.
Wildlife Animal Soud Identification System (WASIS)	Aplicação em versão beta, técnica ainda desconhecida.	WASIS é o fruto de projeto de pesquisa da NavScales, desenvolvido pelos pesquisadores da Fonoteca Neotropical Jacques Vielliard da UNICAMP, em colaboração com os colegas de Sistema de Informação e do Laboratório de História Natural de Anfíbios Brasileiros. Esta aplicação tem como objetivo a identificação das espécies animais através da sua vocalização. Através da frequência e potência do sinal emitido pelos animais, como pássaros, sapos, insetos entre outros, a aplicação é capaz de distinguir as diversas espécies. Este *software* é gratuito.
Avisoft-SASLab Pro	• *Linear Predictive Coding(LPC)*	Avisoft é uma aplicação *desktop*, criada com o objetivo de facultar a investigação da comunicação acústica entre os animais. Também tem a capacidade de verificação da correlação entre dois registos áudios. O Avisoft é gratuito.
Sonobat	• *Analise zero-crossing*	Sonobat, é uma aplicação *desktop*, utilizada para a identificação automática de espécies de morcegos, através do som produzido pelos mesmos. Gravado o som, o utilizador carrega o ficheiro para a aplicação que depois faz a análise e identifica a espécie. Este *software* não é gratuito.

Na Tabela 3, apresenta-se um resumo de experiências com as metodologias e medidas seguidas na avaliação dos resultados encontradas na literatura relacionada com esta área de trabalho.

Tabela 3 - Lista de trabalhos e pesquisas realizadas no ramo da bioacústica.

Referências	**Medidas de Avaliação**	**Teste estatístico**	***Dataset***	**Metodologias/Hipóteses**
[Lopes, Kaestner,Silla e Koerich, 2011]	• *F-measure*	*Friedman* e *post-hoc Shafer*.	Dados de treino obtidos do site XenoCanto composto por 3 classes: • Taraba major (32 amostras) • Cercomacra tyrannina (34 amostras) • Thamnophilus oliatus (35 amostras).	Este trabalho pretende analisar/avaliar diferentes características do sinal áudio do canto de pássaros, para a identificação automática da espécie bem como diferentes algoritmos de aprendizagem máquina. • **Atributos**: *MARSYAS, IOIHC, SOUND* RULER; • **Algoritmos**: *Naive Bayes*, K-*nearest neighbors* (KNN), *Decision Trees J48, Multilayer Perceptron* (MLP), *Support Vector Machines* (SVM) (Polinomial) e SVM (*Pearson*). • 10-*fold cross-validation.*
[Aide, Corrada-Bravo,Campo-Cerqueira,Milan,Vegae Alvarez, 2013]	• *True positives(TP)* • *False Positives(FP)* • *True Negatives(TN)* • *False Negatives(FN)* • *Accuracy* • *Precision* • *Cross Validation* • Matriz de confusão	Não é utilizado nenhum teste estatístico.	Dados de treino constituído por 9 classes de diferentes tamanhos: 183, 395, 342, 407, 127, 231, 130, 190, e 163 amostras.	Este estudo tem com finalidade a demostração do uso da aplicação ARBIMON, determinando a precisão e exatidão na identificação das espécies. **Atributos**: Identificação de *regions of interest* (ROI) **Algoritmos:** Utilizam o *Hidden Markov Models* (HMMs) e o algoritmo Baum-Welch.
[GunaseKaram and Revathy, 2011]	• *Accuracy*	Para avaliação dos classificadores: KNN, MLP e KNN+MLP foi realizada a	Dados de treino composto por 6 classes.	Este trabalho tem como objetivo a descoberta de características acústicas a serem extraídas, de forma a se identificar com eficiência os animais

		comparação da média da *accuracy* obtida dos testes		selvagens através da vocalização. **Atributos**: foram testadas características espectrais, temporal e percetual, e ainda foi explorada para análise e seleção de características a redundância mínima, máxima relevância (mRMR). **Algoritmos**: *K-Nearest Neighbor* (KNN*),* MLP*,* e a junção dos dois. Foram realizados testes individuais para cada classificador, com uso de *minimal Redundancy-Maximal Relevance* (mRMR) e sem mRMR, e foi realizado o mesmo teste para a Fusão dos dois classificadores.
[Gelling, 2010]	• *Accuracy*	O *t-test* foi utilizado para avaliar resultados do mesmo modelo variando as configurações (*two sample t-test*) em conjunto com intervalos de confiança.	Dados de treino composto por 5 classes com 115, 61, 74, 160 e 154 amostras, respetivamente.	Este projeto tem como objetivo a avaliação dos modelos *Gaussian Mixtures Models* (GMMs) e *Hidden Markov Models* (HMMs) **Atributos**: MFCC´s. Os modelos são avaliados de forma independente (variado configuração) e em conjunto. **Algoritmos:** HMM e GMMs*.* Foram também usadas técnicas de *clustering.* *Cross-validation 10-fold*.
[Lee e Huang, 2006]	• *Accuracy*	Não é aplicado nenhum teste estatístico	São testadas 2 bases de dados com 420 e 561 sons de pássaros cada uma. Cada sinal áudio é dividido num conjunto de silabas	Este trabalho propõe um método capaz de identificar as espécies de pássaros pelo som. **Atributos**: baseados na média de MFCCs e *Linear Prediction Cepstral Coefficients* (LPCC)s, utilizando *Linear*

			e metade é usado como dados para treino e outra metade para teste.	*Discriminant Analyses* (LDA). **Algoritmos**:*Progressive Constructive Clustering* (PCC)
[Mitrovic, Zeppelzauer e Breiteneder, 2006]	• *Recall* • *precision*	Não é aplicado nenhum teste estatístico	A base de dados é constituída por 4 classes: • Pássaros: 99 Amostras • Gatos: 110 Amostras • Vacas: 90 Amostras • Cães: 84 Amostras	Este trabalho tem como objetivo estudar a aplicabilidade de diferentes características áudio e classificadores no domínio da bioacústica. **Atributos**:LPCCs, MFCCs e AD. *Lenght of Low Amplitude Sequence* (LOLAS) e *Area of High Amplitude* (AHA). **Algoritmos**: SVM, *Nearest Neighbor* (NN) e *Linear Vector Quantization* (LVQ). *Holdout*

2.5 Bases de dados

Com o objetivo científico de estudar e pesquisar a comunicação entre diversas espécies animais e a sua evolução, os ornitólogos juntamente com investigadores do ramo da biologia, dedicaram-se à construção de uma base de dados de sons de animais na década de 1970. A base de dados, denominada Fonoteca Neotropical Jacques Vielliard (FNJV), contou inicialmente com a gravação dos sons de aves abarcando posteriormente todos os grupos dos vertebrados e alguns invertebrados [Unicamp, 2015].

Jacques Marie Edme Vielliard, um dos ornitólogos muito conceituado, participou ativamente na construção desta base de dados, tendo atualmente a sua fonoteca (FNJV) atingido o patamar de uma das dez maiores do mundo, contando com mais de trinta mil vocalizações digitalizadas [Unicamp, 2015].

Atualmente existe um grande número de sons de animais gravados com vários fins científicos, o que torna morosa a identificação destes pela sua vocalização. São vários os investigadores, que têm dedicado a estudar uma forma de identificação automática de animais através dos sons que emitem [Unicamp, 2015] [Aide, Corrada-Bravo,Campos-Cerqueira,Milan,Vegae Alvarez, 2013] [Carvalho e Machado, 2011] [Chung, Oh, Lee, Park, Chang, Kim, 2013] [Conceição ,2015] [Gelling,2010] [GunaseKaram and Revathy, 2011] [Lee e Huang, 2006].

Como já foi referido, neste projeto pretende-se especificamente desenvolver uma plataforma que, através de registos áudio, seja capaz de identificar de forma automática as espécies dos pássaros aplicando as abordagens dos *motifs* e MFCCs para deteção de atributos.

Dada a grande quantidade de gravações de sons de animais (incluindo os dos passeriformes) existentes, são utilizados algoritmos de reconhecimento de padrões na tarefa de identificação do som. Estes algoritmos são normalmente utilizados para questões de comparação de grandes volumes e complexidade de informações [Lopes, Kaestner,Silla e Koerich, 2011].

Quando se trata de problemas relacionados com reconhecimento de padrões, a extração das características próprias da informação a ser analisada, torna-se uma tarefa muito importante para a obtenção de bons resultados.

Existem muitas técnicas e abordagens já anteriormente utilizadas que permitem extrair características dos registos áudio, que são depois utilizadas para a sua classificação. Na Tabela 3, referem-se alguns trabalhos onde são apresentadas diferentes abordagens. Assim, consideramos que o modelo de identificação automática das espécies animais através de registos áudio é tipicamente dividido em três etapas:

- Segmentação do registo áudio.
- Extração de características (atributos).
- Classificação.

3 Classificação de sons

Neste capítulo apresentamos as principais etapas da classificação de sons. Na subseção 3.1 são descritas algumas das técnicas de pré-processamento de sons. Na subseção 3.2, são apresentadas as duas abordagens para extração de atributos utilizadas neste trabalho. Na subseção 3.3 abordamos o *data minig* e descrevem-se os algoritmos de classificação utilizados neste trabalho.

3.1 Pré-processamento de sons

O primeiro passo no reconhecimento áudio é o pré-processamento dos sons de forma a prepara-los para a extração das suas características, que serão depois usadas para a classificação [Petry, Zanus e Barone,1999] [Carvalho e Machado, 2011]. Este processo geralmente inclui passos como a eliminação da componente contínua do sinal, normalização do sinal e a aplicação de filtros [Carvalho e Machado, 2011].

Na seção 3.1.1 é descrita uma destas técnicas a ser usada neste trabalho.

3.1.1 Envelope

O cálculo do envelope de um sinal consiste na variação de uma das características desse mesmo sinal ao longo tempo, como a amplitude. Esta técnica tem como objetivo a obtenção de uma sequência numérica (sinal discreto) de um sinal contínuo no tempo e no espaço, aplicando um filtro passa-baixo de forma a atenuar as altas frequências e se obter as frequências percetíveis ao ouvido humano, ou seja, baixas frequências [Marques, 2013]. O grau de atenuação de altas frequências irá depender do filtro passa-baixo utilizado e a sua ordem. O envelope de energia de Shannon é muito utilizado na segmentação para o reconhecimento de sons acústicos [Liang, 1997] [Gomes and Batista, 2015b] [Oliveira et al., 2014] [Sharma, Saha e Kumari, 2014].

O envelope de Shannon contém a sequência numérica de um sinal contínuo no tempo e no espaço com base no cálculo da enegia de Shannon. O sinal é dividido igualmente em pequenas janelas de intervalo de tempo, de tal forma que o período de cada janela seja menor que metade da sua frequência e a seguir é calculada a energia de Shannon aplicando a equação 1 para cada janela [Sharma, Saha e Kumari, 2014] [Oliveira et al., 2014] [Kumar, 2006].

$E=-x^2 * \log x^2$	[1]

Depois de calcular o envelope de energia de Shannon, procede-se à sua normalização calculando o valor médio de energia de Shannon (com base na equação 2) ao longo de uma janela contínua obtendo o efeito ilustrado na Figura 3 [Sharma, Saha e Kumari, 2014] [Kumar, 2006].

$En = \frac{E-u}{\sigma}$	[2]

Sendo:

- En – Média de energia de Shannon normalizada.
- E - Energia de Shannon.
- u – Valor médio da energia E do sinal calculado com base na equação 3.
- σ – Desvio padrão da energia E do sinal.

$\mu = -\frac{1}{n}\sum_{i=1}^{n} x_i^2 \log(x_i^2)$	[3]

- n – Número de janelas normalizadas.

Na Figura 3, apresenta-se um envelope de energia de Shannon aplicado a um sinal áudio cardíaco (fonocardiograma).

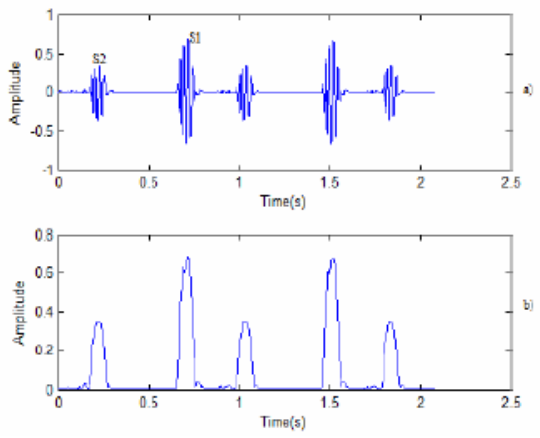


Figura 3 - Exemplo de um envelope de energia de Shannon [Sharma, Saha e Kumari, 2014].

O gráfico a) representa um sinal contínuo no tempo na sua forma original e o gráfico b) mostra o envelope deste sinal com base na energia de Shannon normalizada.

3.2 Técnicas de extração de atributos

Os registos áudio dos sons emitidos pelos pássaros contêm uma grande quantidade de informação. Assim, é muito importante identificar com critério as informações mais relevantes e suficientes, ou seja, os atributos que contêm informação importante para identificar de forma discriminatória uma espécie [Mendes, 2011].

O processo de extração de atributos consiste na caraterização de um segmento áudio e a representação numérica deste. Existem várias caraterísticas que podem ser utilizadas para caraterizar um sinal áudio [GunaseKaram and Revathy, 2011]

Nas subseções seguintes descrevemos as técnicas de extração de atributos usadas neste trabalho.

3.2.1 MFCC

Os *Mel Frequency Cepstral Coefficients* (MFCC) são um conjunto de atributos extraídos de um sinal áudio contínuo no tempo, tendo em base as frequências percetíveis à audição humana [Lyons, b]. Os coeficientes são calculados com base na escala mel (*mel scale*), obtidos através equação 4, que expressa a obtenção da *mel-scale* através das frequências obtidas do sinal áudio, com o objetivo de dar menos importância a altas frequências que são menos suscetíveis ao ouvido humano [Lyons, b] [Stevens, Volkmann e Newman, 1937].

$$M(f) = 1125 \ln(1 + f/700)$$	[4]

Esta técnica de extração de atributos foi introduzida por Davis e Mermelsteins na década de 1980 e é amplamente usada no Reconhecimento Automático de Fala (ASR) e em sons acústicos [Lyons, b] [Davis e Mermelsteins,1980].

Na Figura 4, é ilustrada o procedimento para a extração de coeficientes MFC, os MFCCs.

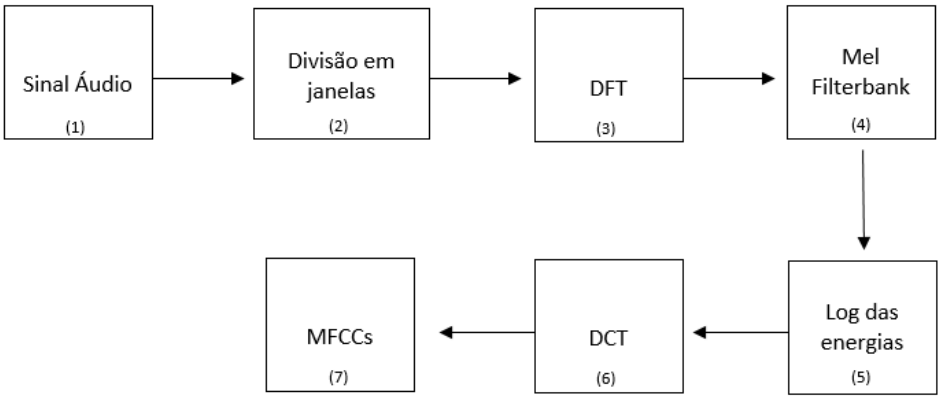


Figura 4 - Computação dos coeficientes MFC [Chung, Oh, Lee, Park, Chang, Kim, 2013].

Funcionamento:

- (1) – Entrada do sinal do áudio.
- (2) – Divisão do sinal em pequenos *frames.*
- (3) – Para cada *frame* é calculada a Transformada Discreta de Fourier.
- (4) - É aplicado o *Mel Filterbank* para os espectros de energia.
- (5) – Calcular o logaritmo de todas as energias obtidas.
- (6) – Calcular a transformada discreta do cosseno dos logaritmos.
- (7) – Saída dos coeficientes MFC.

3.2.2 Motif

Um *motif* num sinal áudio é um padrão que é mais frequentemente observado, isto é, um padrão que se repete numa série temporal [Gomes e Batista, 2015]. Na Figura 5, é apresentado um exemplo da repetição de *motifs* numa série temporal.

Os *motifs* em uma série temporal podem ser extraídos a partir de determinados métodos ou ferramentas. O *Multiresolution Motif Discovery in Time* Series (MrMotif) é uma destas ferramentas, desenvolvida em linguagem Java e disponibilizada pelos autores [Castro e Azevedo,2010].

Esta abordagem tem a sua aplicabilidade em diversas áreas como a medicina [Gomes et a.l,2013], metereóloga [McGovern et al., 2007], videovigilância [Hamid et al.,2005], captura de movimentos [Minnen et al., 2006], entre outras.

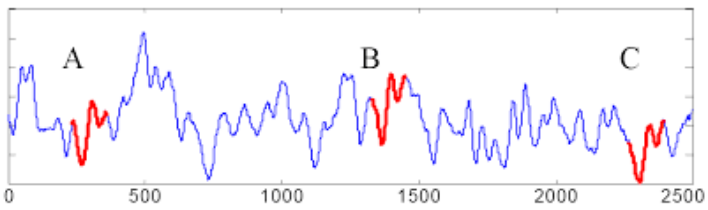


Figura 5 - *Motifs* numa série temporal [Lin et al.,2002].

3.3 Data mining

Data mining, ou extração de conhecimento de dados, refere-se ao tratamento de um grande volume de dados e na aplicação de algoritmos específicos para a extração de informações úteis e não triviais. *Data mining* é um domínio interdisciplinar que envolve várias áreas como a estatística, base de dados e aprendizagem automática [Bhargava e Sharma, 2013].

Dado um conjunto de indivíduos descritos por um conjunto de variáveis existentes, são aplicadas diversas tarefas de *data mining* que serão descritas de seguida de uma forma sucinta [Bhargava e Sharma, 2013]:

- Procura de associações (*association mining*) - o objetivo é encontrar padrões que descrevem relações entre as variáveis.
- *Clustering* - tem como objetivo descobrir grupos de indivíduos com características semelhantes.
- Classificação - nesta tarefa cada individuo tem uma classe (valores discretos) e o objetivo é encontrar padrões que permitem determinar a classe de um novo individuo dada a sua descrição.
- Regressão - esta tarefa é idêntica à classificação, mas em vez de uma classe discreta pretende-se determinar um valor de uma escala continua.

Neste capítulo serão apresentadas as técnicas e algoritmos de *data mining* utilizados neste trabalho.

Na subsecção 3.3.1 são apresentados os algoritmos de classificação e o método de avaliação dos modelos.

3.3.1 Algoritmos de classificação

Os algoritmos de classificação ou algoritmos de aprendizagem supervisionada são algoritmos que implementam uma forma de aprendizagem análoga à dos seres humanos, ou seja, estes algoritmos são capazes de aprender com base nas experiências anteriores [Liu, 2007]. No caso dos computadores, a aprendizagem faz-se a partir de dados etiquetados e caracterizados por atributos. As etiquetas correspondem a classes**.**

Os algoritmos de classificação têm como objetivo obter um modelo de classificação capaz de prever a que classe pertencerá uma nova entrada, com base nos seus atributos e os dados já etiquetados ou dados de treino [Liu, 2007].

Nas secções seguintes serão descritos os seguintes algoritmos de classificação utilizados para este trabalho, *Decision Trees* (J48*), Radom Forest* e *Support Vector Machine,* uma vez que são os mais frequentes na bibliografia, apresentando bons resultados.

3.3.1.1 Decision Trees (J48)

A árvore de decisão (*decision trees*) é um algoritmo onde a árvore é usada como um grafo acíclico de decisão. Cada nó de decisão ou nó interno representa uma questão sobre um determinado atributo e os nós folha afetos a este nó de decisão representam uma classe, ou seja, representam a resposta ao nó de decisão, como ilustrado na Figura 6 [Bhargava e Sharma, 2013][Bing Liu, 2007]. Esta técnica resume-se na divisão de um problema complexo em problemas mais simples, aplicando a teoria de dividir para conquistar. As árvores de decisão são usadas para elaborar uma função de classificação capaz de obter um valor com base nas características (atributos) dos dados já etiquetados, dadas as caraterísticas extraídas do sinal de entrada.

Este algoritmo é vastamente utilizado com sucesso em áreas como *data mining* e reconhecimento de padrões [Bhargava e Sharma, 2013] [Han and Kamber, 2006].

A árvore de decisão carrega fundamentalmente 3 tipos de informações, ilustradas na Figura 6:

- Nó raiz
- Nó Folha – representa uma classe.
- Nó interno - representa as condições de teste aplicado a um atributo.

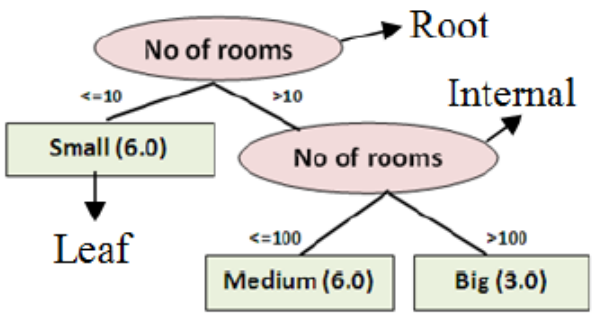


Figura 6 - Árvore de Decisão [Bhargava e Sharma, 2013].

O J48, escrito em Java é uma implementação *open source* do algoritmo C4.5 desenvolvido por Ross Quinlan [Quinlan, 1993] [Hall et al., 2009]. Este algoritmo, tem como objetivo a criação de árvores de decisão com base no critério de ganho de informação, critério este que seleciona o atributo preditivo a ser usado em cada nó da árvore, onde o atributo com maior informação encontra-se na raiz da árvore e os descendentes com menor, seguindo a ordem decrescente [Quinlan, 1993][Gangajot e Chhabra , 2014]. O critério de ganho de informação utiliza a entropia como medida de impureza, onde é comparada o grau de entropia do nó raiz com o grau de entropia dos nós descendentes, sendo escolhido como nó de divisão o nó que apresente o atributo que mais maximiza o ganho de informação [Bing Liu, 2007].

A entropia é calculada com base no número de instâncias de um *dataset* pertencentes a uma determinada classe, ou seja, a entropia tende a zero se a percentagem de instâncias pertencentes a uma única classe tenderem a 100 [Bing Liu, 2007].

