Development of neural network-based speech/non-speech discrimination algorithm for audio files

by

HERMAN REDELINGHUYS

submitted in accordance with the requirements

for the degree of

MAGISTER TECHNOLOGIAE

in the subject

ELECTRICAL ENGINEERING

at the

UNIVERSITY OF SOUTH AFRICA

SUPERVISOR: Prof Z WANG

(JANUARY 2022)

Declaration

I declare that ***Development of neural network-based speech/non-speech discrimination algorithm for audio files*** is my own work and that all the sources that I have used or quoted have been indicated and acknowledged by means of complete references. I further declare that I have not previously submitted this work, or part of it, for examination at UNISA for another qualification or at any other higher education institution.

Herman Redelinghuys

26th day of October 2021

Acknowledgements

I would like to thank Professor Zenghui Wang, my supervisor, for his help and guidance with the dissertation.

I would like to recognize Professor Thomas Niesler from Stellenbosch University for his help to conceptualise the idea for the research and for valuable advice.

I would like to thank Professor Johann Mouton, my line manager for allowing my leave of absence to complete the study.

I wish to express my deepest gratitude to my family for their support and patience, especially Marietjie for her encouragement and inspiration.

Abstract

For research in speech processing and analysis of audio content in general, extensive data sets are required. Creating such a dataset manually turns out to be time and labour consuming, but it is a task very suitable for automation through machine learning. This dissertation describes the development of an algorithm for audio content analysis, discriminating between speech and music audio classes. To detect the different classes, the audio signal needs to be expressed in terms of its statistical properties based on spectral and temporal features and statistical values of these features are used to differentiate between the audio classes. The suitability of various low-level audio features was evaluated to determine suitability and efficiency in discriminating between different audio classes. To gain a better understanding of the feature set, exploratory dimensionality reduction analysis was performed with Principal Component Analysis (PCA). The mean accuracy for the SVM classifier was used to rank combinations of features. A number of feature selection techniques were employed to reduce the feature space and increase the mean accuracy, these included Univariate feature selection, Random Forest Regression, forward and backward Sequential Feature Selection and the highest loading factors of the PCA components.

An optimal subset of features was selected and used in evaluating a Neural Network-based classifier model and validated against a Support Vector Machine model. It was demonstrated that using this novel method of selecting the optimal combination of audio features, a 50% reduction in dimensionality and higher mean accuracy (99.95%) was achieved and proved to be well suited as a tool for extracting and compiling speech data sets.

The algorithm developed in this study is pertinent and applicable to the requirement of the initial motivation for the study, to efficiently create datasets for the study of language identification and recognition for African and other indigenous languages and beyond that, for analysing audio content in general. The reduction in the dimensionality of the feature space and consequently reduction in computational load should capacitate a real-time audio content analysis tool, implemented on low power IoT devices.

Keywords:

Speech discrimination, Audio Features, Machine learning, Short Time Energy Ratio, Zero-Crossing Rate, Spectral Roll-off, Spectral Flux, Spectral Centroid, Spectral Entropy, Mel Frequency Cepstral Coefficients, Neural Network, Support Vector Machine

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACA	Audio Content Analysis
ACF	Auto-correlation Function
AESR	Automatic Environmental Sound Recognition
AM	Amplitude Modulation
ASR	Automatic Speech Recognition
BFGS	Broyden–Fletcher–Goldfarb–Shanno
BN	Bayesian Network
CPU	Central Processing Unit
DC	Direct Current
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
DSP	Digital Signal Processing
EER	Equal Error Rate
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FM	Frequency Modulation
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
Hz	Hertz
HZCRR	High Zero-crossing Rate Ratio
IBM	International Business Machines Corporation
IoT	Internet of Things

kHz	kilohertz (10^3 Hz)
kNN	k-Nearest Neighbours
LDA	Linear Discriminant Analysis
LPC	Linear Predictive Coding
LPCM	Linear Pulse Code Modulation
LP-ZCR	Linear Prediction Zero-crossing Ratio
LSF	Line Spectral Frequencies
LSTER	Low Short-time Energy Ratio
MA	Mean Accuracy
MAP	Maximum A-Posteriori
ME	Modulation Energy
MFCC	Mel- Frequency (Cepstrum|Cepstral) Coefficients
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimator
MLER	Modified Low Energy Ratio
MLP	Multi-layer Perceptron
ms	millisecond
NAPS	Normalized Autocorrelation Peak Strength
NFR	Noise-frame Ratio
PCA	Principal Component Analysis
PDF	Probability Density Function
PHEQ	Polynomial-Fit Histogram Equalization
PLEF	Percentage Low Energy Frames
PSD	Power Spectral Density