A construção de uma árvore de decisão com base no algoritmo J48 segue os seguintes passos, para um dado conjunto *C* [Bhargava e Sharma, 2013] [Gangajot e Chhabra , 2014] [Batista, 2015]:

- Calcular a folha, que é resultado de todos os casos pertencentes a mesma classe e etiqueta-la com essa classe.
- Para cada atributo *A*, calcular a informação e ganho de informação.
- Identificar o atributo com maior ganho de informação e criar um nó com esse atributo, *A*. Para um determinado atributo com n valores A_i, o ganho de informação é calculado com base na equação 5.
- Dividir o Conjunto *C*, em vários subconjuntos $C_1, C_2, C_3 \ldots C_n$, consoante o número de valores A_i obtidos do atributo *A* acima identificado.
- Repetir o processo para todos os subconjuntos $C_1, C_2, C_3 \ldots C_n$.

$G(C, A) = E(C) - \sum_{i=1}^{n} \Pr(A_i) E(C_i)$	[5]

U
n

de *E*(*C*) é a entropia do conjunto *C* calculada com base na equação 6, Pr(*Ai*) a probabilidade de ocorrer *Ai* para *A* no conjunto *C* e *E*(*Ci*) a entropia dos subconjuntos *Ci*, onde *A* tem valores *Ai*.

$\sum_{i=1}^{n} - \Pr(CL_i) * \log_2 \Pr(CL_i)$	[6]

Na equação 6, Pr(*CLi*) representa a probabilidade da classe *CLi* estar no conjunto *C* e *n* é o número de Classes.

3.3.1.2 Random Forest

Random Forest é um algoritmo de classificação em que o resultado da classificação de uma determinada entrada *x*, consiste na combinação dos resultados de classificação de várias árvores de decisão para esta mesma entrada, onde a classe escolhida é a que possuir o voto maioritário. As árvores de decisão são fruto de distribuição idêntica, independente e aleatória das instâncias pertencentes aos dados de treino [Breiman, 2001] [Ho, 1995] [Oshiro, 2013].

As amostras aleatórias (*bootstrap*) de conjuntos de treino são produzidas com base no algoritmo de aprendizagem (*ensemble*) para cada árvore aleatória [Breiman, 1996] [Efron,

1979]. As amostras *bootstraps* são calculadas com base na técnica de reposição, onde a partir de um conjunto de treino são criados outros subconjuntos aleatoriamente.

A árvore é criada com base no novo subconjunto e a seleção aleatória de atributos, o que minimiza a correlação aumentado o tamanho da floresta, fazendo com que a taxa de acerto da classificação seja maior. Como ilustrado na Figura 7, para cada nó da árvore são selecionados e avaliados aleatoriamente *m* atributos de um conjunto, sendo o melhor atributo o critério para a divisão do nó [Oshiro, 2007].

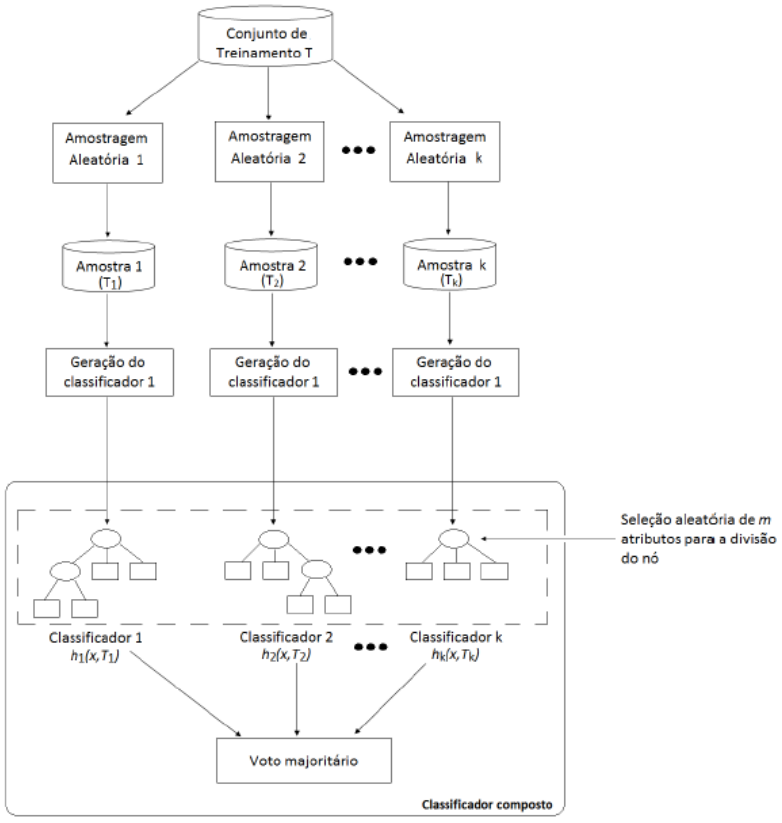


Figura 7 - Random Forest [Oshiro, 2007].

Segundo Breiman, o uso do algoritmo de aprendizagem (*ensemble*) conhecido por método *bagging* em *Random Forests* melhora o desempenho deste, desde que os atributos sejam aleatoriamente selecionados [Breiman , 2001].

Independentemente do número de árvores aleatórias geradas, este algoritmo apresenta melhor taxa de acerto em comparação com outros algoritmos e não existe sobre ajustamento, como se pode constatar em diversas experiências [Breiman, 2001].

O erro de classificação em *Random Forests* depende de dois fatores:

- Correlação entre as árvores geradas – quanto menor for, maior é a taxa de acerto.
- Força individual de cada árvore – quanto maior for, maior é a taxa de acerto.

3.3.1.3 Support Vector Machine

O *Support Vector Machine* (SVM) é um algoritmo de classificação onde os dados de treino se encontram etiquetados, introduzido na década de 1963 por Bernhard E. Boser, Isabelle M. Guyon, e Vladimir N. Vapnik [Vapnik, 1963].

Existem várias extensões deste algoritmo, tais como SVM com margens rígidas, SVM com margens lineares e SVM não lineares.

O SVM classifica os dados de teste, dispondo os dados de treino como pontos no espaço. Neste trabalho, será utilizado o SVM com margens rígidas, onde para um conjunto de amostras *Xi* pertencentes a um conjunto específico de dados de treino inicial *X* com $X_i \in X$ existem Y classes etiquetadas.

O SVM com margens rígidas classifica uma entrada *Z* de dados de teste, para um conjunto de dados de treino *X* pertencentes a duas classes *Y*1=+1 e *Y*2= -1 diferentes e linearmente separáveis da seguinte forma:

- Distribui os dados de treino *Xi* no espaço consoante os seus atributos.
- Encontra um hiperplano capaz de separar os dados de treino *Xi* com maior margem possível, ou seja, com maior distância entre o hiperplano e os dados de treino, de forma que os dados pertencentes a *Y*1 se encontrem num lado do hiperplano e os pertencentes a *Y*2 do lado oposto como ilustrado na Figura 8. O hiperplano é calculado com base na expressão apresentada na equação 7.
- Classifica a entrada *Z* consoante a sua localização no espaço referente a que lado do hiperplano pertence. A localização do *Z* no espaço irá depender igualmente dos seus atributos e é calculada com base na expressão apresentada na equação 8, com *Z*=*x*.

$f(x) = w \cdot x + b = 0$	[7]

Onde *W* representa o vetor que determina a orientação do plano, *b*, o desvio do plano e *x* o vetor que represente o espaço de exemplos.

$g(\mathrm{x}) = \mathrm{sgn}(f(\mathrm{x})) = \begin{cases} +1 \text{ se } \mathrm{w}\cdot\mathrm{x}+b>0 \\ -1 \text{ se } \mathrm{w}\cdot\mathrm{x}+b<0 \end{cases}$	[8]

Em que a função *f*(*x*) = +1 identifica a classe *A* e a função *f*(*x*) = -1 identificada a classe *B* por exemplo.

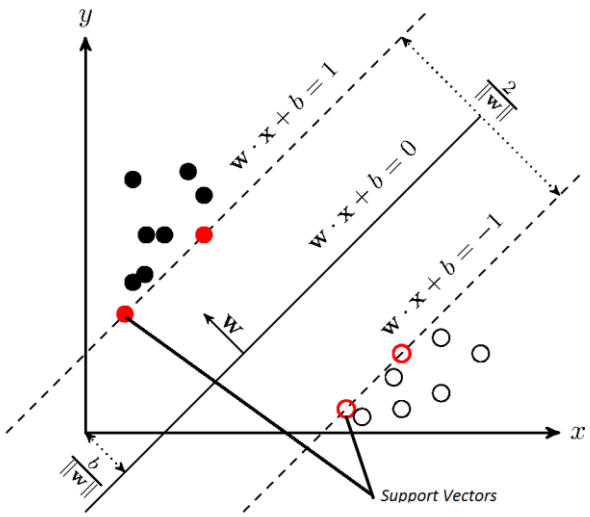


Figura 8 - Exemplo de Support Vector com duas classes [Guru].

Este exemplo funciona porque as classes são linearmente separáveis. Em casos reais existem outros fatores como o ruido que podem condicionar a disposição dos pontos [Gama et al, 2012].

A plataforma Weka permite-nos implementar o SVM através do algoritmo *Sequencial Minimal Optimization* (SMO). Este ainda permite transformar os atributos nominais em binário e por omissão normaliza-os [Hall et al., 2009].

4 Ferramentas e Tecnologias utilizadas

Neste capítulo, apresentamos as ferramentas e tecnologias utilizadas para a construção da solução do trabalho que aqui se descreve.

Neste trabalho é utilizada a linguagem de programação Java para o desenvolvimento da aplicação servidor (*web*) e da aplicação cliente (*Android*). Do lado da aplicação servidor é utilizada a API Weka, implementada em Java, que agrega diversos algoritmos de aprendizagem para o tratamento de diversas tarefas do *data mining* [Witten, 2005]. Do lado da aplicação cliente, é utilizada a linguagem de programação Java (*Android*). Esta escolha deve-se ao fato dos *smartphones* com o sistema operativo *Android* terem um maior domínio sobre o mercado móvel [IDC] o que permitirá alcançar o maior número de utilizadores.

4.1 Java

A ferramenta Java, lançada em 1995 pela Sun Microsystems, é uma plataforma computacional e uma linguagem de programação orientada ao objeto [Oracle, a], tornando-se um *software* livre em 13 de fevereiro de 2006 sob os termos da *General Public Licence (GNU)* [Oracle, c]. Em 2010, a Sun Microsystems foi adquirida pela Oracle e desde então tem estado a trabalhar de forma a construir sistemas totalmente integrados e soluções otimizadas com grandes níveis de desempenho. Esta plataforma foi projetada para ser compatível com o maior número possível de plataformas computacionais com um custo baixo em relação a propriedade das aplicações tanto do lado de quem as desenvolve com a de quem as compra [Oracle, b].

4.2 NetBeans IDE

O NetBeans IDE é o IDE oficial do Java que permite facilmente e de forma rápida desenvolver aplicações Java para ambientes móvel, *web e desktop.* O NetBeans, para além de suportar a linguagem Java, também suporta outras linguagens de programação tais como HTML5, JavaScript, CSS, PHP e C/C++. É uma plataforma de programação gratuita e de código aberto, usada por um grande número de utilizadores [NetBeans].

4.3 Android Studio

Até a data 26 de Maio de 2015, as aplicações para o sistema operativo *Android* eram desenvolvidas através da plataforma Eclipse *Android Developer Tools* (ADT), mas esta deixou de receber suporte [Android Developers, b]. Assim, o *Android Studio* baseado na plataforma

de desenvolvimento IntelliJ IDEA, passou a ser utilizado como a ferramenta oficial para o desenvolvimento de aplicações para o sistema operativo móvel *Android* [Android Developers, a].

4.4 Weka

O Weka é um *software* inovador na história de *data mining* e *machine learning*, desenvolvido pela Universidade WaiKato, em nova Zelândia. O *software* está implementado em linguagem Java, é gratuito, de código aberto e é multiplataforma [Gangajot e Chhabra , 2014] [Witten, 2005]. O Weka disponibiliza ferramentas que permitem a visualização e pré-processamento de dados, seleção de atributos, classificação, predição, regressão, *clustering,* regras de associação e avaliação de modelos [Gangajot e Chhabra , 2014] [Hall, et al., 2009]. As funcionalidades do Weka tanto podem ser utilizadas por invocação de código Java como através da sua interface de utilizador.

5 Construção da Solução

Este capítulo está dividido em 2 subseções.

Na subsecção 5.1 apresenta-se a análise e modelação da solução.

Na subsecção 5.2 descreve-se a implementação da solução.

5.1 Análise e Modelação da Solução

A solução proposta para este projeto está constituída por 3 componentes:

- Aplicação *web* (*Web Services*).
- Aplicação móvel (*Android*).
- Aplicação *desktop* (*Backoffice*)

Aplicação *Web* – É a componente onde se encontram alojados todos os serviços necessários para a identificação automática das espécies de pássaros.

Aplicação Móvel – É a componente responsável por gravar os sons dos pássaros e enviar para a aplicação *web* para que seja feita a sua identificação. Através desta aplicação "*BirdSound Identification*" também é possível que o utilizador dê o seu *feedback* acerca da identificação realizada.

Aplicação *Desktop* – Esta componente tem a função de gestão, controlo e teste dos serviços disponibilizados pela aplicação *web*.

Na seção 5.1.1 apresenta-se os requisitos não funcionais

Na seção 5.1.2 apresenta-se os requisitos funcionais.

Na seção 5.1.2 apresenta-se o desenho da aplicação *web* (*Web Services*).

Na seção 5.1.3 descreve-se a arquitetura da solução.

5.1.1 Requisitos não funcionais.

Os requisitos funcionais são de elevada importância uma vez que espelham a qualidade das funcionalidades disponibilizadas pela solução.

Nesta solução foram considerados os seguintes requisitos não funcionais:

Facilidade de Uso: A *interface* da aplicação móvel é uma *interface* intuitiva e de fácil uso, onde todas as funcionalidades são disponibilizadas através do ecrã inicial.

Integração: Os serviços da aplicação *web* são disponibilizados através do *web service* do tipo "*Big Services*" onde as mensagens se encontram no formato *EXtensible Markup Language* (XML) seguindo o padrão *Simple Object Access Protocol* (SOAP), o que permite a integração com outros sistemas por se tratar de uma linguagem interoperável.

Portabilidade: A aplicação móvel está preparada parar ser executada em qualquer plataforma *Android* com as versões compreendidas entre a versão *Android* 4.4 (*KitKat*) e a versão *Android* 6.0 (*MarshMallow*).

Implementação: A solução está implementada em linguagem Java.

Padrões: Foi aplicado o conceito de programação orientada a objeto em todos os componentes da solução.

Confiabilidade: Pretende-se que a aplicação *web* seja alojada num servidor de acesso público, para que os serviços possam ser acedidos em qualquer lugar e a qualquer momento.

Disponibilidade: A aplicação móvel permite que o utilizador realize a tarefa de identificação em qualquer lugar e a qualquer momento desde que haja uma ligação a *internet*, separando esta a tarefa da tarefa de gravação mantendo os dados da gravação salvaguardados.

5.1.2 Requisitos funcionais

Os requisitos funcionais estão relacionados com as funcionalidades que a solução disponibiliza aos utilizadores. A solução é composta por duas componentes de interação com os utilizadores. A primeira componente é a aplicação móvel e a segunda é a aplicação *desktop*.

5.1.2.1 Aplicação móvel

A aplicação móvel "*BirdSound Identification"* apresenta as seguintes funcionalidades:

- Gravar sons.
- Identificar o som gravado.
- Dar *feedback*.
- Gerir os sons.

Desta forma, é apresentado através da Figura 9 o diagrama de casos de uso para a aplicação móvel.

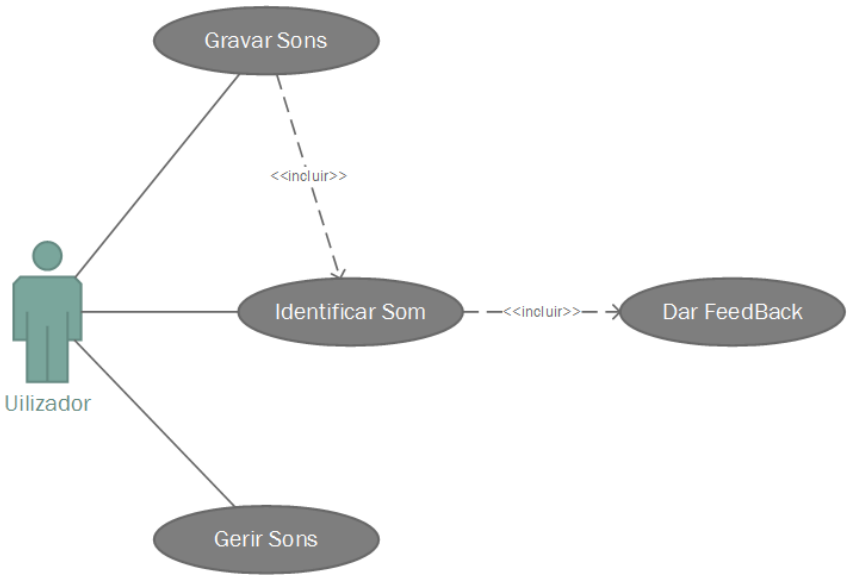


Figura 9 - Diagrama de casos de uso (Aplicação Móvel).

Para cada caso de uso será apresentado um diagrama de sequência de forma a permitir uma melhor perceção dos cenários e de seguida será apresentado o diagrama de classes e o diagrama de base de dados da aplicação.

5.1.2.1.1 **Caso de uso "Gravar Sons".**

A funcionalidade de gravar sons, está disponível a partir da aplicação móvel, onde para gravar os sons, clica-se no botão "*Record*" e para parar a gravação clica-se no botão "*Stop*". Após a gravação, o som é imediatamente guardado na pasta reservada para sons gravados e de seguida são recolhidos os detalhes de gravação e guardados na base de dado local. As anotações são recolhidas através de um módulo da aplicação que usa as funcionalidades *Global Positioning System* (GPS). As anotações serão as seguintes: Longitude, Latitude, Data, Hora e o Endereço/Local da gravação.

A seguir, são apresentadas duas opções, efetuar a identificação e não efetuar a gravação.

Após o som ser enviado para identificação e depois de receber o resultado, o utilizador poderá dar o *feedback* sobre o resultado recebido. Quando o servidor envia o resultado, do lado do cliente é-lhe perguntado se o resultado esta "Correto" ou "Incorreto". Depois de enviar o feedback, esta informação será registada do lado da aplicação *web (Web Services)*.

Caso o utilizador não pretenda efetuar a identificação, o cenário de "gravar som" termina e o som é disponibilizado através de uma lista onde a sua identificação poderá ser realizada posteriormente.

O diagrama de sequência para este caso de uso é apresentado através da Figura 10 .

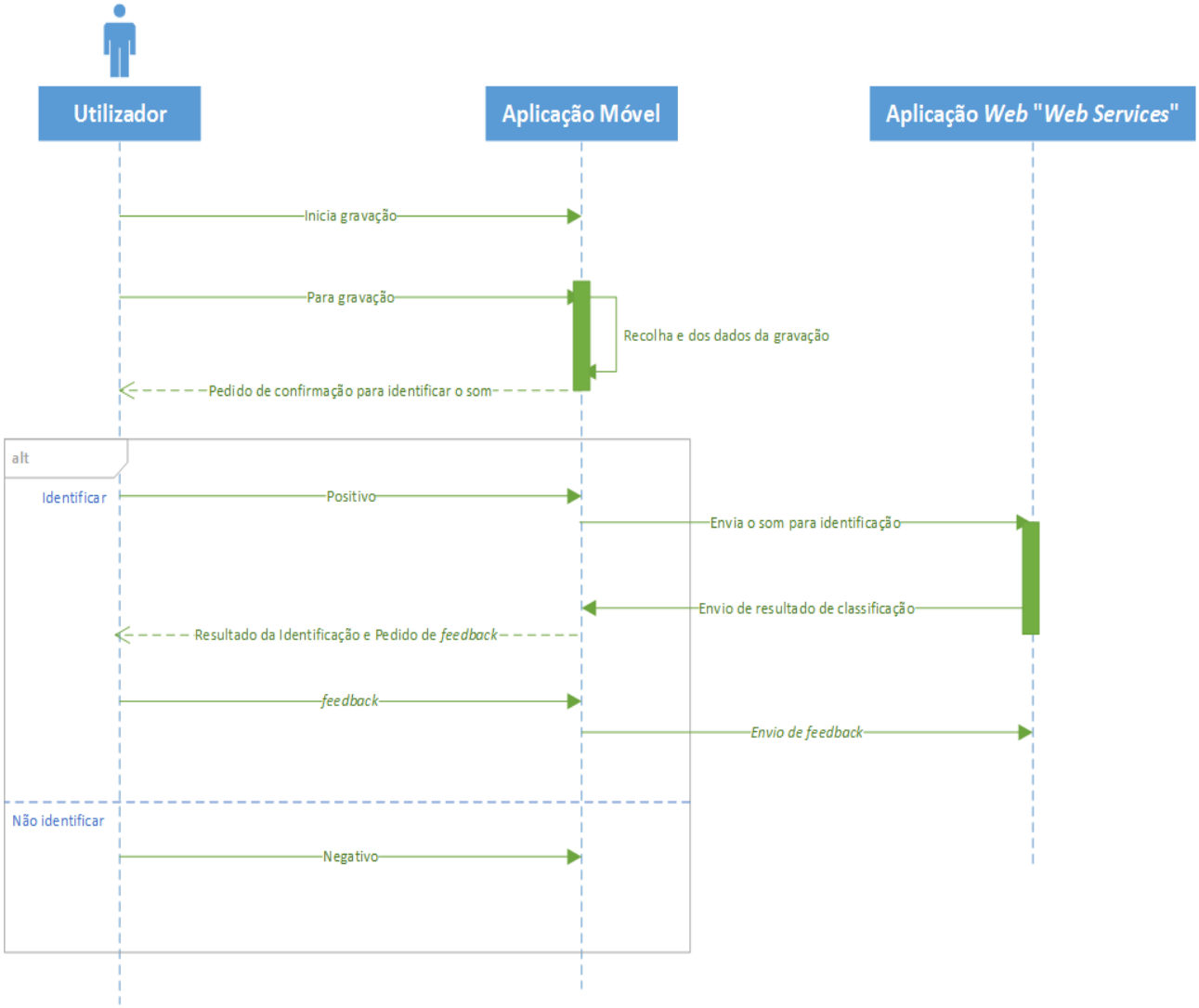


Figura 10 – Diagrama de sequência “Gravar Som".

5.1.2.1.2 **Caso de uso “Identificar o som”**

Esta funcionalidade, é acedida pelo utilizador quando este executa a gravação do som e/ou através do *MenuPopUp* afeto a cada som que se encontra na lista de sons gravados. A partir desta funcionalidade, o som é enviado para a aplicação *web* que por sua vez identifica o som. Feita a identificação do som, é enviado o resultado ao utilizador e de seguida é-lhe pedido o *feedback* da identificação.

De seguida é ilustrado o diagrama de sequência deste caso de uso através da Figura 11.

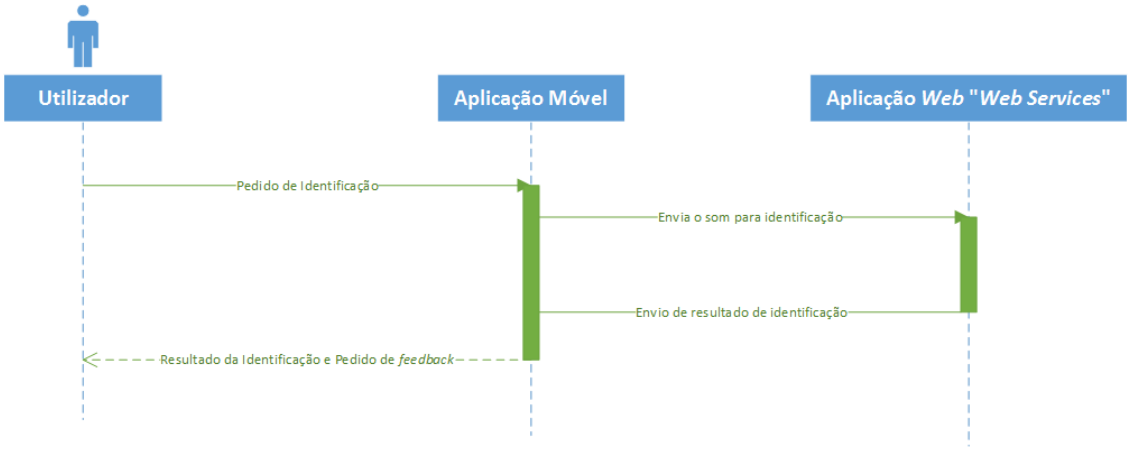


Figura 11 - Diagrama de sequência “Identificar Som”.

5.1.2.1.3 **Caso de uso “Dar *feedback*”**

Esta funcionalidade é executada a cada vez que é efetuada uma identificação. Juntamente com o resultado da identificação é pedido um *feedback* ao utilizador.

Todos os *feedbacks*, bem como os sons recolhidos e os seus detalhes, são gravados numa base de dados do lado da aplicação *web* para uma futura gestão.

Na Figura 12 é apresentado o diagrama de sequência para este caso de uso.

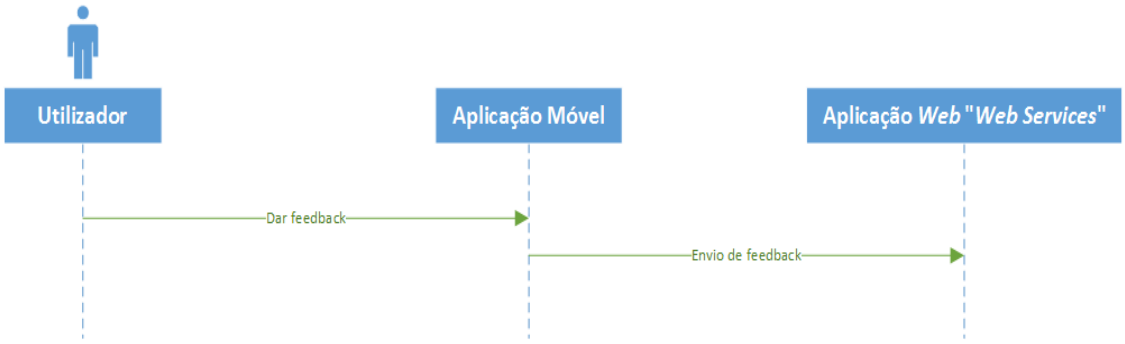


Figura 12 – Diagrama de Sequência “Dar *feedback*”.

5.1.2.1.4 **Caso de uso “Gerir sons”**

A partir deste caso de uso, o utilizador tem acesso a duas opções:

- Apagar som.
- Ver Detalhes.

A gestão do som é feita graças ao *MenuPopUp* afeto a cada som que se encontra na lista. Clicando no menu, o utilizador tem a opção de ver os detalhes do som como de o eliminar.

Na Figura 13, é apresentado o diagrama de sequência deste caso de uso.

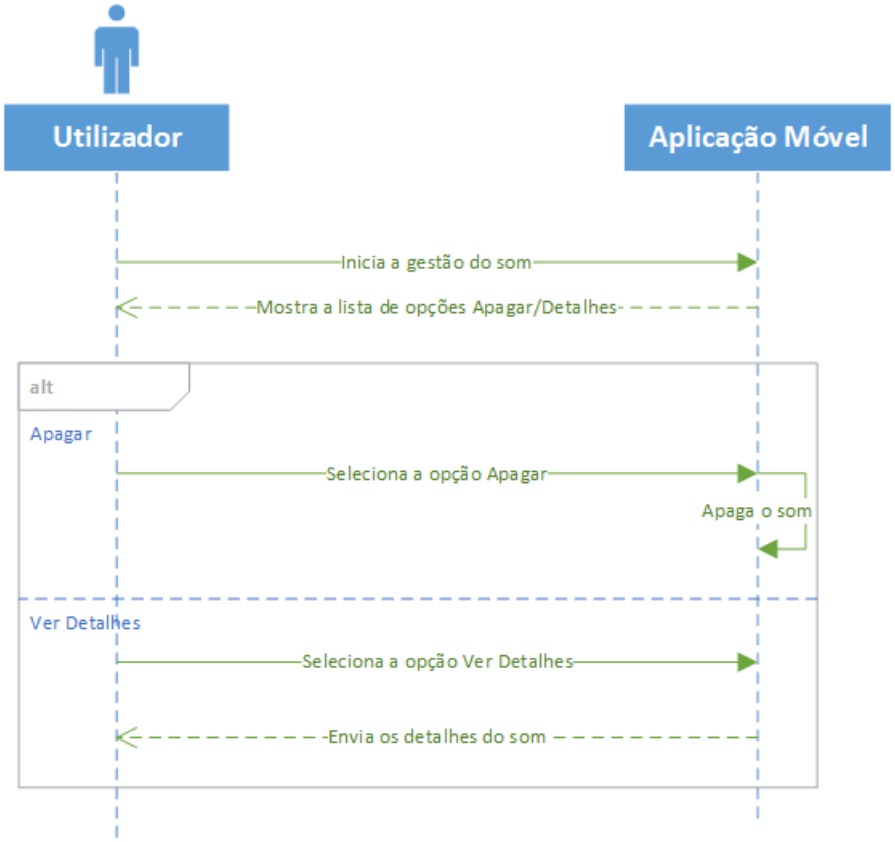


Figura 13 – Diagrama de sequência "Gerir sons".

5.1.2.1.5 **Diagrama de classes da aplicação móvel.**

Na Figura 14, é apresentado o diagrama de classes da aplicação móvel "*Android*". Esta aplicação está constituída por 8 classes e faz recurso a duas bibliotecas, nomeadamente *WaveWriter* e *WaveReader:*

- ***"MainActivity"****:* É a única atividade da aplicação onde a *interface* principal está associada.
- ***"BirdAudio"****:* É a classe que representa o registo áudio.
- ***"MyAlert"****:* É a classe que representa o alerta em caso da ligação de dados/*WIFI* ou ligação GPS estarem indisponíveis.

- ***"BirdSoundIdentificationServer"****:* É a classe que representa a ligação com aplicação *web* "*Web Services"*. Nesta classe se encontram todos os parâmetros de configuração bem com os métodos que permitem chamar as operações do serviço disponibilizado.
- ***"DBHELPER":*** Classe responsável pela criação e gestão da base de dados.
- ***"Signal_Processing":*** É a classe responsável pelo pré-processamento do som gravado.
- ***"My_Files_Adapter":*** É a classe que representa o ficheiro na lista de ficheiros.
- ***"My_Files"****:* É classe que herda as características da *ListActivity* e representa a lista dos ficheiros.
- ***"WaveWriter":*** Biblioteca responsável por converter o ficheiro áudio gravado no formato RAW para o formato WAV.
- ***"WaveReader":*** Biblioteca responsável por ler os ficheiros WAV e converte-los em dados do tipo "*double".*

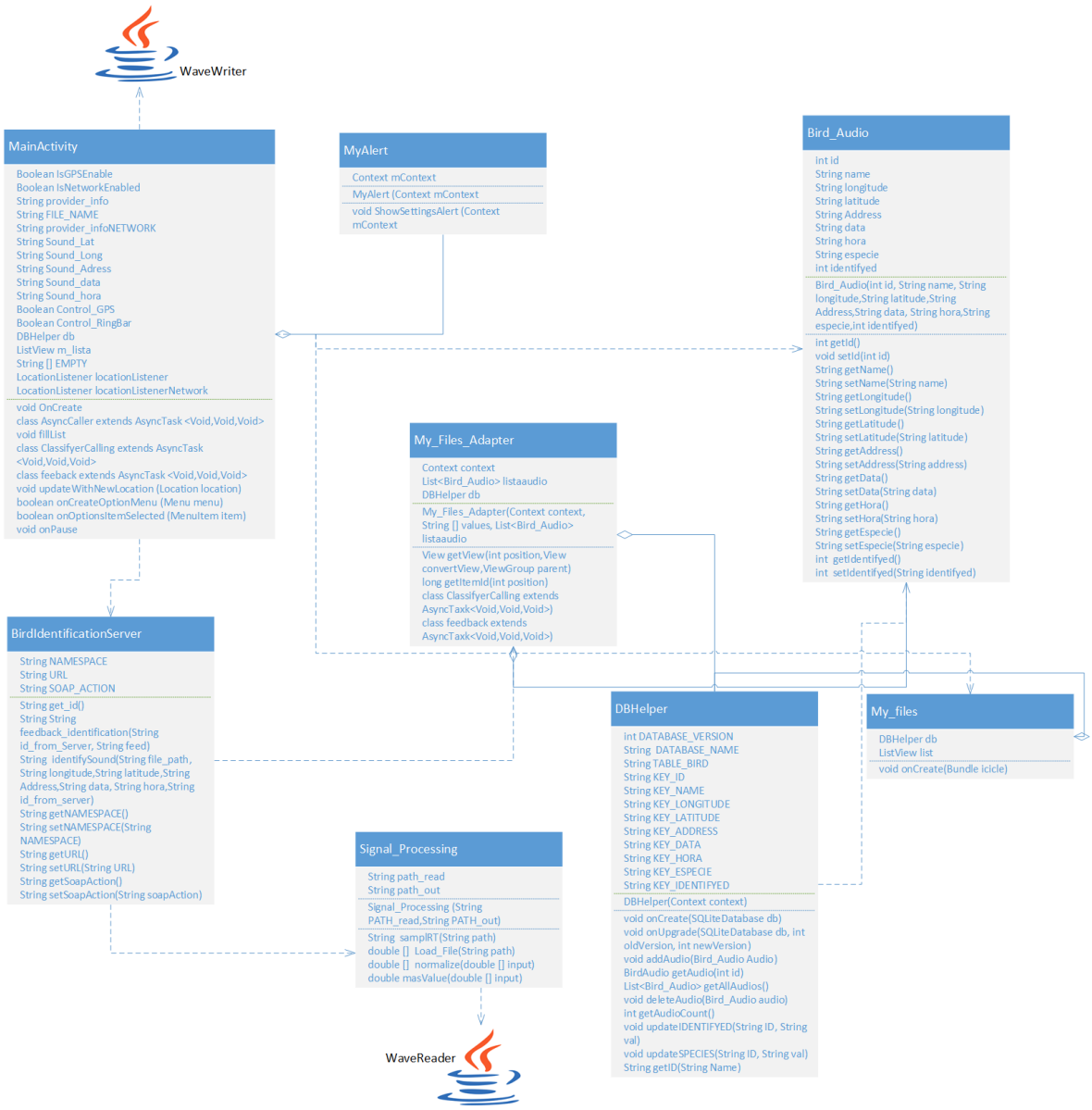


Figura 14 – Diagrama de classes da aplicação móvel "*Android*".

5.1.2.1.6 **Diagrama de base de dados.**

A base de dados "*Bird_Identification_dat*" da aplicação móvel contém apenas uma tabela onde serão guardados o nome do ficheiro e as anotações da gravação, como se pode verificar na Figura 15.

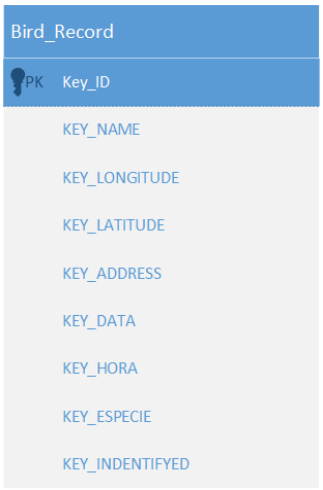


Figura 15 – Diagrama de base de dados da aplicação móvel.

A tabela "*Bird_Record"* está constituída por 9 colunas:

- ***KEY_ID***: Nesta coluna é guardado o id do ficheiro.
- ***KEY_*NAME**: Nesta coluna é guardado o nome do ficheiro.
- ***KEY_LONGITUDE*:** Campo onde é guardado a Longitude onde foi realizada a gravação do som.
- ***KEY_LATITUDE***: Campo onde é guardado a Latitude de onde foi realizada a gravação do som.
- ***KEY_ADDRESS***: Campo onde é guardado o endereço de onde foi realizada a gravação do som.
- ***KEY_DATA***: Campo onde é guardada a data da gravação do som.
- ***KEY_HORA***: Campo onde é guardada a hora da gravação do som.
- ***KEY_ESPECIE***: campo onde é guardado o resultado da identificação do som.
- ***KEY_IDENTIFYED***: o campo onde é guardado o valor que indica se o som identificado recebeu um *feedback* positivo ou negativo.

5.1.2.2 Aplicação *Desktop* (BackOffice)

A aplicação *desktop* tem como o objetivo a gestão, controlo e teste da aplicação *web*. Assim, esta componente apresenta as seguintes funcionalidades:

- Identificar som.
- Carregar dados de treino.
- Testar o modelo classificação.
- Iniciar *Web Services*.
- Parar *Web Services.*

- Reiniciar *Web Services.*
- Recriar a base de dados.

Através da Figura 16, é apresentado o diagrama de casos de uso para a aplicação *desktop (Backoffice).*

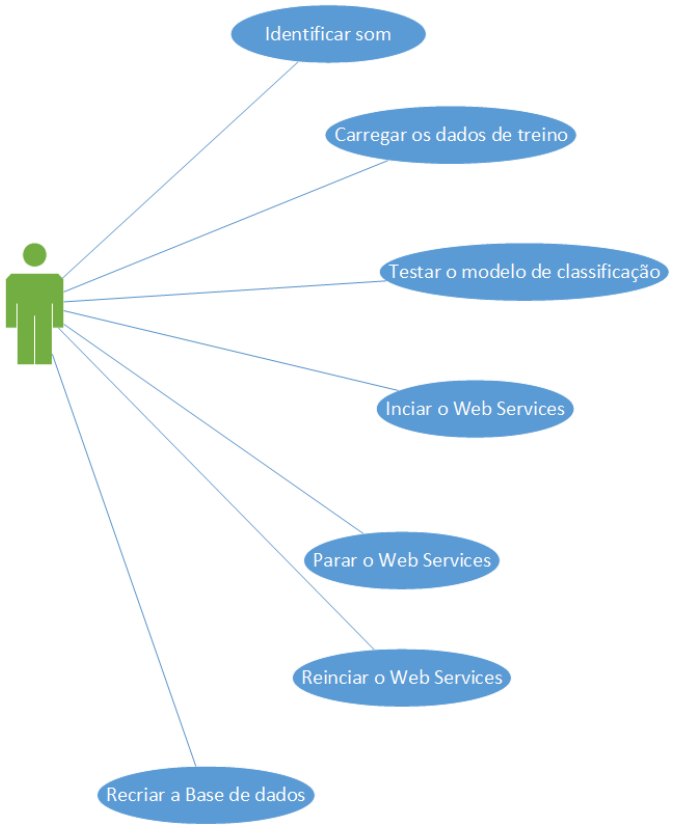


Figura 16 – Diagrama de Casos de uso "Aplicação *Desktop*".

5.1.2.2.1 **Caso de uso "Identificar som"**

Esta funcionalidade permite efetuar a identificação de um determinado som de pássaro gravado através de qualquer fonte, usando este *software* de gestão.

Clicando no botão identificar, o ficheiro áudio é enviado para o servidor e é feita a sua identificação e devolvido o resultado da identificação. O diagrama de sequência apresentado na Figura 17 permite-nos ter uma melhor visão do cenário.

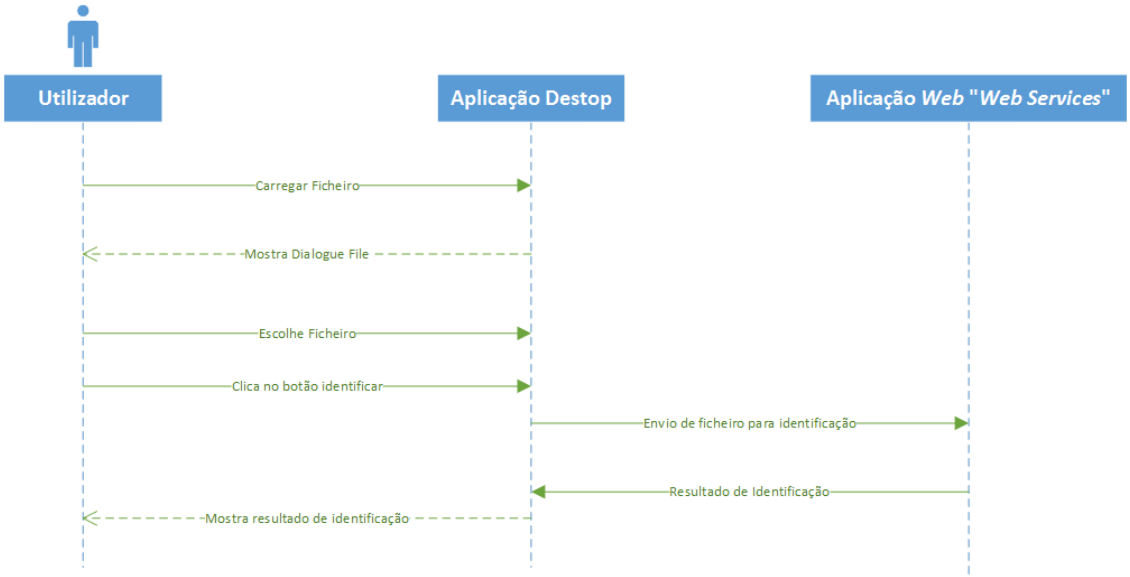


Figura 17 – Diagrama de sequência “Identificar som”.

5.1.2.2.2 **Caso de uso “Carregar dados de treino”.**

Esta funcionalidade permite que o utilizador carregue novos dados de treino para a base de dados. O diagrama de sequência deste caso de uso é ilustrado através da Figura 18.

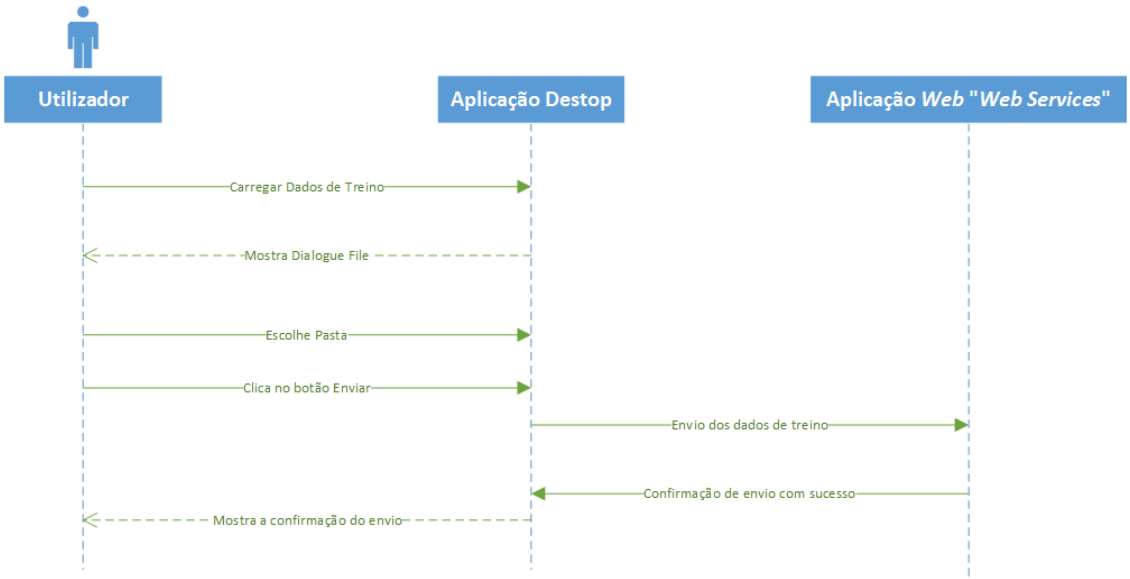


Figura 18 – Diagrama de sequência “Carregar dados de treino”

5.1.2.2.3 **Caso de uso "Testar modelo de classificação"**

Esta funcionalidade permite ao utilizador testar o modelo de classificação, recebendo como resultado diversas medidas de desempenho da solução. Na Figura 19, é ilustrado o diagrama de sequência para este caso de uso.

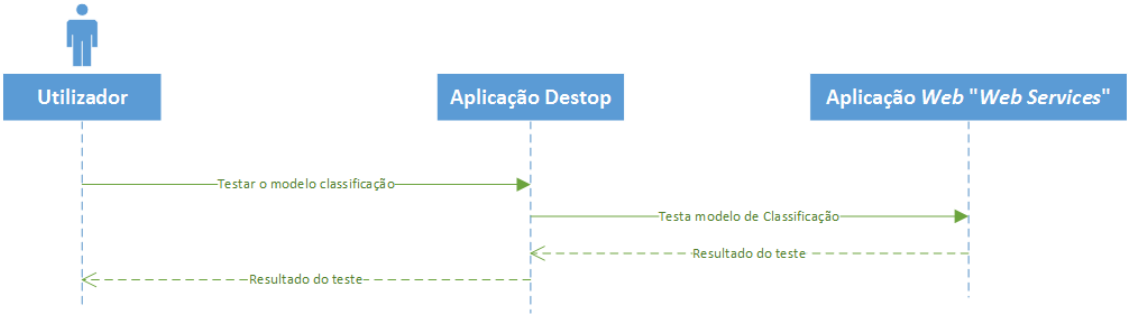


Figura 19 – Diagrama de sequência "Testar modelo de classificação".