PSR	Peak-to-Sidelobe ratio
QCG	Quadratic Gaussian Classifier
RBF	Radial Basis Function
RBFNN	Radial Basis Function (kernel) Neural Network
ReLU	Rectified Linear Unit
RFR	Random Forest Regression
RIFF	Resource Interchange File Format
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
SC	Spectral Centroid
SF	Spectral Flux
SFS	Sequential Feature Selection
SR	Spectral Roll-off
STE	Short Time Energy
STFT	Short-time Fourier Transform
SVM	Support Vector Machine
VDHMM	Variable Duration Hidden Markov Model
XOR	Exclusive OR (Logic)
ZCR	Zero-Crossing Rate

Mathematical Notation

x	Random variable sample value
$\boldsymbol{x}$	Random variable sample vector or matrix (bold font)
$\hat{y}$	Predicted value
X	Set of vectors
D	Dimension of the sample vector
k	(Algorithmic) index of a class
K	Number of classes
n	(Algorithmic) index of a sample
N	Normal (or Gaussian) distribution
$p(X)$	Probability density function (continuous values)
$P(X)$	Probability (mass) function (discrete values)
μ	Mean vector of a Gaussian distribution
σ	Variance of a Gaussian distribution
$\sum$	(variance-) covariance matrix of a Gaussian distribution
$\Sigma\|$	Determinant of a covariance matrix
θ	Full set of distribution parameters

List of publication(s)

Redelinghuys, H. and Wang, Z., Evaluating audio features for speech/non-speech discrimination, presented at ICAITPR2022: International Conference on Artificial Intelligence Trends and Pattern Recognition, Hyderabad, India, March 10-12, 2022

Chapter 1 - Introduction

1.1 Background

The field of speech recognition has been studied widely, however, the research focused mainly on a few major languages, specifically English. Recently, interest emerged in applying speech recognition to smaller indigenous languages. To develop and optimise a speech recognition engine, a researcher requires many hours of speech data to work on. For a small language without extensive libraries, this can be quite a challenge. In South Africa, many of our indigenous languages receive dedicated time allocations on national television service and most of them have dedicated local radio stations devoted to a specific language. Recording these audio sources and separating the speech and non-speech parts thereof, provides a simple and cheap method of compiling a dataset library for further analysis. The problem of discriminating between the different audio components in a noisy environment or where the audio components overlap has not been studied extensively. This research project aims to develop an algorithm for automatic speech / non-speech discrimination to annotate two classes of audio, speech and non-speech, including music, background noise and silence in audio files. Different audio features will be extracted, evaluated and statistically compared with different classifier algorithms. Feature extraction is a method of data reduction and consists of discovering the unique properties of each class.

For humans, the distinction between various types of sound is a very simple and involuntary process. According to Hepper and Shahidullah (1994), a human foetus responds to different frequency sounds from as early as 19 weeks of gestational age and infants usually utter their first words between 12 to 18 months old. The Cognitive Trade-off Hypothesis (Matsuzawa, 2007) suggest that our language facility was developed at the cost of short-term memory compared to other primate species. For human infants to master speech communication at a very early age, they need the ability to separate speech from environmental sound and other non-speech human sounds and so by nature humans became formidable at discriminating sound classes. To mimic this human ability with the use of software is no simple task. The first challenge is that humans perceive audio in the form of analogue signals in a non-linear way, while machines need to convert this audio signal to a

digital form. This is done by converting the analog signal to a continuous varying electrical signal. This signal is then converted to a sequence of discrete digital values by sampling the continuous signal at sufficiently regular intervals and storing a binary representation of the signal amplitude as a time series. The second challenge is to extract unique properties from the digital representation of the signal that can aid in identifying the various classes of sounds. This is the domain of the study of Audio Content Analysis (ACA). Lavner and Ruinskiy (2009) and Burred and Lerch (2004) describe audio content analysis as a field of research within machine learning, also called computer audition or machine listening, which deals with the extraction of information from audio signals through statistical modelling of data and applications of probability and decision theory with the aim of classification of sound categories. It combines knowledge and methods from a variety of disciplines including signal processing, machine learning, sound perception (psychoacoustics) and cognitive musicology (Ajmera, McCowan and Bourlard, 2003)(Lavner and Ruinskiy, 2009)(Burred and Lerch, 2004).