5.1.2.2.4 **Caso de uso "Iniciar *Web Services*"**

Esta funcionalidade permite que o utilizador inicie os serviços, tornando possível que estes sejam executados com sucesso. Na Figura 20 é ilustrado o diagrama de sequência para este caso de uso.

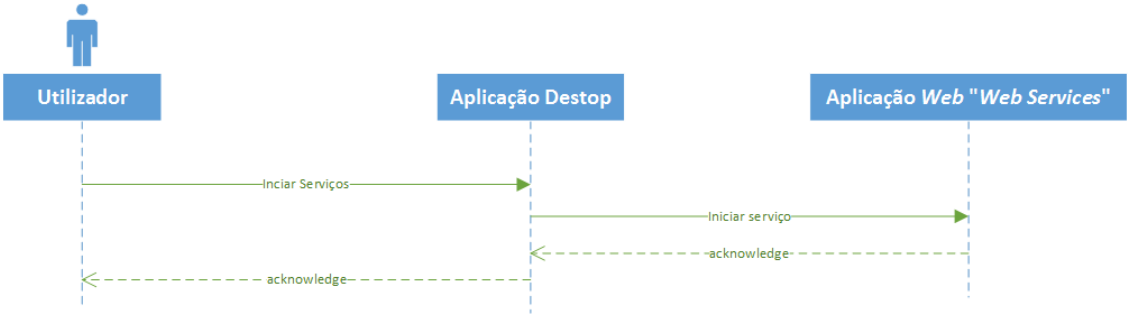


Figura 20 – Diagrama de sequência "Iniciar *web services*".

5.1.2.2.5 **Caso de uso "Parar *Web Services*"**

Esta funcionalidade permite que o utilizador pare os serviços, impossibilitando-os de ser executados com sucesso. Na Figura 21, é ilustrado o diagrama de sequência para este caso de uso.

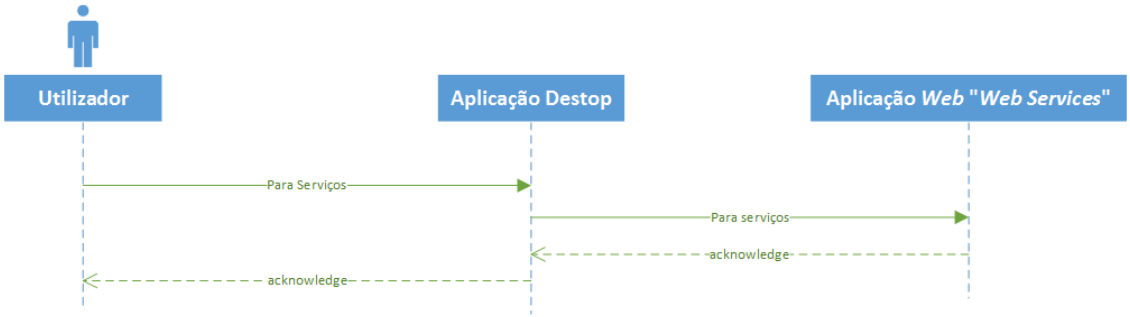


Figura 21 - Diagrama de sequência "Parar *web services*".

5.1.2.2.6 **Caso de uso "Reiniciar *Web Services*".**

Esta funcionalidade permite que o utilizador reinicie os serviços. No caso de serem introduzidos novos dados de treino a partir desta funcionalidade, é possível fazer o *refresh* do sistema. Na Figura 22, é ilustrado o diagrama de sequência para este caso de uso.

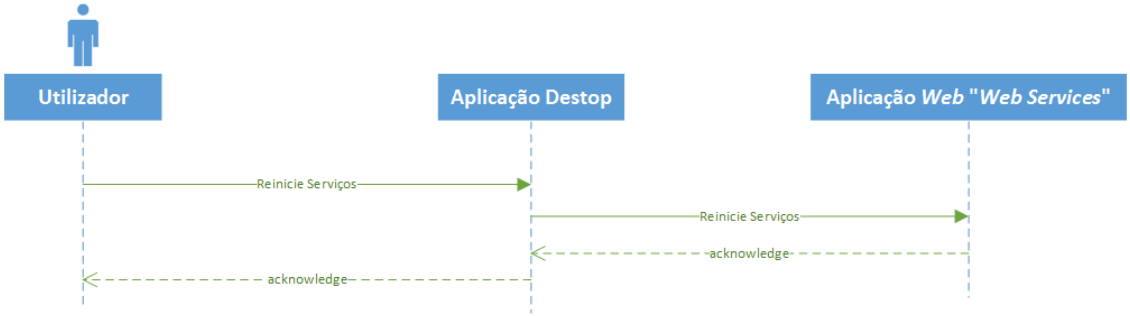


Figura 22 - Diagrama de sequência "Reiniciar *web services*".

5.1.2.2.7 **Caso de uso "Recriar a base de dados"**

Esta funcionalidade permite que a base de dados "SQL" seja recriada em caso do utilizador pretender apagar todos os dados de treino. Na Figura 23 é ilustrado o diagrama de sequência para este caso de uso.

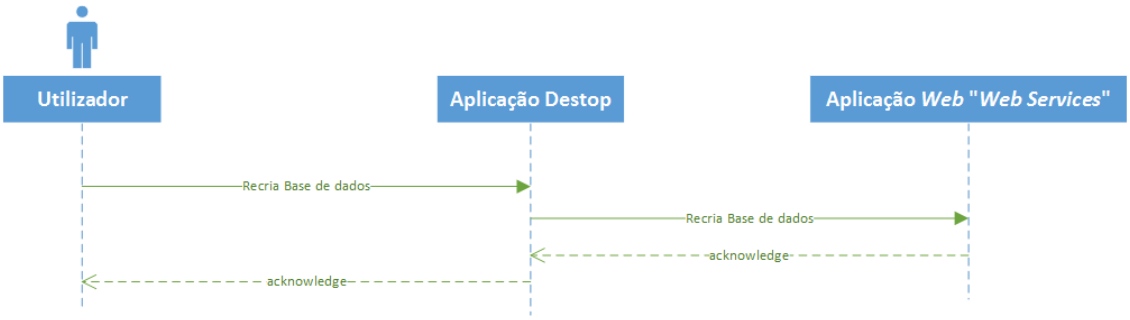


Figura 23 - Diagrama de sequência “Recriar a base de dados”.

5.1.2.2.8 **Diagrama de classes da aplicação *desktop*.**

Esta aplicação é constituída por 3 classes fundamentais:

- **“*BirdSound_Identification*”**: É a classe onde se encontram disponibilizadas todas as funcionalidades da aplicação *desktop* por meio de uma *interface* intuitiva, com exceção da funcionalidade “identificar som”.
- **“Identify_Sound”**: É a classe onde se encontra disponibilizada a funcionalidade “Identificar som” por meio de uma *interface.*
- **“Signal_Processing”**: Esta classe é constituída por um conjunto de métodos necessários que permitem realizar o pré-processamento do ficheiro áudio e faz o uso da biblioteca “WaveReader” para realizar determinadas tarefas.

Na Figura 24 é ilustrado o diagrama de classes desta aplicação.

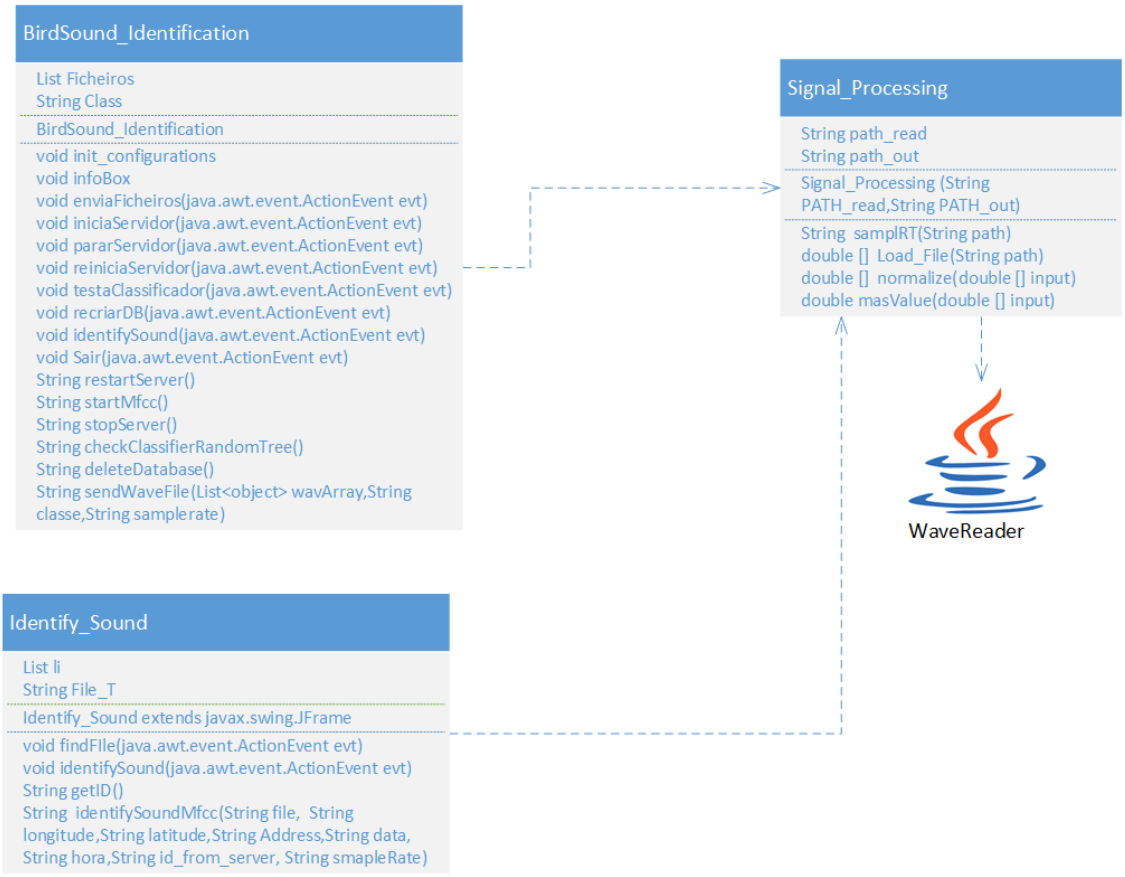


Figura 24 – Diagrama de classes da aplicação *desktop*.

5.1.3 Desenho da Aplicação *web* "*Web Services*"

A aplicação *web* está constituída por um serviço *web* que contém 9 operações, que são descritas na Tabela 4.

Tabela 4 - Serviço *BirdSoundManagement_Service*.

Serviço *WEB*	BirdSoundManagement_Service
Operações	1. Enviar Ficheiros
	2. Testar o classificador
	3. Receber ID
	4.Identificar o som
	5.Arrancar o Servidor
	6.Parar o Servidor
	7.Reiniciar o Servidor
	8. Dar o *feedback*

	9.Reconstruir a Tabela de dados de Treino

Estas operações são chamadas pela aplicação móvel (*Android*) e a aplicação *desktop* (*Backoffice*) a cada vez que é executada uma funcionalidade. No entanto é necessária a criação de uma base de dados onde são guardadas os dados de treino e os registos a serem identificados. Esta base de dados é acedida através da aplicação *web* que será responsável pela gestão destes dados.

Do lado da aplicação *web* são executadas as operações *Create, Read,Update* e *Delete* (CRUD) sobre a base de dados.

De seguida, é ilustrado o diagrama de classes da aplicação *web* "*Web Services*" através da Figura 25 e o diagrama de base de dados através da Figura 26.

- **Diagrama de classes da aplicação *web* "*Web Service*".**

A aplicação *web* está constituída por 3 classes principais e 2 bibliotecas, como ilustrado na Figura 25.

"***BirdSoundManagement_Service"*:** É a classe que representa o serviço *web.* Nela se encontram todas as operações que permitem a execução dos requisitos funcionais.

"***Control"*:** É a classe que contém um conjunto de métodos relacionados com as operações à base de dados e operações auxiliares.

"***Process_MFCC"*:** Esta classe é responsável pelas operações relacionadas com a extração de atributos usando a abordagem dos MFCCs.

"***MFCC":*** É uma biblioteca constituída por um conjunto de operações que permitem a extração dos atributos, seguindo a abordagem dos MFCCs.

*"**WEKA":*** É a biblioteca que fornece um conjunto de algoritmos de aprendizagem para a realização de tarefas relacionadas com o *data mining*.

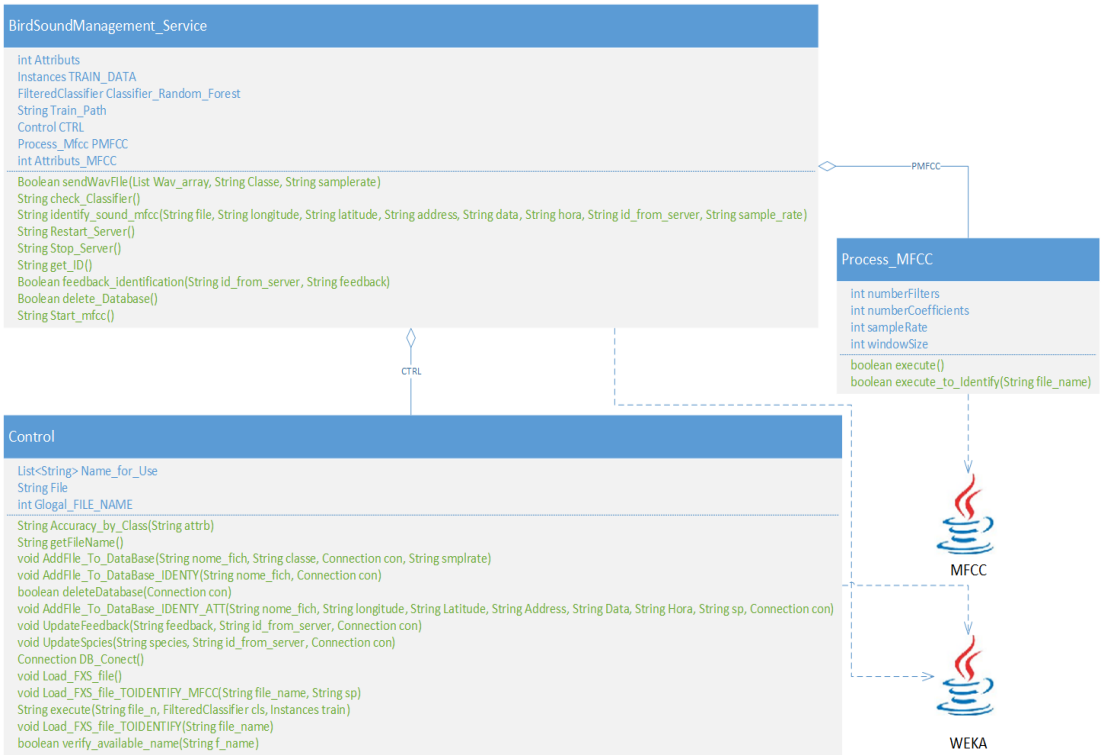


Figura 25 – Diagrama de classes da aplicação *web*.

- **Diagrama de base de dados**

A base de dados é constituída apenas por duas tabelas. A tabela TB_TRAIN_DATA é a tabela onde são guardados os dados de treino, sendo constituída por 4 colunas:

- **File_name**: Coluna onde é guardado o nome do ficheiro. O Nome do ficheiro é gerado com base num contador inicializado em 1000, com o seguinte formato: wavS_(contador).
- **Classe:** Coluna onde é guardado o nome da espécie a que se refere o registo áudio.
- **sp:** Coluna onde é guardado o *sample rate* do registo áudio.
- **File_number:** O número do registo áudio.

A tabela TB_IDENTY é a tabela onde são guardados todos os sons a serem identificados. Está constituída por nove colunas:

- **File_name:** Coluna onde é guardado o nome do ficheiro. O Nome do ficheiro é igual ao identificador UUID recebido para a identificação.
- **Longitude:** Campo onde é guardado a longitude onde foi realizada a gravação do som.
- **Latitude:** Campo onde é guardado a Latitude onde foi realizada a gravação do som.
- ***Address*:** Campo onde é guardado o endereço onde foi realizada a gravação do som.

- **Data:** Campo onde é guardada a data da gravação do som.
- **Hora:** Campo onde é guardada a hora da gravação do som.
- **Espécie:** campo onde é guardado o resultado da identificação do som.
- ***Feedback*:** o campo onde é guardado o *feedback* da identificação.
- **sp:** Coluna onde é guardado o *sample rate* do som.

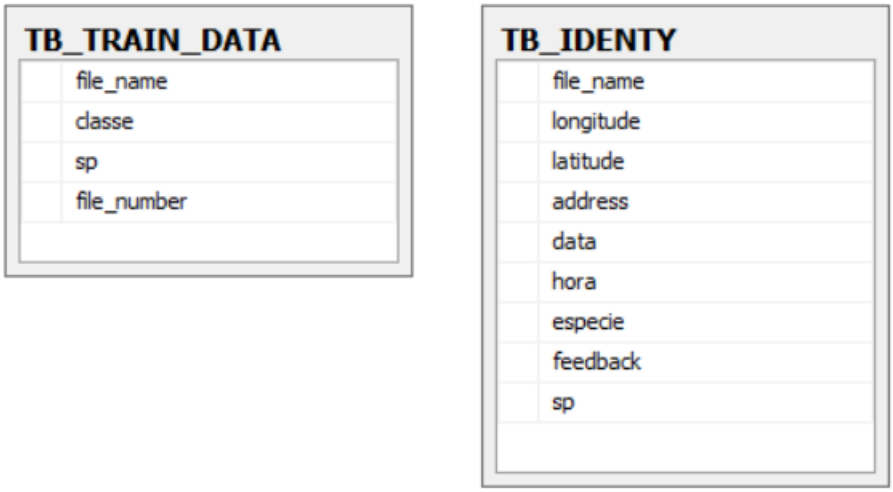


Figura 26 – Diagrama de base de dados da aplicação *web*.

5.1.4 Arquitetura da Solução

Como já foi referido, este projeto de tese tem como objetivo a criação de uma plataforma composta por 3 componentes para a recolha e identificação automática de espécies de pássaros utilizando registos áudio.

Para a comunicação entre as componentes é utilizado o *web Service* do tipo "*Big*" *web services* em que as mensagens estão no formato *EXtensible Markup Language* (XML) seguindo o padrão *Simple Object Access Protocol* (SOAP) e, *como* meio de transmissão, utiliza o protocolo *Hypertext Transfer Protocol* (HTTP) [java EE 6]. Na Figura 27, apresentam-se os componentes da solução e as respetivas ligações.

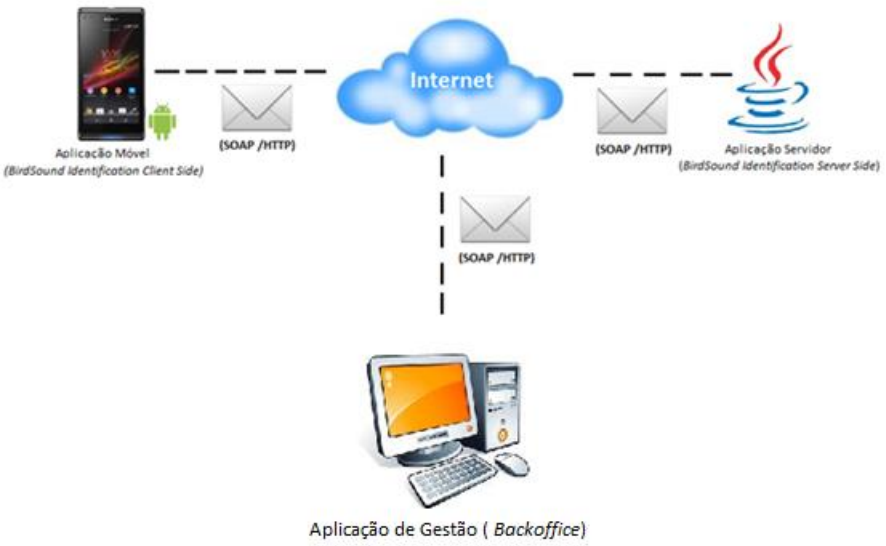


Figura 27 - Arquitetura da solução.

Do lado da aplicação servidor, os serviços "*Service Contract*" são disponibilizados através de *Uniform Resource Locator* (URL) permitindo que a aplicação móvel e a aplicação *desktop* comuniquem com a aplicação *web* e façam o uso dos seus serviços.

5.2 Implementação da Solução

Este capítulo contém a descrição da implementação da solução que é formada por 3 componentes. Assim esta seção é composta por 3 subseções:

A subseção 5.2.1, onde se descreve a implementação da componente Aplicação *web* (*Web Services*).

A subseção 5.2.2, onde se descreve a implementação da componente Aplicação *desktop* (*Backoffice*).

A subseção 5.2.3, onde se descreve a implementação da componente Aplicação móvel (*Android*).

5.2.1 Aplicação *web* (*Web Services*)

A componente aplicação *web* foi desenvolvida no NetBeans IDE, em Java. A aplicação *web* é constituída por um serviço *web* "*BirdSoundManagement_Service*", onde se encontram disponibilizadas todas as operações de gestão e classificação (ver Figura 28).

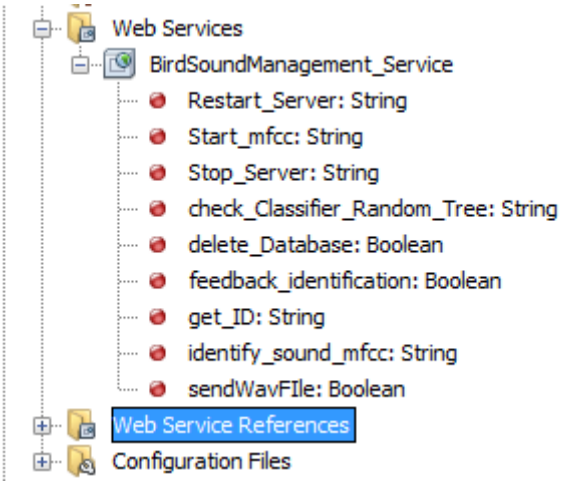


Figura 28 - Operações do serviço *BirdSoundManagement_Service*.

De seguida, são descritas as operações disponibilizadas pelo serviço *web* "*BirdSoundManagement_Service*".

- **Operações do "BirdSoundManagement_Service":**

1. Operação "*sendWavFIle*".

Esta operação permite que a aplicação *desktop* (*Backoffice)* carregue os dados de treino.
Recebe como parâmetros o registo áudio já processado sob formato de uma lista de "*double*", o nome da classe a que pertence o registo e o *sample rate* do som, como ilustrado na Figura 29.

Esta operação é chamada para cada ficheiro áudio a ser enviado. Após receber o ficheiro sob o formato de lista de "*double*" é criado um ficheiro do tipo "CSV" com todos os valores em "*double*" recebidos e guardado numa pasta. De seguida, é guardado o nome do ficheiro, a classe e o *sample rate* do mesmo na base de dados SQL na tabela "TB_TRAIN_DATA".

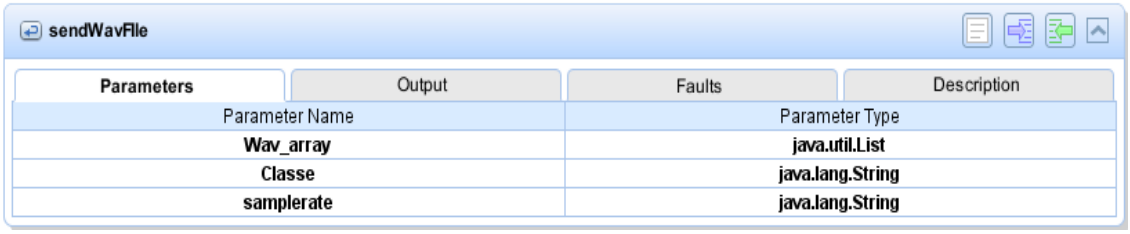


Figura 29 – Operação "*sendWavFIle*".

2. Operação "*check_Classifier_Random_Tree*".

Esta operação permite que, do lado da aplicação *desktop* (*Backoffice)*, seja possível testar o modelo de classificação obtido com o algoritmo *Random Forest* e os dados de treino, de forma a se obter informações acerca da taxa de acerto e outras medidas de avaliação.

Como se pode observar na Figura 30, esta operação não recebe quaisquer parâmetros. Após ser chamada, é aplicado o método de validação cruzada (*cross validation*) com 10 subconjuntos (*folds*) sob os dados de treino e devolvido o resultado do teste em formato de "*string"* como ilustrado no excerto de código apresentado na Figura 31.

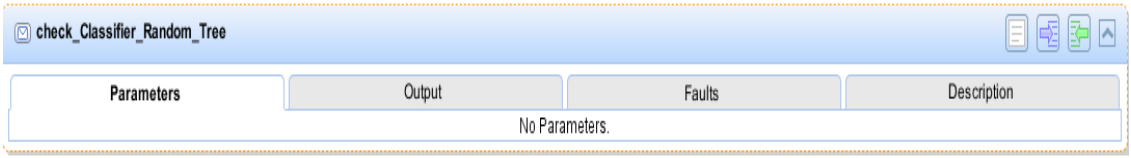


Figura 30 – Operação "*check_Classifier_Random_Tree*".

```
public static String Accuracy_by_Class(String attrb) throws IOException, Exception
{

    if(new File(attrb).exists())
    {
        String inFile= attrb;
       CSVLoader loader = new CSVLoader();
       loader.setSource(new File(inFile));
       Instances train = loader.getDataSet();
       train.setClassIndex(train.numAttributes() - 1);
       RandomForest cls = new RandomForest();
       cls.setNumTrees(200);
       cls.setNumFeatures(0);
       cls.setSeed(1);
       cls.setMaxDepth(0);
       cls.buildClassifier(train);
       Evaluation eval = new Evaluation(train);
       eval.crossValidateModel(cls, train, 10, new Random(1));
       return eval.toSummaryString("\nResults\n======\n", false) + "\n" + eval.toClassDetailsString();

    }
    else
    {
        return "Attributs file is missing";
    }
}
```

Figura 31 – Excerto de código da operação "*check_Classifier_Random_Tree*".

3. Operação "*get_ID*"

Nesta operação, é gerado aleatoriamente um identificador único (UUID) e enviado como resposta (como se pode ver na Figura 32). Este valor é posteriormente usado para realizar a identificação do som, sendo utilizado como identificador único dos registos áudios a serem identificados e depois utilizados para realizar o *feedback*.

```
@WebMethod(operationName = "get_ID")
public String get_ID() {
    //TODO write your implementation code here:
    String file_n = UUID.randomUUID().toString();
    file_n = file_n.replaceAll("-", "_");
    try {
        while(!CTRL.verify_available_name(file_n))
        {
            file_n = UUID.randomUUID().toString();
            file_n = file_n.replaceAll("-", "_");
        }
    } catch (SQLException ex) {
        Logger.getLogger(BirdSoundManagement_Service.class.getName()).log(Level.SEVERE, null, ex);
    }


    try {
        CTRL.AddFIle_To_DataBase_IDENTY(file_n,CTRL.DB_Conect());
    } catch (SQLException ex) {
        Logger.getLogger(BirdSoundManagement_Service.class.getName()).log(Level.SEVERE, null, ex);
    }

    return file_n;
}
```

Figura 32 – Código da operação "*get_ID*".

4. Operação "*identify_sound_mfcc*".

Esta operação permite que a aplicação móvel *(Android*) e a aplicação *desktop* (*Backoffice)* identifiquem/classifiquem o registo áudio. Recebe como parâmetros o registo áudio em formato "*string*", os atributos da gravação nomeadamente: data, hora, longitude, latitude, endereço-completo: Rua, País, cidade, Código-postal, o "id de identificação" retornado pela operação "get_ID" que deverá ser chamada antes desta e por fim o *sample rate* do som, que é utilizado na extração do MFCC. Na Figura 33 é ilustrado o desenho desta operação.

identify_sound_mfcc

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
file		java.lang.String	
longitude		java.lang.String	
latitude		java.lang.String	
address		java.lang.String	
data		java.lang.String	
hora		java.lang.String	
id_from_server		java.lang.String	
sample_rate		java.lang.String	

Figura 33 – Operação "*identify_sound_mfcc*".

Após receber o ficheiro sob o formato de um vetor de *"string"* é criado um ficheiro do tipo *Comma-Separated Values* (CSV) com o nome igual ao valor "id_from_server" (Figura 33) e guardados os valores do vetor. As respetivas anotações e o nome do ficheiro são guardados na base de dados SQL na tabela "TB_IDENTY". De seguida é extraído os atributos MFCC e feita a classificação.

5. Operação "*Start_mfcc*".

Esta operação permite iniciar o servidor. Sem esta, as outras operações não funcionarão. A operação "*Start_Server*" é chamada pela aplicação *desktop* (*Backoffice*) com o objetivo de construir o modelo de classificação. São processados os coeficientes MFCC dos dados de treino que se encontram na base de dados através do método "*PMFCC.execute*" e é construído o modelo como ilustrado na Figura 34.

```
@WebMethod(operationName = "Start_mfcc")
public String Start_mfcc() {

    try
    {
        CTRL.Load_FXS_file();
    }
    catch(Exception e)
    {
    }
    try {
        if(PMFCC.execute())
        {
            CSVLoader loader = new CSVLoader();
            loader.setSource(new File(Train_Path));
            Instances train = loader.getDataSet();
           train.setClassIndex(train.numAttributes() - 1);
            RandomForest  _Random_Forest = new RandomForest();
            _Random_Forest.setNumTrees(200);
           _Random_Forest.setNumFeatures(0);
            _Random_Forest.setSeed(1);
          _Random_Forest.setMaxDepth(0);
           Classifier_Random_Forest =new FilteredClassifier();
           Classifier_Random_Forest.setClassifier(_Random_Forest);
           Classifier_Random_Forest.buildClassifier(train);
           TRAIN_DATA=train;
```

Figura 34 – Excerto de código da operação "*Start_mfcc*".

6. Operação "*Stop_Server*".

Esta operação coloca o Classificador a *NULL* o que impossibilita que seja possível realizar a identificação uma vez que um dos pré-requisitos para se realizar a identificação é ter o classificador construído. Tem a sua utilidade em caso de manutenção da aplicação *web* "*Web Services*" e da base de dados, fazendo chegar aos clientes uma notificação de serviço indisponível na tentativa de identificação. Na Figura 35 é ilustrado o código desta operação.

```
@WebMethod(operationName = "Stop_Server")
public String Stop_Server() {
    Classifier_Random_Forest=null;
    return "Server Stoped";
}
```

Figura 35 – Código da operação "*Stop_Server*".

7. Operação "*Restart_Server*".

Esta operação executa a operação "*Stop_server*" e a seguir "*Start_server*". É utilizada em caso de introdução de novos elementos a base de dados de treino, face a necessidade de reconstrução do modelo de classificação. Na Figura 36 é ilustrado o código desta operação.

```
@WebMethod(operationName = "Restart_Server")
public String Restart_Server() {
    Classifier_Random_Forest= null;
    String res=Start_mfcc();
    return res;
}
```

Figura 36 – Código da operação "*Restart_Server".*

8. Operação "*feedback_identification*".

Esta operação, permite que do lado da aplicação móvel (*Android*) seja possível o utilizador dar o *feedback* sobre a identificação de um determinado registo áudio. Como ilustrado na Figura 37, esta operação recebe como parâmetro o ID do som identificado e o *feedback*.

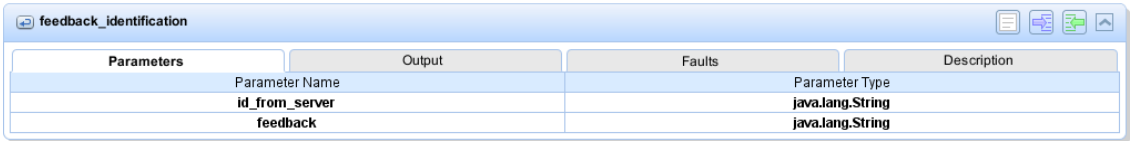
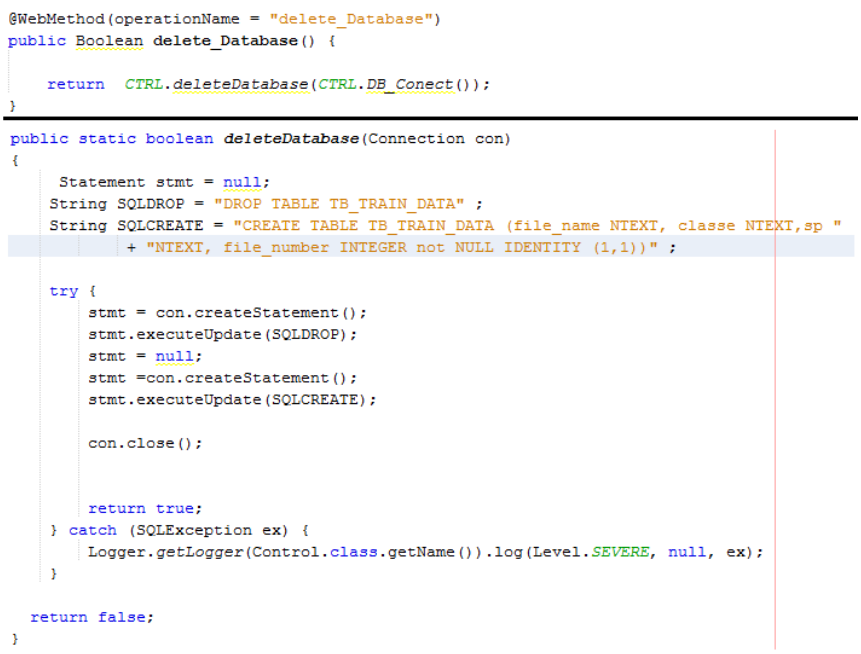


Figura 37 – Operação "*feedback_identification*".

9. Operação "*delete_Database*".

Esta operação permite que do lado da aplicação *desktop* (*Backoffice*), seja possível reconstruir a tabela dos dados de treino, ou seja, é apagado todos os registos da tabela e criada uma nova tabela vazia como ilustrado na Figura 38.



```
@WebMethod(operationName = "delete_Database")
public Boolean delete_Database() {

    return  CTRL.deleteDatabase(CTRL.DB_Conect());
}
```

```
public static boolean deleteDatabase(Connection con)
{
     Statement stmt = null;
    String SQLDROP = "DROP TABLE TB_TRAIN_DATA" ;
    String SQLCREATE = "CREATE TABLE TB_TRAIN_DATA (file_name NTEXT, classe NTEXT,sp "
            + "NTEXT, file_number INTEGER not NULL IDENTITY (1,1))" ;

    try {
        stmt = con.createStatement();
        stmt.executeUpdate(SQLDROP);
        stmt = null;
        stmt =con.createStatement();
        stmt.executeUpdate(SQLCREATE);

        con.close();


        return true;
    } catch (SQLException ex) {
        Logger.getLogger(Control.class.getName()).log(Level.SEVERE, null, ex);
    }

  return false;
}
```

Figura 38 – Código da operação "*delete_Database*".

5.2.2 Aplicação *Desktop* de Gestão (*BackOffice*)

A aplicação *desktop* desempenha um conjunto de funcionalidades que permitem a gestão e o controlo da aplicação *web "Web Services"*.

Na Figura 39, é apresentado o ecrã principal desta aplicação.

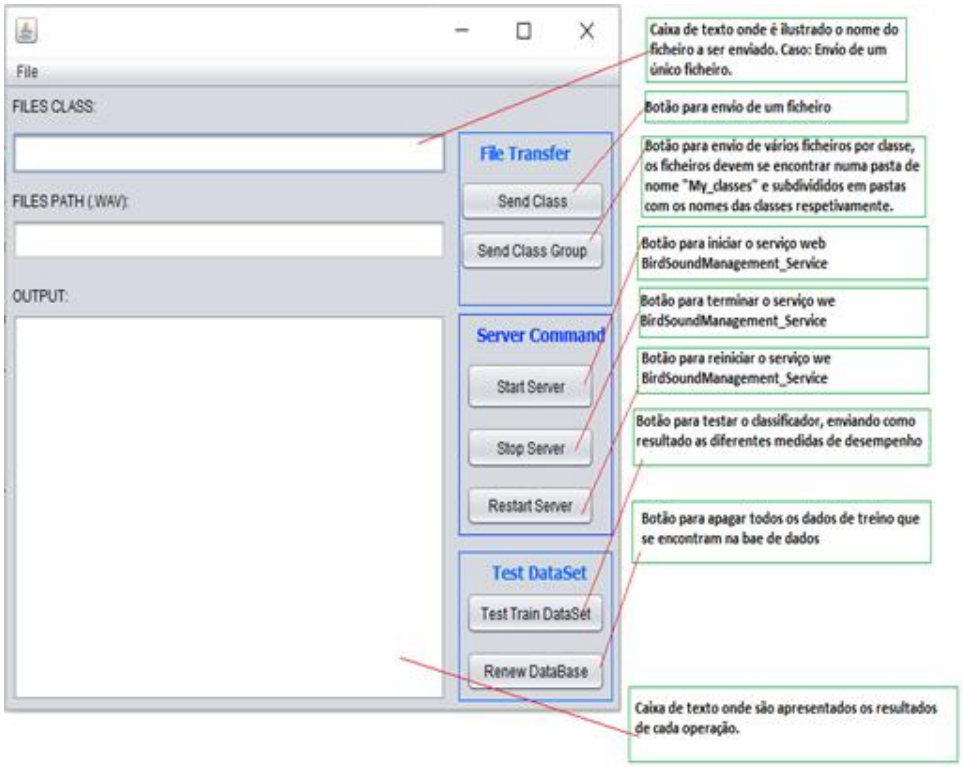


Figura 39 – Ecrã principal da aplicação *desktop* (*Backoffice*).

De seguida é feita a descrição das seguintes funcionalidades implementadas:

- **Identificar som**

Para aceder a esta funcionalidade o utilizador deverá abrir o menu *"File"* e clicar no submenu *"Identify Sound"* que irá abrir uma nova janela, como ilustrado na Figura 40.

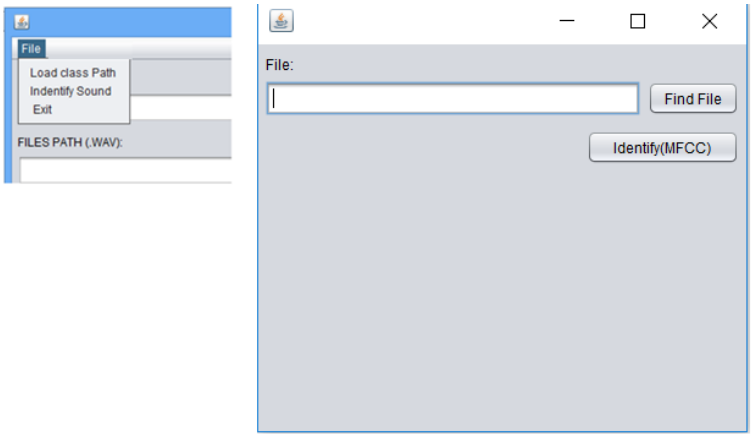


Figura 40 – Aceder a funcionalidade identificar som da aplicação *desktop*.

De seguida, clicando no botão "*Find File*" é aberto uma janela "*filechooser"* onde o utilizador poderá escolher o ficheiro que pretende identificar. Escolhido o ficheiro, este é processado e convertido numa lista de *"double"* como é ilustrado no excerto de código na Figura 41.

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    li= new ArrayList();
  fileChooser.setCurrentDirectory(new java.io.File("."));
  int returnVal = fileChooser.showOpenDialog(this);

  if (returnVal == JFileChooser.APPROVE_OPTION) {

      try {
          double[] res = Signal_Processing.Load_File(fileChooser.getSelectedFile().toString(),Signal_Processing.Filter_Normalize);
          File_T = fileChooser.getSelectedFile().toString();
          for (int i = 0; i < res.length; i++) {
              li.add(res[i]);
          }
          jTextField1.setText(fileChooser.getSelectedFile().toString());
      } catch (IOException ex) {
          Logger.getLogger(Identify_Sound.class.getName()).log(Level.SEVERE, null, ex);
      }


} else {
    System.out.println("File access cancelled by user.");
}
}
```

Figura 41 – Excerto de código para a conversão do ficheiro áudio em lista de "*double".*

O som, uma vez escolhido, é convertido numa lista de "*double"*. Depois de se clicar no botão "*Identify(MFCC)*", a lista de "*double"* é convertido em "*string"* e só depois é enviado, como se pode verificar no excerto de código apresentado na Figura 42.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(li.size()>0 )
    {
        String file_string ="";
        for (int i = 0; i < li.size(); i++) {
            if(i!=li.size()-1)
            {
                file_string+=li.get(i)+"|";
            }
            else
            {
                file_string+=li.get(i);
            }

        }

        try {
        String my_id = getID();

            JOptionPane.showMessageDialog(null,"Este som é um:"+ identifySound(file_string,"0","0","","00/00/0000","00:00",my_id));
        } catch (FileNotFoundException_Exception ex) {
            Logger.getLogger(Identify_Sound.class.getName()).log(Level.SEVERE, null, ex);
        } catch (UnsupportedEncodingException_Exception ex) {
            Logger.getLogger(Identify_Sound.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    else
    {
        JOptionPane.showMessageDialog(null, "File not Loaded!");
    }
}
```

Figura 42 - Excerto de código para execução da operação "*identify_sound_mfcc*" na aplicação *desktop.*

O resultado da classificação é apresentado através de uma "*messageBox*", como ilustrado na Figura 43.

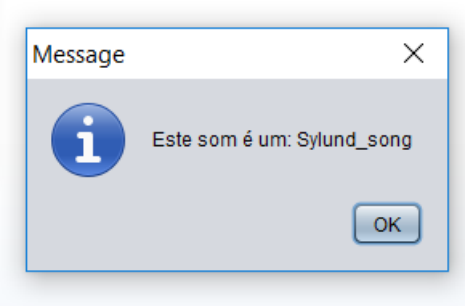


Figura 43 - Resultado da classificação na aplicação *desktop*.

- **Carregar dados de treino**

Esta funcionalidade é executada clicando no botão "*Send Class Group*" como é ilustrado na Figura 44, em que o utilizador deverá criar um pasta no seu ambiente de trabalho com o nome "*My_Classes*" contendo os ficheiros dentro de pastas com o nome da classe a que pertencem.

Uma vez enviado os dados do treino o utilizador receberá uma mensagem confirmando o envio dos dados, como se pode verificar na Figura 44.

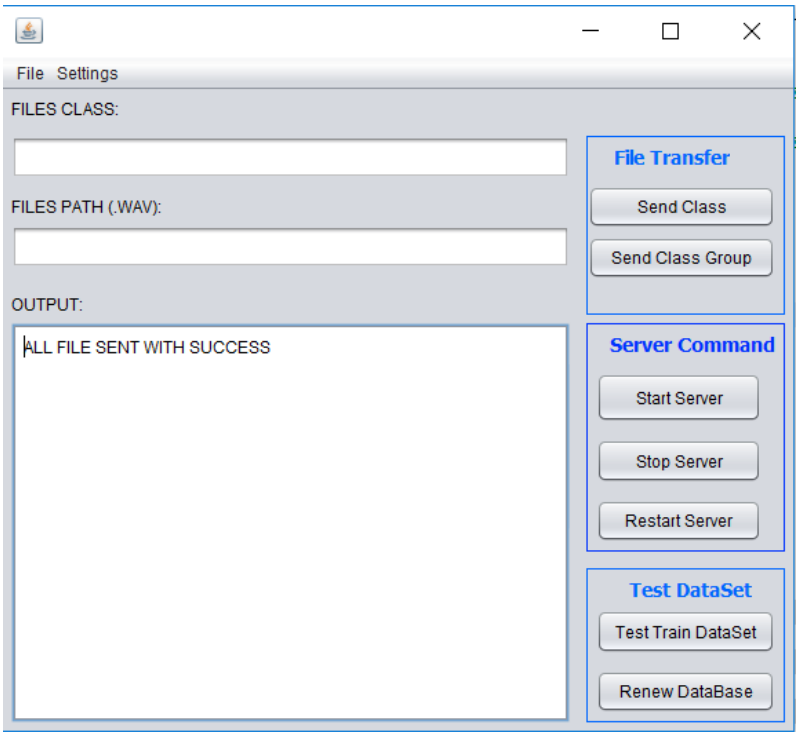


Figura 44 – Mensagem após o envio de dados de treino.