Applications for speech/music discrimination include automatic speech recognition (ASR) systems, transcription systems for broadcast audio, and variable or low bit-rate audio coding. Traditionally, separate codec designs are used to digitally encode speech and music signals. An effective speech/music discrimination decision will enable these to be merged into a universal coding scheme capable of reproducing well both speech and music. (Ajmera, McCowan and Bourlard, 2003)

Machine learning is a branch of artificial intelligence that aims to create systems that automatically detect patterns in data, and use these to make predictions or decisions. These systems rely on programs with tunable parameters that can be adjusted to improve the performance of the system by adapting to previously seen data. Machine learning is not a new concept, but it has been gaining a lot of momentum in recent years. The resurging interest in machine learning is due to factors like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage. Cloud computing allows engineers to access vast processing resources on demand while distributed computing like the internet of things (IoT) allows machine learning applications to be scaled out over a multitude of nodes. It is now possible to quickly and

automatically produce models that can analyse larger, more complex data and deliver more accurate results faster, resulting in predictions that can guide better decisions and smart actions in real-time without human intervention. This also benefitted the study of ACA, where a segmented audio stream, is analysed to identify a sequence of audio segments exhibiting common properties or features. These coherent features are then either grouped together (clustered) or through a supervised learning process, taxonomically classified.

1.2 The research problem

To achieve the goal of developing a neural network-based speech/non-speech discrimination algorithm for audio files using audio content analysis, a set of challenges needs to be solved. These challenges can be grouped and summarised in the following criteria:

- The necessity to research and evaluate various low level statistical audio features to find a sub-set of features that describe the statistical differences between the audio classes accurately and efficiently.
- The need to design a classifier model that can accurately and efficiently discriminate between the audio classes. A software implementation of a multi-layer perceptron neural network classifier model was proposed. The model needs to be robust enough to adapt to various speech sources, male and female voices, and various music genres.
- To develop a software algorithm to extract a collection of statistical features from a particular audio signal. The audio signal needs to be processed and normalised before analysis can take place. To evaluate features and train a classifier algorithm, an annotated dataset (corpus) of multiple audio files, comprising a wide range of sources and genres representing diversity within the different audio classes is required. This dataset should contain a balanced representation of the classes to be evaluated.

1.3 Research Objectives

This research project aims to develop an algorithm for automatic speech / non-speech discrimination to identify pure speech and non-speech classes in audio files. Non-speech can include music, background noise and silence. Different audio features will be extracted, evaluated and statistically compared to determine the best possible set of features and to evaluate with different

classifier algorithms. This study will focus on separating pure speech and music classes as well as removing silent segments in a pre-processing step.

- The first objective of this research will be to investigate numerous low-level features commonly used in audio content analysis to determine their theoretical suitability in discriminating between speech and non-speech audio classes. Some latest or less frequently used features will also be considered. The statistical properties of these theoretical features need to be extracted from an annotated dataset and this must be implemented in a software algorithm, as efficiently as possible.
- The second objective of this research will be to evaluate the list of features selected in the first objective individually and in combinations to determine suitability and efficiency in discriminating different audio classes. An optimal subset of features will be selected and used in a classifier model. The main audio classes that will be focused on are pure speech, music and low-level random noise, humanly perceived as silence.
- The third objective of this research will be to develop a multi-layer perceptron neural network classifier model and avoid using existing standard classification software libraries such as Theano and Tensorflow. The model will be evaluated and compared to a standard Support Vector Machine software library for suitability with the aim of implementation in the final algorithm.

1.4 Research methodology

This research builds on prior work performed in conjunction with the Stellenbosch University Department of Electrical Engineering's DSP Research Laboratory. This forms part of a study group focusing on speech and language processing, specifically, multilingual speech processing for Southern African languages. The preparation of datasets for this research, especially the manual extraction of speech data from mixed media recordings, is a laborious and time-consuming exercise, impeding the progress of the research. Typically the annotation ratio is > 1:10, meaning that it takes more than ten hours to accurately extract about one hour of speech data. This type of problem is particularly well suited to be solved by pattern recognition and automated by machine learning.