- **Testar modelo de classificação**

Esta funcionalidade permite que o utilizador teste o modelo de classificação. Clicando no botão "*Test Train DataSet*" (Figura 44), a funcionalidade é executada. Após ter sido efetuado o teste é retornado ao utilizador o resultado do teste, como ilustrado na Figura 45.

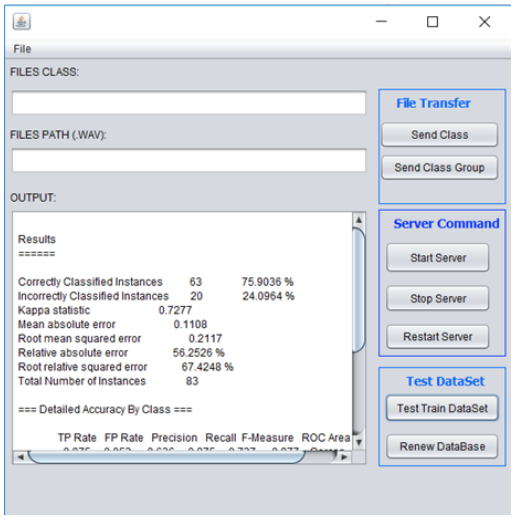


Figura 45 – Mensagem de resposta com sucesso após a execução da operação de teste do classificador.

- **Iniciar *Web Services***

Esta funcionalidade é executada clicando no botão *"Start Server"*. De seguida é executado o código que chama a operação "*Start_mfcc*", como ilustrado na Figura 46.

```
private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

        init_cofigurations();
        relatorio.setText(startMfcc());

}
```

Figura 46 – Código do botão *"Start Server"*.

- **Parar *Web Services***

Esta funcionalidade é executada clicando no botão "*Stop Server*". De seguida é executado o código que chama a operação "*stopServer*", como ilustrado na Figura 47.

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
        relatorio.setText(stopServer());          // TODO add your handl
}
```

Figura 47 - Código do botão *"stopServer"*.

- **Reiniciar *Web Services***

Esta funcionalidade é executada clicando botão *"Restart Server"*. De seguida é executado o código que chama a operação "*restartServer*", como ilustrado na Figura 48.

```
private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {
    relatorio.setText(restartServer());        // TODO add your handling code here:
}
```

Figura 48 - Código do botão *"restartServer"*

- **Recriar a base de dados**

Esta funcionalidade é executada clicando no botão *"Renew DataBase"*. De seguida é executado o código que chama a operação "*deleteDatabase*", como ilustrado na Figura 49.

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(deleteDatabase())
    {
        relatorio.setText("Data Base renewed with sucess");
    }
    else
    {
        relatorio.setText("Renew DataBase Error");
    }
}
```

Figura 49- Código do botão *"Renew DataBase"*.

5.2.3 Aplicação Móvel (*Android*)

A aplicação móvel, cujo desenho foi já apresentado no capítulo 5.1.2.1, tem a funcionalidade de gravar os sons, geri-los bem como identifica-los e dar o *feedback* sobre o resultado da classificação.

Para tal, foi construída uma interface interativa e intuitiva, que permitisse o desempenho destas funcionalidades.

A interface da aplicação foi construída com base nas cores da bandeira do país na qual a aplicação esta voltada, neste caso S.Tomé e Príncipe.

Nas figuras seguintes serão ilustradas as interfaces para o desempenho de cada uma das funcionalidades supracitadas.

- **Gravar sons.**

Na Figura 50 é ilustrada a interface principal da aplicação onde, por meio de uma lista, é possível ver todos os sons gravados. Os ficheiros com o pássaro a cores indicam que estes já foram identificados e tiveram o *feedback* positivo, enquanto que os ficheiros com o pássaro de cor cinza indicam que estes não foram identificados ou o seu *feedback* foi negativo. Esta *interface* é apresentada quando o utilizador se encontrar na atividade *MainActivity*.

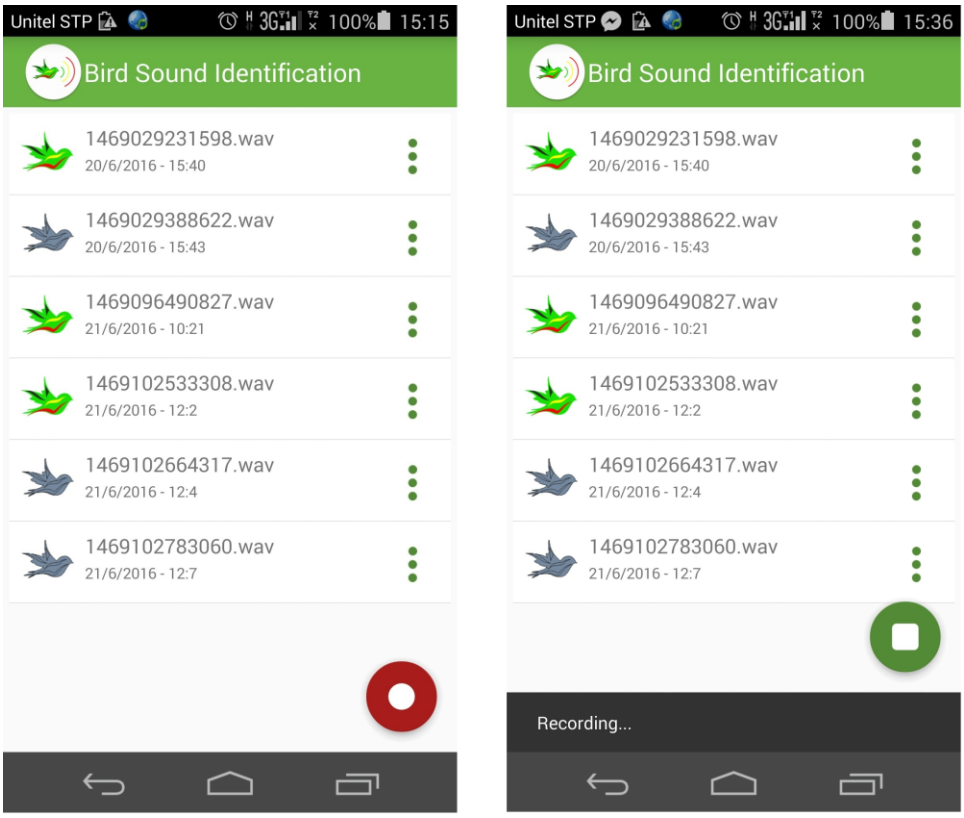


Figura 50 - Interface gravar sons.

Para gravar o som, basta clicar no botão inferior vermelho e para parar a gravação clicar no botão verde. Clicando no botão "*Record*" é verificado em primeiro lugar a existência da ligação de dados ou de GPS. Caso não haja nenhuma destas ligações é enviado um alerta ao utilizador pedindo que as ative, como ilustrado na Figura 51.

```
if (Start != null) {
    Start.setOnClickListener((view) -> {
            control_RingBar=false;
            control_GPS = false;

            locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
            locationManagerNETWORK = (LocationManager) getSystemService(Context.LOCATION_SERVICE);

            //getting GPS status
            isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

            //getting network status
            try{
                ConnectivityManager cm =
                        (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
                NetworkInfo netInfo = cm.getActiveNetworkInfo();
                isNetworkEnabled = netInfo.isConnected();//locationManager.isProviderEnabled(Locat
            }
            catch (Exception e)
            {
                isNetworkEnabled=false;
            }


            if(!isGPSEnabled || !isNetworkEnabled)
            {
                alerta.showSettingsAlert();
```

Figura 51 – Código do botão *"Record"*.

Existindo uma destas ligações, a aplicação prossegue iniciando a gravação do som, como ilustra a Figura 52.

```
if(!isGPSEnabled || !isNetworkEnabled)
{
    alerta.showSettingsAlert();
}
else
{
    if(isGPSEnabled)
    {

        provider_info = LocationManager.GPS_PROVIDER;

    }
    if(isNetworkEnabled)
    {

        provider_infoNETWORK = LocationManager.NETWORK_PROVIDER;
    }

    gravador.startRecording();
    Snackbar.make(view, "Recording...", Snackbar.LENGTH_LONG)
            .setAction("Action", null).show();

    Start.setVisibility(View.INVISIBLE);
    Stop.setVisibility(View.VISIBLE);

}
```

Figura 52 - Código do botão *"Record"* (Continuação).

Após ser clicado o botão "*Stop*", são recolhidas as informações necessárias do local da gravação, como as coordenadas, hora, data e endereço , como ilustrado na Figura 53.

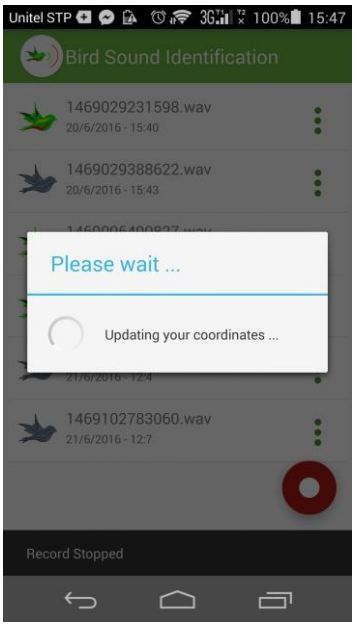


Figura 53 - Interface para recolha de informações sobre a gravação.

Para a recolha das anotações, com exceção da data e hora, é utilizada a classe *LocationManager* do *Android*, como ilustrado na Figura 54. O *update* da localização do utilizador é feito através da ligação GPS e a ligação de dados. A ligação que estiver disponível no momento ou que for mais rápida a retornar o resultado é a que será levada em conta.

```
if (Stop != null) {
    Stop.setOnClickListener((view) → {
            Snackbar.make(view, "Record Stopped", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
            FILE_NAME =gravador.stopRecording();
            if (!provider_info.isEmpty() && !provider_infoNETWORK.isEmpty()) {
                try {
                    locationManager.requestLocationUpdates(provider_info, 0, 0, locationListener);
                    locationManagerNETWORK.requestLocationUpdates(provider_infoNETWORK,0,0,locationListenerNETWPRK);

                }catch (SecurityException e)
                {
                    return;
                }

            }
            new  AsyncCaller().execute();

            Stop.setVisibility(View.INVISIBLE);
            Start.setVisibility(View.VISIBLE);
    });
}
```

Figura 54 - Recolha de anotações “Gravar som”.

- **Identificar o som**

A identificação do som pode ser realizada de duas formas:

A primeira forma é a automática. Quando o utilizador clica no botão "*Stop*", a seguir à recolha de dados sobre o local e a data da gravação, é-lhe perguntado se deseja efetuar a classificação através de um "*AlertDialog*". A aplicação oferece ao utilizador essa escolha, pelo fato desta funcionalidade requerer a utilização da internet. Assim, o utilizador poderá gravar os sons e as suas respetivas coordenadas e efetuar a classificação no momento que desejar. Na Figura 55 é ilustrada a interface que surge após serem guardadas as coordenadas.

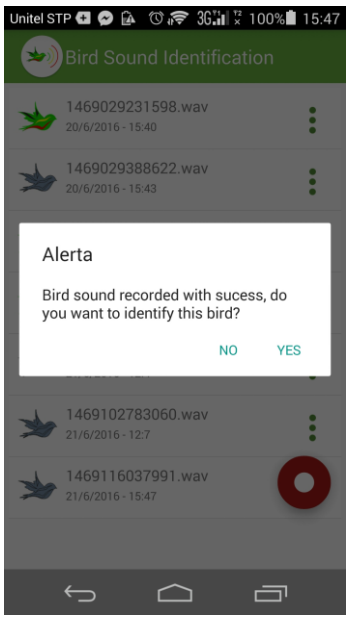


Figura 55 - Interface para pedido de identificação

Optando por identificar o som é chamada uma tarefa do tipo assíncrona "*ClassifyerCalling*" que irá executar o serviço *web* "*BirdSoundManagement_Service*", chamando a operação "*identify_sound_mfcc*", como ilustrado na Figura 56.

```
Bird_Audio a = new Bird_Audio(0,FILE_NAME,Soung_Long,Sound_Lat,Sound_Adress,date,time,"",0);
db.addAudio(a);
fillList();

AlertDialog.Builder alertDialog = new AlertDialog.Builder(MainActivity.this);

//Setting Dialog Title
alertDialog.setTitle("Alerta");

//Setting Dialog Message
alertDialog.setMessage("Bird sound recorded with sucess, do you want to identify this bird?")

//On Pressing Setting button
alertDialog.setPositiveButton("Yes", (dialog, which) → {
        new ClassifyerCalling().execute();
});

//On pressing cancel button
alertDialog.setNegativeButton("No", (dialog, which) → {
        dialog.cancel();
});

alertDialog.show();
```

```
private class ClassifyerCalling extends AsyncTask<Void, Void, Void>
{
    String res_especie;
    String id;
    public ProgressDialog ringProgressDialog ;

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        //this method will be running on UI thread
        ringProgressDialog = ProgressDialog.show(MainActivity.this, "Please wait ...", "Identifying...", true);
        ringProgressDialog.setCancelable(true);
    }
    @Override
    protected Void doInBackground(Void... params) {

        //this method will be running on background thread so don't update UI frome here
        //do your long running http tasks here,you dont want to pass argument and u can access the parent class'
        //identificar o som
        id= BirdIdentificationServer.get_id();
        res_especie = BirdIdentificationServer.indentifySound(FILE_NAME,Soung_Long,Sound_Lat,
                Sound_Adress,Sound_data,Sound_hora,id);


        return null;
    }
```

Figura 56 - Código de execução da operação "*identify_sound_mfcc*" através da aplicação móvel.

A segunda forma, é através do "*Menupopup*", onde aparece a opção *"Identify"*, como ilustrado na Figura 57.

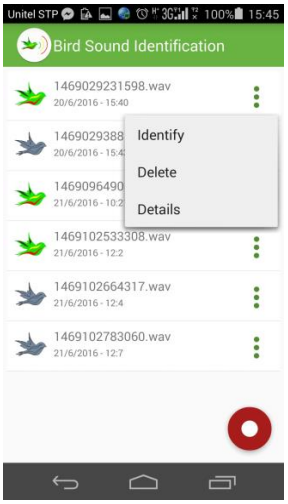


Figura 57 - Função identificar som através do *MenuPopUp*.

Clicando na opção "*Identify*" são seguidos os mesmos procedimentos da primeira forma.

- **Dar *feedback*.**

Como referido acima, após a gravação do som é questionado ao utilizador se pretende identificar o som gravado. Respondendo de forma positiva à questão ou por meio do "*MenuPopUp*" escolhendo a opção "Identify" é realizada a classificação do som, identificando a espécie do pássaro como ilustrado na Figura 58.

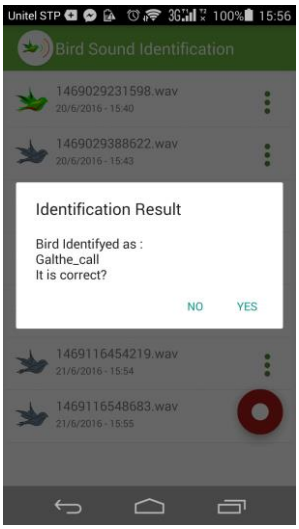


Figura 58 - Interface para dar o *feedback.*

Como se pode verificar na Figura 58, é apresentado o resultado da classificação e pedido ao utilizador o *feedback* do resultado.

Dando o *feedback* é executada uma tarefa assíncrona "*feedback"* que por sua vez irá chamar a operação "*feedback_identification"* do serviço "*BirdSoundManagement_Service*" da aplicação *web.* Na Figura 59, são apresentados os excertos de código afetos a esta funcionalidade.

```
protected void onPostExecute(Void result) {
    super.onPostExecute(result);

    ringProgressDialog.dismiss();

    AlertDialog.Builder alertDialog = new AlertDialog.Builder(MainActivity.this);

    //Setting Dialog Title
    alertDialog.setTitle("Identification Result");

    //Setting Dialog Message
    alertDialog.setMessage("Bird Identifyed as :\n" + res_especie+"\n"+ "It is correct?");

    //On Pressing Setting button
    alertDialog.setPositiveButton("Yes", (dialog, which) → {
            feedback fe = new feedback();
            fe.id=id;
            fe.feed="YES";
            fe.espcie=res_especie;
            fe.execute();
            dialog.cancel();
    });

    //On pressing cancel button
    alertDialog.setNegativeButton("No", (dialog, which) → {
            feedback fe = new feedback();
            fe.id=id;
            fe.feed="No";
            fe.espcie="Unknow";
            fe.execute();
            dialog.cancel();
    });

    alertDialog.show();
```

```
private class feedback extends AsyncTask<Void, Void, Void>
{

    String id;
    String feed;
    String espcie="";

    @Override
    protected void onPreExecute() {
        super.onPreExecute();

    }
    @Override
    protected Void doInBackground(Void... params) {

        //this method will be running on background thread so don't update UI f
        //do your long running http tasks here,you dont want to pass argument a
        //identificar o som
        String res = BirdIdentificationServer.feedback_identification(id,feed);

        return null;
    }
```

Figura 59 - Excertos de código para a operação *feedback* da aplicação móvel.

- **Gerir os sons.**

Por meio do "*MenuPopUp*", é possível gerir individualmente os sons gravados pela aplicação. Clicando no "*MenuPopUp"* de um determinado ficheiro é possível efetuar as seguintes ações: Apagar, Identificar e Ver os detalhes do registo áudio, como ilustrado na Figura 60.

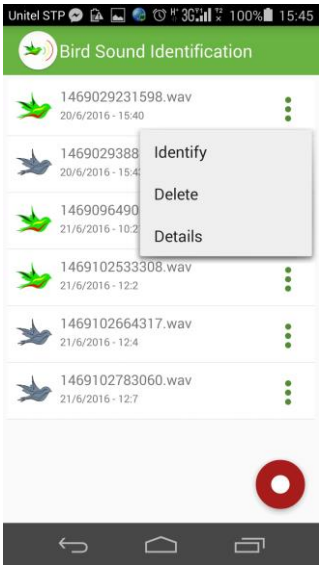


Figura 60 - *MenuPopUp.*

Clicando em "*Details"* será possível aceder a todas as anotações tomadas no ato da gravação, como se pode verificar na Figura 61.

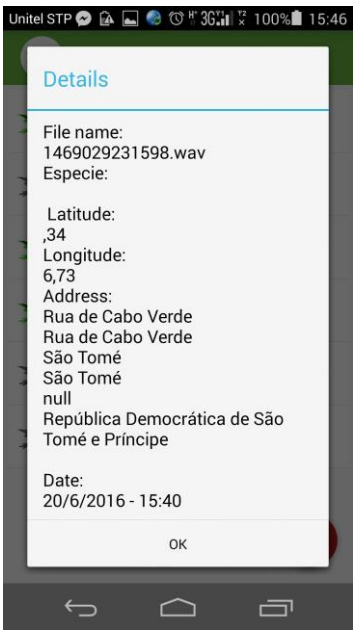


Figura 61 - Interface com detalhes do som gravado.

6 Testes

6.1 Testes e Avaliação da Aplicação *web*.

Nesta seção, apresenta-se a metodologia de avaliação, os testes efetuados aos modelos de classificação da aplicação *web* e os resultados obtidos. Esta seção está dividida em três subseções.

Na subseção 6.1.1 , são descritas as hipóteses, metodologias e medidas de avaliação e testes a serem utilizados na avaliação da aplicação *web*.

Na subseção 6.1.2, são descritos os modelos implementados e os respetivos resultados obtidos.

Na subseção 6.1.3, é feita a conclusão das experiências realizadas com base na taxa de acerto de cada modelo.

6.1.1 Avaliação da Solução

6.1.1.1 Medidas de avaliação

Nesta seção é apresentado as medidas de avaliação de desempenho utilizadas para avaliar os resultados obtidos pelos diferentes modelos de classificação utilizados neste trabalho. Estas medidas de desempenho podem ser explicadas com base na Figura 62.

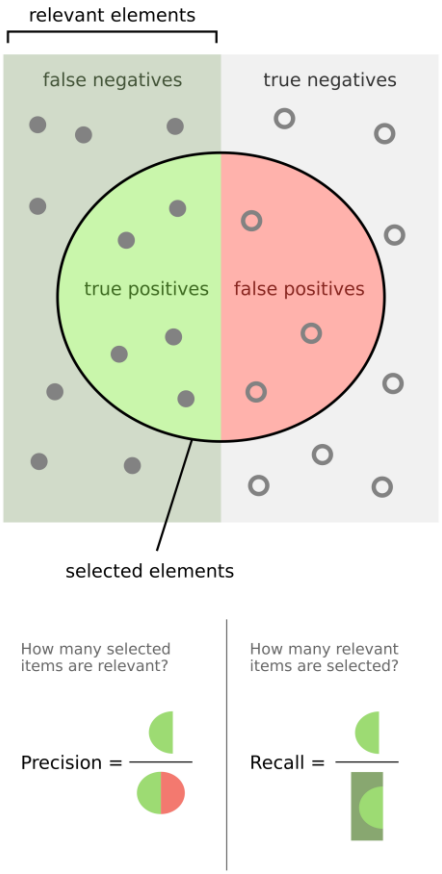


Figura 62 - *Precision* e *Recall* [Wikipedia].

Medidas de desempenho:

- ***Precision*:** É uma medida que para a identificação de uma determinada classe, nos dá a informação relevante sobre um processo de identificação, ou seja, sobre um universo de identificações realizadas, quantas estão corretas. Esta medida é calculada com base na equação 9.

$Precision = \frac{TP}{TP+FP}$	[9]

Onde *true positives* (TP) são as predições corretas e *false positives* (FP) as predições incorretas.

- ***Recall***: Ao contrário do *precision*, para a identificação de uma determinada classe esta medida dá-nos a informação acerca do número de predições corretas em relação

ao universo de todas instâncias existentes para esta classe. Esta medida é calculada com base na equação 10.

$Recall = \frac{TP}{TP+FN}$	[10]

Onde *true positives* (TP) são as predições corretas *e false negative* (FN) é total de instâncias para uma determinada classe com exceção das que foram corretamente identificadas.

- **F-*measure***: É uma medida que nos dá a média ponderada do *precision* e *recall* calculada a partir da equação 11.

$F\text{-}measure = 2 * \frac{Precision * Recall}{Precision + Recall}$	[11]

- ***Accuracy***: Esta medida (taxa de acerto) diz-nos o número de predições corretas sobre o total de predições realizadas, calculadas a partir da equação 12.

$Accuracy = \frac{TP + TN}{TP+TN+FP+FN}$	[12]

Onde *true positives* (TP) são as predições corretas, *true negative* (TN) são as instâncias que foram corretamente rejeitadas na classificação, *e false negative* (FN) é total de instâncias para uma determinada classe com exceção das que foram corretamente identificadas e *false positives* (FP) as predições incorretas.

6.1.1.2 Hipóteses

Pretende-se testar a hipótese de que um dos modelos de classificação obtém os melhores resultados para uma dada medida de avaliação (podendo coincidir o mesmo para diferentes medidas).

Assim, a hipótese será: O classificador A consegue melhores resultados (segundo a medida M) que o classificador B.

No caso deste trabalho, em nosso entender apenas se justifica testar o caso da abordagem usando MFCC, pois como se pode observar na secção 6.1.2, os resultados obtidos seguindo a abordagem dos *motifs* foram, em todos os casos testados, inferiores aos resultados obtidos seguindo a abordagem dos MFCCs.

No entanto, como se dedicou também à aplicação móvel, não foram efetuados testes exaustivos aos modelos de classificação, apesar destes serem bastante importantes, propondo-os como trabalho futuro.

Também, para uma avaliação com mais segurança em relação ao desemprenho dos modelos de classificação, propõe-se como trabalho futuro o uso do *t-test* para amostras emparelhadas, dado se ter usado a API do Weka.

6.1.1.3 Metodologia de avaliação

Quando se trabalha sobre um conjunto de dados é necessário ter o cuidado de o conjunto usado para treino dos algoritmos não seja o mesmo conjunto usado para a avaliação (de teste). Caso contrário, os resultados produzidos serão otimistas dado que os algoritmos tentaram melhorar a sua capacidade preditiva sobre esse conjunto. Assim, devem usar-se métodos de amostragem mais fiáveis definindo um conjunto de teste e outro de treino. Um dos métodos de amostragem muito utilizado é a validação cruzada (*cross-validation*).

Para efetuar as nossas experiências com os algoritmos de classificação será usado o método de validação cruzada com 10 subconjuntos (*folds*). Este método consiste em dividir o conjunto de dados de entrada em 10 subconjuntos e construir 10 submodelos, onde em cada um são utilizados 9 conjuntos para treino e um conjunto para teste. Em cada submodelo, o conjunto de teste usado será diferente, assim como os dados de treino que não terão presentes os dados de teste. Para cada submodelo é então calculada a respetiva medida de desempenho (por exemplo, *accuracy*) e o resultado da avaliação é a média dos desempenhos observados.

6.1.2 Soluções e Resultados obtidos

Para a avaliação dos diferentes classificadores foi utilizado o método de validação cruzada com 10 *Folds.*

6.1.2.1 Dataset

Para a realização das experiências foi utilizado o *dataset* disponibilizado pela *Neural Information Processing Scaled for Bioacustic Bird song competition* (NIPS4B) [Kaggle,2016].

O *dataset* utilizado é constituído por 87 classes e 687 elementos (registo áudio). Nestes elementos, encontram-se sons de pássaros, insetos, anfíbios e mamíferos e para alguns elementos, pode-se encontrar mais que uma classe associada.

Para as experiências foram utilizados apenas os sons/elementos correspondentes aos pássaros e foi feita a filtragem dos que pertenciam apenas a uma única espécie não havendo a necessidade de efetuar alguma modificação nos elementos originais.

A filtragem e a separação dos elementos foi possível através das informações contidas num conjunto de ficheiros obtidos a partir de [Kaggle,2016].

Sendo assim, após a filtragem do *dataset* foram escolhidas as classes com um número de elementos disponíveis superior a 7, isto com o objetivo de construir um *dataset* com menos disparidade possível em relação ao número de elementos por classe. No final obteve-se assim o total de 9 classes e 103 elementos, como pode ser visualizado na Tabela 5.

Tabela 5 - Classes e elementos utilizados nos testes dos classificadores.

Classe	**Número de Elementos**
Carcan_call	11
Carcan_song	15
Galthe_call	8
Galthe_song	12
Oriori_call	8
Sylcan_call	13
Sylmel_call	11

Sylund_call	10
Sylund_song	15

6.1.2.2 Experiências e Resultados obtidos com base na abordagem *motifs.*

Neste capítulo serão descritas as experiências realizadas com base na abordagem *motifs* para extração de atributos, explicada no capítulo 3.2.2. Serão também apresentados os respetivos resultados obtidos.

Foi aplicado ao sinal áudio duas abordagens de pré-processamento antes da extração dos atributos *motifs*. Experimentaram-se duas abordagens de pré-processamento, de modo a tentar melhorar os nossos resultados.

A primeira abordagem consistiu na aplicação do filtro passa-baixo, a seguir foi feito a dizimação do sinal resultante com o objetivo de diminuir o número de amostras do sinal no tempo e depois feito a sua normalização e calculado o envelope de Shannon.

A segunda abordagem baseava-se na aplicação do filtro passa-banda ao som original com frequência de corte inferior e superior a 1500Hz e 18000Hz respetivamente [curió,2016] e de seguida a normalização e cálculo do envelope de Shannon do sinal resultante.

Nas experiências foi utilizado o programa Mrmotif apenas com os seguintes parâmetros:

- Tamanho do *motifs*: 40
- Resolução: 8
- *Top-k* :40
- *Overlap*:10

Com o objetivo da a construção do melhor modelo de classificação foram realizadas experiências com os 3 algoritmos de classificação citados no capítulo 3.3.1.

Os resultados obtidos são ilustrados por meio de tabelas apresentadas na subsecção seguinte.

6.1.2.2.1 Pré-processamento: Aplicação de Filtro passa baixo, dizimação, normalização e cálculo do envelope de Shannon.

Na Tabela 6 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (G) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-baixo (fc-18000 HZ), *decimate*, normalização do sinal e posteriormente cálculo do envelope de Shannon.

- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Random Forest* com 200 árvores.

Tabela 6 - Resultados obtidos utilizando os atributos *motifs* e o *Random Forest* com 200 árvores.

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.333	0.091	0.143
Carcan_song	0.625	0.33	0.435
Galthe_call	0	0	0
Galthe_song	0.5	0.333	0.4
Oriori_call	0.75	0.375	0.5
Sylcan_call	0.833	0.385	0.526
Sylmel_call	0.4	0.182	0.25
Sylund_call	0	0	0
Sylund_song	0.19	0.8	0.308
Média	**0.419**	**0.311**	**0.302**

Como ilustrado na Tabela 6, este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual a 41.9%. O que nos indica que em média de todas as predições efetuadas apenas 41.9% estão corretas.
- *Recall*: Igual a 31.1 %. O que nos indica que em média, para cada classe dado o número de instâncias existentes, apenas 31.1% delas foram corretamente identificadas.
- *F-Measure*: Igual 30.2%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 30.2%.

Na Tabela 7 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (F) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-baixo (fc-18000 HZ), *decimate*, normalização do sinal e posteriormente cálculo do envelope de Shannon.

- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Decision Tree* (j48).

Tabela 7 - Resultados obtidos utilizando os atributos *motifs* e o *Decision Trees* (J48).

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0	0	0
Carcan_song	0	0	0
Galthe_call	0	0	0
Galthe_song	0	0	0
Oriori_call	0.6	0.375	0.462
Sylcan_call	1	0.154	0.267
Sylmel_call	0	0	0
Sylund_call	0	0	0
Sylund_song	0.165	1	0.283
Média	**0.197**	**0.194**	**0.111**

Como se pode observar na Tabela 7, este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual a 19.7%. O que nos indica que em média de todas as predições efetuadas apenas 19.7% estão corretas.
- *Recall*: Igual a 19.4%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, apenas 19.4% delas foram corretamente identificadas.
- *F-Measure*: Igual 11.1%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 11.1%.

Na Tabela 8 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (H) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-baixo (fc-18000 HZ), *decimate*, normalização do sinal e posteriormente cálculo do envelope de Shannon.
- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Support Vector Machine/SMO* da ferramenta LIBSVM com os parâmetros padrão [weka,2016] .

Tabela 8 - Resultados obtidos utilizando os atributos *motifs* e o *Support Vector Machine* com as configurações por defeito [weka,2016].

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0	0	0
Carcan_song	0.333	0.267	0.296
Galthe_call	0	0	0
Galthe_song	0.5	0.167	0.25
Oriori_call	1	0.125	0.222
Sylcan_call	0.5	0.462	0.48
Sylmel_call	0.5	0.91	0.154
Sylund_call	0	0	0
Sylund_song	0.181	0.867	0.299
Média	**0.327**	**0.262**	**0.21**

Como ilustrado na Tabela 8, este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual a 32.7%. O que nos indica que em média de todas as predições efetuadas apenas 32.7 % estão corretas.
- *Recall*: Igual a 26.2 %. O que nos indica que em média, para cada classe dado o número de instâncias existentes, apenas 26.2 % delas foram corretamente identificadas.
- *F-Measure*: Igual 21 %, ou seja, a média ponderada do *Precision* e *Recall* é igual a 21 %.

6.1.2.2.2 **Pré-processamento: Aplicação do filtro passa-banda (fcmin-1500Hz e fcmax-18000Hz) ao sinal original, normalização do sinal resultante e o cálculo do envelope de Shannon.**

Na Tabela 9 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (J) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon.
- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Random Forest* com 200 árvores.

Tabela 9 - Resultados obtidos utilizando os atributos *motifs* e o *Random Forest* com 200 árvores.

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.2	0.091	0.125
Carcan_song	0.207	0.8	0.329
Galthe_call	0	0	0
Galthe_song	0	0	0
Oriori_call	0	0	0
Sylcan_call	0.545	0.462	0.5
Sylmel_call	0.273	0.273	0.273
Sylund_call	0	0	0
Sylund_song	0.111	0.067	0.083
Média	**0.166**	**0.223**	**0.166**

Como ilustrado na Tabela 9 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual a 16.6%. O que nos indica que em média de todas as predições efetuadas apenas 16.6% estão corretas.
- *Recall*: Igual a 22.3%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, apenas 22.3% delas foram corretamente identificadas.
- *F-Measure*: Igual 16.6%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 16.6 %.

Na Tabela 10 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (I) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon.

- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Decision Tree* (j48).

Tabela 10 - Resultados obtidos utilizando os atributos *motifs* e o *Decision Trees* (J48).

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0	0	0
Carcan_song	0.186	0.867	0.306
Galthe_call	0	0	0
Galthe_song	0	0	0
Oriori_call	0	0	0
Sylcan_call	0.4	0.308	0.348
Sylmel_call	0.111	0.091	0.1
Sylund_call	0	0	0
Sylund_song	0.167	0.067	0.095
Média	**0.114**	**0.184**	**0.113**

Como ilustrado na Tabela 10 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 11.4%. O que nos indica que em média de todas as predições efetuadas apenas 11.4% estão corretas.
- *Recall*: Igual a 18.4%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, apenas 18.4% delas foram corretamente identificadas.
- *F-Measure*: Igual 11.3%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 11.3%.

Na Tabela 11 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (K) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon.
- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Support Vector Machine/SMO* da ferramenta LIBSVM com os parâmetros padrão [weka,2016].

Tabela 11 - Resultados obtidos utilizando os atributos *motifs* e o *Support Vector Machine* com as opções por defeito [weka,2016].

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0	0	0
Carcan_song	0.206	0.867	0.333
Galthe_call	0	0	0
Galthe_song	0.333	0.083	0.133
Oriori_call	0	0	0
Sylcan_call	0.375	0.692	0.486
Sylmel_call	0.5	0.091	0.154
Sylund_call	0	0	0
Sylund_song	0.273	0.2	0.231
Média	**0.209**	**0.262**	**0.176**

Como ilustrado na Tabela 11 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 20.9%. O que nos indica que em média de todas as predições efetuadas apenas 20.9% estão corretas.
- *Recall*: Igual a 26.2%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, apenas 26.2% delas foram corretamente identificadas.
- *F-Measure*: Igual 17.6%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 17.6%.

Das experiências realizadas para a abordagem dos *motifs*, podemos observar que os melhores resultados foram para o modelo de classificação (G) com a seguinte configuração:

- **Pré-processamento:** Aplicação de filtro passa-baixo (fc-18000 HZ), *decimate*, normalização do sinal e posteriormente cálculo do envelope de Shannon.
- **Atributos:** *Motifs.*
- **Algoritmo de Classificação:** *Random Forest* com 200 árvores.
-

6.1.2.3 Experiências e Resultados obtidos com base na abordagem MFCC

Após a realização de experiências com base na abordagem dos *motifs* foram realizadas experiências com base na abordagem usando MFCCs.

Nesta abordagem foram aplicadas as técnicas descritas no capítulo 3.2.1 e utilizados os seguintes parâmetros para a configuração dos MFCCs:

- Número de Filtros: 40
- Número de Coeficientes: 14
- Tamanho da Janela: 1024

Para a realização dos testes, antes da extração de atributos MFCCs foram aplicadas duas abordagens no pré-processamento do som que se mostraram muito relevantes devido aos resultados obtidos. A primeira abordagem baseou-se apenas na normalização do som original, enquanto a segunda baseou-se na aplicação do filtro passa banda ao sinal original com frequência de corte inferior e superior a 1500Hz e 18000Hz respetivamente [curió,2016] e a normalização e cálculo do envelope de Shannon dos resultados obtidos da filtragem.

Nas tabelas a seguir serão apresentados os resultados obtidos pelas diferentes abordagens de pré-processamento em combinação com os algoritmos de classificação citados na secção 3.3.1.

6.1.2.3.1 Pré-processamento: Normalização do Sinal Original.

Na Tabela 12 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (B) com as seguintes configurações:

- **Pré-processamento:** Normalização do sinal.
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Random Forest* com 200 árvores.

Tabela 12 - Resultados obtidos utilizando MFCCs e o *Random Forest* com 200 árvores.

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.692	0.818	0.75
Carcan_song	0.8	0.8	0.8
Galthe_call	0.6	0.375	0.462

Galthe_song	0.667	0.5	0.571
Oriori_call	0.778	0.875	0.824
Sylcan_call	0.643	0.692	0.667
Sylmel_call	0.8	0.727	0.762
Sylund_call	0.8	0.8	0.8
Sylund_song	0.833	1	0.909
Média	**0.741**	**0.748**	**0.739**

Como ilustrado na Tabela 12 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 74.1%. O que nos indica que em média de todas as predições efetuadas 74.1 % estão corretas.
- *Recall*: Igual a 74.8%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, 74.8% delas foram corretamente identificadas.
- *F-Measure*: Igual 73.9%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 73.9%.

Na Tabela 13 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (A) com as seguintes configurações:

- **Pré-processamento:** Normalização do sinal.
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Decision Tree* (j48).

Tabela 13 - Resultados obtidos utilizando MFCCs e o *Decision Trees* (J48).

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.545	0.545	0.545
Carcan_song	0.667	0.667	0.667
Galthe_call	0.2	0.125	0.154
Galthe_song	0.455	0.417	0.435
Oriori_call	0.5	0.5	0.5

Sylcan_call	0.714	0.769	0.741
Sylmel_call	0.778	0.636	0.7
Sylund_call	0.667	0.667	0.727
Sylund_song	0.611	0.733	0.667
Média	**0.59**	**0.602**	**0.593**

Como ilustrado na Tabela 13 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 59 %. O que nos indica que em média de todas as predições efetuadas 59% estão corretas.
- *Recall*: Igual a 60.2%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, 60.2% delas foram corretamente identificadas.
- *F-Measure*: Igual 59.3%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 59.3%.

Na Tabela 14 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (C) com as seguintes configurações:

- **Pré-processamento:** Normalização do sinal.
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Support Vector Machine/SMO* da ferramenta LIBSVM com os parâmetros padrão [weka,2016].

Tabela 14 - Resultados obtidos utilizando MFCCs e o *Support Vector Machine* com as opções por defeito [weka,2016].

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.643	0.818	0.72
Carcan_song	0.545	0.8	0.649
Galthe_call	0	0	0
Galthe_song	0.429	0.5	0.462
Oriori_call	0.4	0.5	0.444

Sylcan_call	1	0.077	0.143
Sylmel_call	0.8	0.364	0.5
Sylund_call	0.538	0.7	0.609
Sylund_song	0.652	1	0.789
Média	**0.588**	**0.563**	**0.505**

Como ilustrado na Tabela 14 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 58.8 %. O que nos indica que em média de todas as predições efetuadas 58.8 % estão corretas.
- *Recall*: Igual a 56.3%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, 56.3% delas foram corretamente identificadas.
- *F-Measure*: Igual 50.5%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 50.5%.

6.1.2.3.2 **Pré-processamento: Aplicação do filtro passa-banda (fcmin-1500Hz e fcmax-18000Hz) ao sinal original, normalização do sinal resultante e o cálculo do envelope de Shannon.**

Na Tabela 15 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (E) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon..
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Random Forest* com 200 árvores.

Tabela 15 - Resultados obtidos utilizando MFCCs e o *Random Forest* com 200 árvores.

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.25	0.091	0.133
Carcan_song	0.35	0.467	0.4
Galthe_call	0	0	0

Galthe_song	0.167	0.167	0.167
Oriori_call	0.2	0.125	0.154
Sylcan_call	0.526	0.769	0.625
Sylmel_call	0.462	0.545	0.5
Sylund_call	0.2	0.1	0.133
Sylund_song	0.381	0.533	0.444
Média	**0.303**	**0.35**	**0.314**

Como ilustrado na Tabela 15 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 30.3%. O que nos indica que em média de todas as predições efetuadas 30.3 % estão corretas.
- *Recall*: Igual a 35%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, 35% delas foram corretamente identificadas.
- *F-Measure*: Igual 31.4%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 31.4%.

Na Tabela 16 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (D) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon..
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Decision Tree* (j48).

Tabela 16 - Resultados obtidos utilizando MFCCs e o *Decision Trees* (J48).

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.077	0.091	0.083
Carcan_song	0.455	0.333	0.385
Galthe_call	0.167	0.25	0.2

Galthe_song	0.385	0.417	0.4
Oriori_call	0.222	0.25	0.235
Sylcan_call	0.571	0.615	0.593
Sylmel_call	0.111	0.091	0.1
Sylund_call	0.222	0.2	0.211
Sylund_song	0.385	0.333	0.357
Média	**0.311**	**0.301**	**0.303**

Como ilustrado na Tabela 16 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 31.1%. O que nos indica que em média de todas as predições efetuadas 31.1% estão corretas.
- *Recall*: Igual a 30.1%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, 30.1% delas foram corretamente identificadas.
- *F-Measure*: Igual 30.3%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 30.3%.

Na Tabela 16 é apresentado os resultados do *Precision, Recall e F-measure* para o modelo de classificação (E) com as seguintes configurações:

- **Pré-processamento:** Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon.
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Support Vector Machine/SMO* da ferramenta LIBSVM com os parâmetros padrão [weka,2016].

Tabela 17- Resultados obtidos utilizando MFCCs e o *Support Vector Machine* com as opções por defeito [weka,2016].

Classes	Precision x(100%)	Recall x(100%)	F-Measure x(100%)
Carcan_call	0.375	0.273	0.316
Carcan_song	0.4	0.667	0.5
Galthe_call	0.5	0.125	0.2
Galthe_song	0.333	0.333	0.333

Oriori_call	0	0	0
Sylcan_call	0.833	0.769	0.8
Sylmel_call	0.667	0.182	0.286
Sylund_call	0	0	0
Sylund_song	0.333	0.8	0.471
Média	**0.401**	**0.408**	**0.361**

Como ilustrado na Tabela 17 este modelo de classificação apresenta como média das medidas de desempenho os seguintes resultados:

- *Precision*: Igual 40.1 %. O que nos indica que em média de todas as predições efetuadas 40.1 % estão corretas.
- *Recall*: Igual a 40.8%. O que nos indica que em média, para cada classe dado o número de instâncias existentes, 40.8% delas foram corretamente identificadas.
- *F-Measure*: Igual 36.1%, ou seja, a média ponderada do *Precision* e *Recall* é igual a 36.1%.

Das experiências realizadas para a abordagem dos MFCC, podemos observar que os melhores resultados foram para o modelo de classificação (B) com a seguinte configuração:

- **Pré-processamento:** Normalização do sinal.
- **Atributos:** MFCC.
- **Algoritmo de Classificação:** *Random Forest* com 200 árvores.

6.1.3 Conclusão das experiências realizadas com base na Taxa de Acerto de cada modelo.

De forma a se ter uma melhor visão dos resultados obtidos para as diferentes experiências, é apresentado de forma resumida na Figura 63 a taxa de acerto obtida para os modelos apresentados neste capítulo.

Modelo	Algoritmo de Classificação			Técnicas de Extração de Atributos		Tecnicas de Pré-Processamento					Exatidão (%)
	Decision Trees (J48)	Random Forest-200 Árvores	Support Vector Machine	Motifs	Mfcc	Filtro Passa Baixo	Filtro Passa-Banda	*Decimate*	Envelope Shannon	Normalização	
Modelo A	X				X					X	60,2
Modelo B		X			X					X	74,8
Modelo C			X		X					X	56,3
Modelo D	X				X		X		X	X	30,1
Modelo E		X			X		X		X	X	34,9
Modelo E			X		X		X		X	X	40,8
Modelo F	X			X		X		X	X	X	19,4
Modelo G		X		X		X		X	X	X	31
Modelo H			X	X		X		X	X	X	26,2
Modelo I	X			X			X		X	X	18,4
Modelo J		X		X			X		X	X	22,3
Modelo K			X	X			X		X	X	26,2
	Decision	Support	Random Forest -								

Figura 63 - Quadro resumo dos resultados obtidos em função da taxa de acerto.

Como se pode constatar, observando o quadro da Figura 63 o modelo (B) consegue a maior taxa de acerto. O modelo B está constituído por:

- **Técnica de pré-processamento**: Normalização do sinal original.
- **Técnica de Extração de Atributos**: MFCCs.
- **Algoritmo de classificação**: *Random Forest* com 200 Árvores.

O modelo (B) apresenta uma taxa de acerto de 74,8 %, correspondente ao melhor valor obtido dos testes realizados. Em contrapartida o modelo (I) apresenta o pior resultado com uma taxa de acerto igual a 18,4%, modelo este constituído por:

- **Técnica de pré-processamento**: Aplicação de filtro passa-banda (1500Hz-18000Hz), normalização do sinal e posteriormente cálculo do envelope de Shannon.
- **Técnica de Extração de Atributos**: *Motifs.*
- **Algoritmo de classificação**: *Decision Tree* (j48).

Em relação ainda a outras medidas de desempenho como (*Precision, Recall e F-measure*), é possível constatar que, em média, os resultados obtidos nas experiências efetuadas pelos modelos baseados na abordagem dos MFCCs são melhores que os modelos baseados na abordagem *motifs*.

Assim, mediante os resultados obtidos nas experiências descritas (Figura 64), para a construção da solução de identificação automática de espécies de pássaros utilizando registo áudio foi utilizado o modelo B.

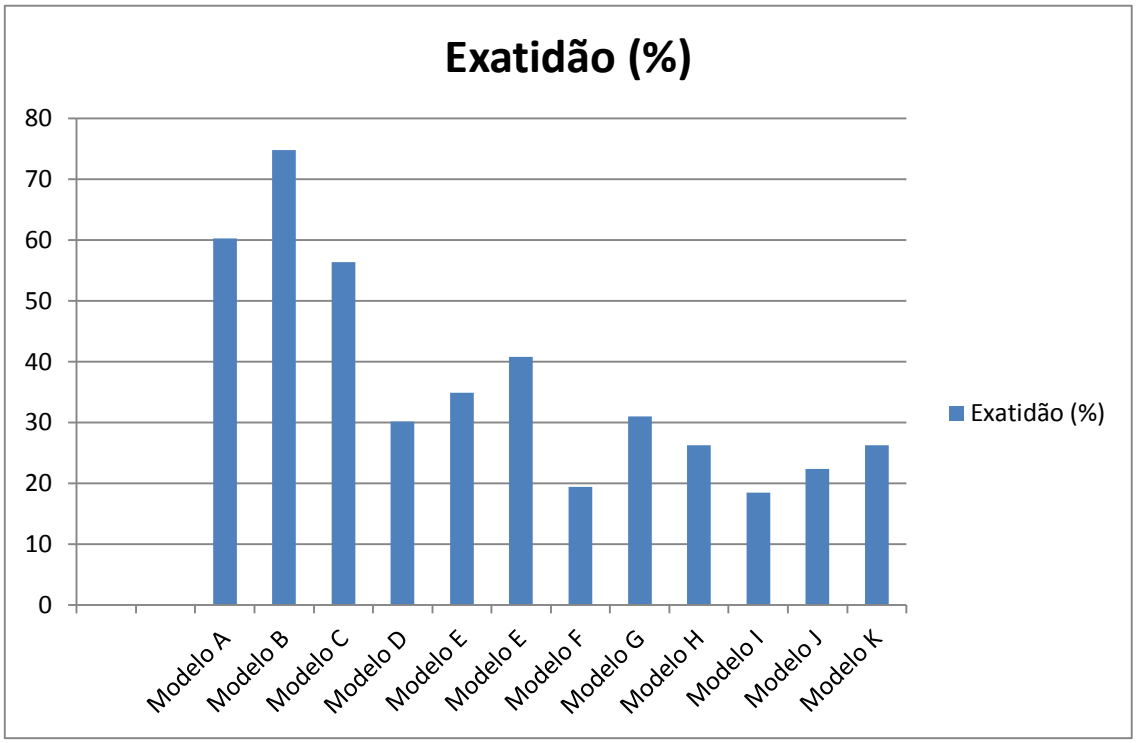


Figura 64 - Resultados obtidos apresentados por meio de um gráfico.

6.2 Testes e Avaliação da Aplicação Móvel

Nesta secção, é apresentada a metodologia de avaliação de desempenho da plataforma Cliente/Servidor.

Para efetuar o teste foram utilizados os seguintes equipamentos:

- HUAWEIi Ascend P7 – Com a aplicação cliente "Bird Sound Identification".
  - Sistema Operativo*: Android* OS, v4.4.2 (KitKat)
  - Processador: Quad-Core 1.8 GHz Cortex-A9
  - RAM: 2 GB

- TOSHIBA TECRA W50-A-117 – Com a aplicação Servidor.
  - Sistema Operativo: Windows 10 Pro, 64 bits
  - Processador: Intel Core i7-4810MQ vPRO 2.80/3.80 Turbo GHZ, Cache: 6MB
  - RAM: 32 GB

Para o teste de desempenho do sistema foram retirados 18 sons dos dados de treino e enviados para o dispositivo móvel.

Estando ambos os dispositivos na mesma rede, através da funcionalidade *MenuPopUp* da aplicação móvel foi possível realizar a identificação individual de cada som e a cronometração do tempo de resposta.

Na Tabela 18, apresentam-se os resultados obtidos no teste de desempenho do sistema. Na referida tabela, apresenta-se para cada ficheiro, a classe obtida pela classificação (na coluna "Classe Obtida") e a classe real (na coluna "Classe Etiquetada"). Apresenta-se ainda o tempo de resposta da aplicação, em milissegundos.

Tabela 18 - Resultado do teste de desempenho do sistema.

Nº	**Nome do ficheiro**	**Classe Etiquetada**	**Classe Obtida**	**Tempo de resposta (milissegundos)**
1	1475854258516.wav	Carcan_call	Carcan_call	28.711
2	1475855762758.wav	Carcan_call	Carcan_call	26.960
3	1475855788738.wav	Carcan_call	Sylund_call	18.432
4	1475855794591.wav	Carcan_song	Carcan_song	16.888
5	1475855799355.wav	Carcan_song	Carcan_song	16.472
6	1475855815989.wav	Carcan_song	Sylcan_call	21.375
7	1475855821697.wav	Carcan_song	Carcan_song	15.032
8	1475855830728.wav	Galthe_call	Sylcan_call	26.304
9	1475855845479.wav	Oriori_call	Carcan_song	26.968
10	1475855850570.wav	Sylcan_call	Carcan_song	17.957
11	1475855856190.wav	Sylcan_call	Sylcan_call	16.899
12	1475855860910.wav	Sylcan_call	Sylcan_call	22.088
13	1475855866020.wav	Sylmel_call	Sylmel_call	28.520
14	1475855871672.wav	Sylmel_call	Sylmel_call	23.200
15	1475855877439.wav	Sylund_call	Carcan_song	08.191
16	1475855881231.wav	Sylund_call	Sylund_call	08.698
17	1475855886048.wav	Sylund_song	Sylund_song	16.560
18	1475855920775.wav	Sylund_song	Sylund_song	23.330

Como se pode observar, para este teste, o sistema apresentou uma taxa de acerto de 66.66 % com 12 predições corretas e um tempo médio de resposta de 20143 milissegundos (aproximadamente 20 segundos).

É de se salientar ainda que o tempo de resposta do sistema irá variar muito consoante a duração do som e da velocidade da conexão entre os dispositivos (Cliente/Servidor). Os sons utilizados para este teste tinham uma duração média de 5 segundos.

Dependendo das condições ambientais estimou-se que o tempo para que sejam carregadas as coordenadas GPS e as informações relevantes da gravação pode variar entre os 1576 milissegundos a 5350 milissegundos.

7 Conclusões

Este trabalho apresenta de fato um grande contributo para a área da bioacústica, no que diz respeito a reconhecimento e identificação dos sons de pássaros.

Dos resultados obtidos nas experiências realizadas, foi possível observar que a abordagem baseada nos atributos MFCC obtém maior capacidade para discriminar as espécies dos pássaros através dos registos áudios do que a abordagem baseada nos atributos *motifs.*

A solução desenvolvida, por apresentar uma taxa de acerto igual a 74,76 %, demonstra que a mesma serve de ferramenta de base para o controlo e gestão das espécies de pássaros endémicas nas ilhas de S.Tomé e Príncipe, país pelo qual o autor reside e pretende fazer o sistema funcionar.

De momento o sistema não tem a capacidade de permitir a expansão do *dataset*, visto que para realizar essa tarefa é necessário o *feedback* do utilizador, o que apresenta um grande risco para a integridade do *dataset*. No entanto, como os registos áudio colhidos pelos utilizadores bem como as coordenadas e outas informações de interesse são guardadas numa base de dados, estes por intervenção do pessoal competente na área (Biólogo) poderão ser adicionados aos dados de treino diminuindo o risco de afetar a integridade do *dataset.*

Sendo assim propõe-se como trabalho futuro a construção de um módulo que permita a expansão do *dataset*. Aumentando o *dataset*, pretendemos também encontrar uma metodologia que permita discriminar melhor as classes dos pássaros, melhorando o desempenho da solução.

Referências

[Abe and Yamaguchi, 2005] Abe, H. & Yamaguchi, T., 2005. *Implementing an Integrated Time-Series Data Mining Environment - A Case Study of Medical KDD on Chronic Hepatitis.* s.l., 1st Internacional Conference on Complex Medical Engineering (CME 2005).

[Aide, Corrada-Bravo,Campos-Cerqueira,Milan,Vegae Alvarez, 2013] Aide, T Mitchell, Corrada-Bravo, Carlos, Campos-Cerqueira, Marconi, Milan, Carlos ,Vega,Giovany and Alvarez, Rafael. Real-time bioacoustics monitoring and aotomated species identification.2013.

[Antas e Cavalcanti,1988] Antas, P. de T. Z. ; Cavalcanti, R. B. Aves Comuns do Plananlto Central.Universidade de Brasília,1988.

[Batista, 2015] Batista, Fábio Miguel Moreira. Classificação de sons urbanos usando motifs e MFCC. 2015.

[Beritelli, 2008] Beritelli, F., 2008. *A Pattern Recognition System for Environmental Sound Classification based on MFCCs and Neural Networks.* s.l., ICSPCS.

[Bhargava e Sharma, 2013] Neeraj Bhargava, Girja Sharma. Decision Tree Analysis on J48 Algorithm for Data Mining, Volume 3. International Journal of Advanced Reserarch in Computer Science and Software Engineering. 2013.

[Bing Liu, 2007] Bing Liu.Web Data Mining, Exploring Hyperlinks,Contents, and Usage Data. Department of Computer Science University of Illinois at Chicago.2013.

[Boser et al., 1992] Boser, B., Guyon, I. & Vapnik, V., 1992. *A Training Algorithm for Optimal Margin Classiers.* s.l., COLT '92 Proceedings of the fifth annual workshop on Computational learning theory, pp. 144-152.

[Breiman , 1996] Breiman,L. Bagging predictors. Machine Learning.p 123-140. 1996.

[Breiman, 2001] Breiman, L., 2001. Random Forests. *Machine Learning*, pp. 5-32.

[Campbell., 1997] Campbell, Joseph P. Speaker recognition : A tutorial. Proceedings of the IEEE, p. 1437-1462, 1997.

[Carvalho e Machado, 2011] Carvalho, Geranado Aparecido e Machando, Paulo César Miranda. Pré-processamento de sons para Reconhecimento Automático de Pássaros. UFG, Goiânia, Escola de Engenharia Elétrica e de Computação. 2011.

[Castro and Azevedo, 2010] Castro, N. & Azevedo, P., 2010. *Multiresolution Motif Discovery in Time Series.* Columbus, Ohio, s.n., pp. 665-676.

[Chung, Oh, Lee, Park, Chang, Kim, 2013]. Yongwha Chung, Seunggeun Oh, Jounguk Lee, Daihee Park, Hong-Hee Chang and Suk Kim, 2013. Automatic Detection and Recognition of Pig Wasting Diseases Using Sound Data in Audio Surveillance Systems.

[Conceição ,2015] Paulo Francisco da Conceição ,2013. Reconhecimento Automático de Aves da Família Tinamidae Através da Vocalização. Pós Graduação em Engenharia Elétrica e de Computação

[Davis and Mermelstein, 1980] Davis, S. & Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Trans. on ASSP 28*, pp. 357-366.

[Efron ., 1979] Efron, B. Bootstrap Methods: Another Look at the JackKnife. The Annals of Statistics.p 1-26. 1979.

[Gama et al., 2012] Gama, J. et al., 2012. *Extração Conhecimento de Dados.* ed. 1, 1 vol., ISBN: 978-972-618-698-4 ed. Lisboa: Silabo.

[Gangajot e Chhabra , 2014] Gaganjot Kaur e Amit Chhabra. improved J48 Classification Algorithm for the Prediction of Diabetes. Interntional Journal of Computer Apllication (0975- 8887) Volume 98- No.22, 2014.

[Gelling,2010] Gelling, Douwe. Bird Sound Recognition using GMMs and Hmms.Master project Dissertation.2010.

[Gomes and Batista, 2015a] Gomes, E. F. & Batista, F., 2015. Classifying Urban Sounds using Time Series Motifs. *Advanced Science and Technology Letters,(SUComS),97*, pp. 52-57.

[Gomes and Batista, 2015b] Gomes, E. F. & Batista, F., 2015. Using Multiresolution Time Series Motifs to Classify Urban Sounds. *International Journal of Software Engineering and Its Applications,9,8*, pp. 189-196.

[GunaseKaram and Revathy, 2011] S. GunaseKaram and K Revathy, 2011. Automatic Recognition and Retrieval of Wild Animal Vocalizations.Internacional Journal of Computer Theory and

Engineering, Vol 3, No.1,February, 2011 1793 -8201.

[Hall et al., 2009] Hall, M. et al., 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations, Volume 11, Issue 1*.

[Hamid et al., 2005] Hamid, R. et al., 2005. *Unsupervised Activity Discovery and Characterization From Event-Streams.* s.l., Proc. of the 21st Conference on Uncertainty in Artificial Intelligence (UAI05).

[Han and Kamber, 2006] Han, J. & Kamber, M., 2006. *Data Mining: Concepts and Techniques.* 2nd Edition s.l.:ELSEVIER.

[Ho ., 1995] Ho, Tin Kam. Random Decision Forests. Proceedings of the 3rd International Conference on Document Analysis and Recognaition,Montreal.p278-282.1995.

[Huang, Yang, Yang e Chen, 2009] Huang, Chenn-Jung, Yang, Yi-Ju, Yang, Dian-Xiu and Chen, You-Jia. Frog classification using machine learning techniques.2009.

[Kumar, 2006] Kumar, Ajith PC e Ananthapadmanabha, TV. Heart Rate Variability using Shannon Energy, vol 5, No 2. 2006.

[Lee e Huang, 2006] Lee, Chang-Hsing, Lee, Yeuan-Kuen and Huang, Ren-Zhuang. Automatic Recognition of Birds Song Using Cepstral Coefficientes. Journal of Information and Applications. Vol 1 . No 1 .2006

[Liang, 1997] H Liang, S Lukkarinen, e I hartimo. Heart Sound segmentation algorithm based on heart sound envelogram. In computers in Cardiology, volume 24. P 105-108. 1997.

[Lin et al.,2002] Lin, J., Keogh, E., Patel, P. & Lonardi, S., 2002. *Finding motifs in time series.* Edmonton, Alberta, Canada, Proceedings of the 2nd Workshop on Temporal Data Mining, at the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

[Liu, 2007] Liu, Bing. Web Data Mining, Exploring Hyperlinks, Contents, and Usage Data. 2007.

[Lopes, Kaestner,Silla e Koerich, 2011] Lopes, Marcelo T, Kaester, Celso A. A., Silla, Carlos, N, Koerich, Alessandro. Identificação Automática de Espécies de Pássaros a partir da Análise do Canto. 2011

[Lyons,a] Lyons, J., 2015. A tutorial on Cepstrum and LPCCs. [Online] Available at: http://practicalcryptography.com/miscellaneous/machine-learning/tutorial-cepstrum-and-lpccs/

[Lyons,b] Lyons, J., 2015. Mel Frequency Cepstral Coefficient(MFCC). [Online] Available at: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[Marques, 2008] Marques, A. B. Abordagens sobre a bioacústica na ornitologia –conceitos básico, Universidade Estadual do Norte Fluminense, 2008.

[Marques, 2009] Marques, A. B. Tese de Doutorado. Avaliação do Canto do Trinca-Ferro (Saltator similis lafresnaye e dórbigny 1837) em Relação ao Processo de Domesticação e suas Implicações na Conservação das Aves Canoras. Universidade Estadualdo Norte Fluminense,2009.

[Marques, 2013] Marques, Nuno Miguel Santos, Heart Sound Segmentation : A Stationary Wavelet Transform Based Approach. 2013.

[McGovern et al., 2007] McGovern, A. et al., 2007. *Anticipating the formation of tornadoes through data mining.* s.l., Conference on Artificial Intelligence and its Applications to Environmental Sciences at the American Meteorological Society.

[Mendes,2011] Diana Rocha Mendes. Reconhecimento de Orador em Dois Segundos. Feup, dissertação para obtêncão de grau em mestrado em Engenharia Electrotécnica e de Computadores.2011.

[Minnen et al., 2006] Minnen, D., Starner, T., Essa, I. & Isbell, C., 2006. *Discovering Characteristic Actions from On-Body Sensor Data.* s.l., 10th International Symposium on Wearable Computers (ISWC06)..

[Mitrovic,Zeppelzauer e Breiteneder, 2006] Motrovic, Dalibor, Zeppelzauer, Mathias and Breiteneder, Christian. Discrimination and Retrieval of Animal Sounds.2006.

[Oliveira,2014] Oliveira, Soraia Carvalho da Cruz. Classificação de sons cardíacos usando motids. 2014.

[Oliveira et al., 2014] Oliveira, S. C., Gomes, E. F. & Jorge, A. M., 2014. *Heart Sounds Classification using Motif based Segmentation.* IDEAS '14 Proceedings of the 18th International Database Engineering & Applications Symposium, s.n., pp. 370-371.

[Oshiro, 2013] Oshiro, Thais Mayumi. Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica.2013.

[Pereira,2011] Susana Maria Ferreira Pereira,2011. A influência da Bioacústica na Evolução da Ciência em Portugal. Interface

da bioacústica e monitorização da biodiversidade. 13 p.

[Petry, Zanus e Barone,1999] Petry, Adriano., Zanus, Adriano e Barone, Dante Augusto Couto. Utilização de técnicasde processamento digital de sinais para a identificação automática de pessoas pela voz. 1999.

[Quinlan ., 1993] Quinlan, J.Ross. C4.5 Programs for Machine Learning. 1993.

[Sharma, Saha e Kumari , 2014] Sharma, Praveen K, Saha,Sourav e Kumari, Saraswati. Study and Design of a Shannon-Energ-Envelope based Phonocardiogram Peak Spacing Analyseesis for Estimating Arrythmic Heart Beat.Internacional Journal os Scientifcs and Research Publication, vol 4.2014

[Sick, 1984] Sick, H. Ornitologia Brasileira. Universidade de Brásilia,1984.

[Sick,1997] Sick, H. Ornitologia Brasileira Rio de Janeiro: Ed. Nova Fronteira, 1997. 912 p.

[Stevens, Volkmann and Newmn,1937] Steven, Stanley Smith ; Volkmann; John ; & Newmann,Edwin B. 19937. " A scale for the measurement of the psychological magnitude pitch". *Journal of the Acoustical Society of America 8 (3).* p 185-190.

[Vielliard 2009] Jacques Vielliard e Maria Luisa da Silva. A Bioacústica como ferramenta de pesquisa em Comportamento animal. 2009.

[Witten, 2005] Witten, I. H. & Frank, E., 2005. Data Mining: Practical Machine Learning Tools and Techniques.

Referências URL

[Android Developers, a] Android Developers, s.d. *Android Studio.* [Online] Available at: https://developer.android.com/tools/studio/index.html [Acedido em 13 Outubro 2015].

[Android Developers, b] Android Developers, s.d. *Android Developer Tools.* [Online] Available at: https://developer.android.com/tools/help/adt.html [Acedido em 13 Outubro 2015].

[Biomania,2015] Biomania,2015. Portal biológico [Online] Available at : http://www.biomania.com.br/bio/conteudo.asp?cod=3325

[Curió,2016] Curió, Sitio do Curió, 2016. [Online] Available at: ttp://www.sitiodocurio.com.br/si/site/004205

[Guru] Guru, SVM trained with samples from two classes. [Online]

Available at: http://blog.pengyifan.com/tikz-example-svm-trained-with-samples-from-two-classes/

[Acedido em 05 de Fevereiro 2016]

[IDC] International Data Corporation, Analyze the Future.[Online] Availabel at : http://www.idc.com/prodserv/smartphone-os-market-share.jsp [Acedido em 18 de Janeiro 2016]

[Kaggle] Kaggle, 2016.Your Home for Data Science. [Online] Availabel at: https://www.kaggle.com/c/multilabel-bird-species-classification-nips2013/data.

[Lee] Lee, M., 2006. *Sun begins releasing Java under the GPL.* [Online] Available at: http://www.fsf.org/news/fsf-welcomes-gpl-java.html

[Lyons] Lyons, J., 2015. *Mel Frequency Cepstral Coefficient (MFCC) tutorial.* [Online]
Available at:
http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[NetBeans] NetBeans, 2015. *NetBeans IDE - Overview.* [Online]
Available at: https://netbeans.org/features/index.html

[Oracle, a] Oracle, 2015. *O que é a Tecnologia Java e porque preciso dela?.* [Online]
Available at:
http://www.java.com/pt_BR/download/faq/whatis_java.xml

[Oracle, b] Oracle, 2015. *Obtenha Informações sobre a Tecnologia Java.* [Online]
Available at: http://www.java.com/pt_BR/about/

[Oracle, c] Oracle, 2015. *Oracle and Sun Microsystems.* [Online]
Available at: http://www.oracle.com/us/sun/index.html

[publico.pt , 2016] Público.2016. As últimas notícias, opinião, fotos e vídeos de Lisboa, Porto, Portugal e do Mundo. [avaible ate: https://www.publico.pt/ciencia/noticia/quatro-aves-unicas-de-sao-tome-e-principe-a-beira-da-extincao-1695230] [Acedido em 06 de Fevereiro de 2016]

[SAVE ,2015] Sociedade para a Conservação das Aves do Brasil
[Online]
Avaible at :
http://www.savebrasil.org.br/?q=content/por-que-aves

[Acedido em 15 de Novembro de 2015]

[Unicamp, 2015] Universidade Estadual de Campinas. Fonoteca Neotropical Jacques Vielliard. [Online] Available at : http://www.ib.unicamp.br/museu_zoologia/colecao_sonora [Acedido em 15 de Dezembro de 2015].

[WebM] WebM, s.d. *WebM - an open web media project.* [Online] Available at: https://code.google.com/p/webm/ [Acedido em 5 Agosto 2015].

[weka,2016] Weka, 2016- Weka classifier functions. [Online] Available at: http://weka.sourceforge.net/doc.stable/weka/classifiers/functions/LibSVM.html

[Wikiaves] Wikiaves. A vocalização das aves. [Online] Available at: http://www.wikiaves.com.br/a_vocalizacao_das_aves. [Acedido em 12 de Dezembro de 2015].

[Wikiaves, a] Wikiaves.Siringe [Online] Available at: http://www.wikiaves.com.br/_detail/areas:siringe.jpg?id=a_vocalizacao_das_aves [Acedido em 12 de Dezembro de 2015].

[Wikipedia] Wikipedia. Precision and Recall [Online] Available at: https://en.wikipedia.org/wiki/Precision_and_recall [Acedido em 11 de Outrubro de 2016].