While researching the application of machine learning for speech and music processing, it became evident that while the fields of speech recognition and music genre classification have been researched frequently in the past, the study of speech/non-speech discrimination, especially the problem of discriminating between the different audio components in a noisy environment or where the audio components overlap have not been studied extensively. Most studies in the past focused on only a few audio features and their results in a mixture of audio components are often inconclusive or vague. The most comprehensive comparison of features and classification algorithms was done by Scheirer and Slaney (1997), who compared 13 temporal, spectral and cepstral features and four classification schemes, namely the Gaussian mixture model (GMM), K nearest neighbour (KNN), K-D trees and a multidimensional Gaussian MAP. Alexandre et al had done a study where they compared 14 different audio features (Alexandre *et al.*, 2008). Many other studies tend to focus on a limited feature set or one or more classification algorithms. A trend among these studies seems to be that the role of a strong feature set is just as important, if not more important than the classifier algorithm used, yet none of the studies performs a thorough process of selecting an optimal feature set prior to the classification step.

Features are extracted at the short term frame (few milliseconds) level and evaluated at the medium to longer-term segment level (few seconds). Frame level analytics represent a quasi-stationary position but to determine the semantic meaning of an audio signal, we need to observe the frame-level statistical variations over a longer period.

A classifier algorithm is required to discriminate between the different audio classes and for this study, a neural network algorithm was developed in Python and validated against a Support Vector Machine algorithm. The study also requires a dataset representative of all the audio classes and large and diverse enough to be able to use as a training and a test set without causing overfitting. An initial annotated training set is required to train the neural network and for this purpose a data corpus containing close to 7 hours, and 45 minutes of audio data was compiled, representing each class equally. The data was divided randomly into training and testing sets with k-fold iterations.

The output decision of the classifier will be compared to the values of an annotated test data set to calculate evaluation metrics whereby the performance of the algorithm can be measured.

This study adopted a positivist quantitative research paradigm. Figure 1.1 shows common research methods on a Positivist–Interpretivist axis, tending from quantitative to qualitative. Quantitative research methodology focuses on testing theories, determining facts, and demonstrating relationships between variables, thereby enabling the researcher to objectively predict outcomes (Van der Merwe, 1996). It is further the aim of the positivist approach to discover knowledge by controlled empirical means and experiments that are scientific, objective and reproducible (De Villiers, 2012).

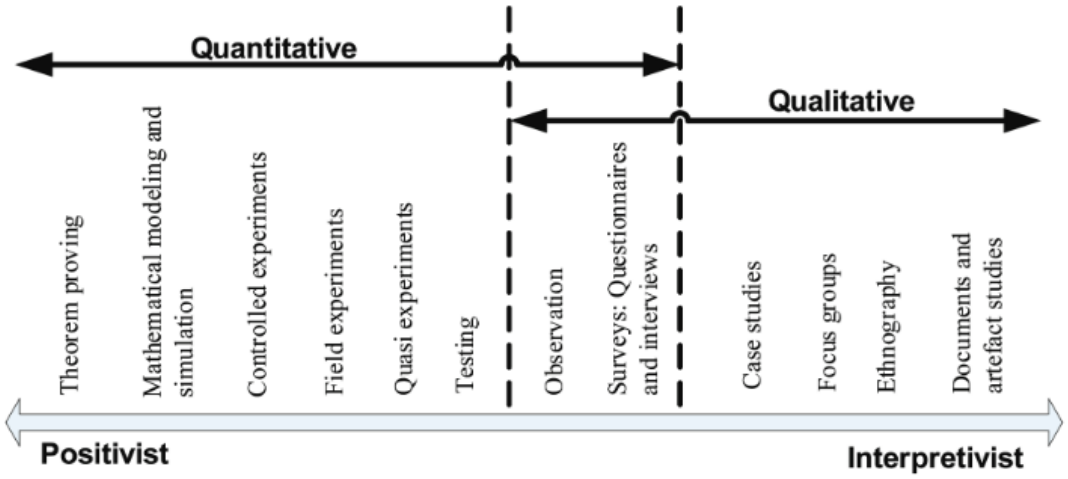

Figure 1.1: Research Methodologies (De Villiers, 2012)

A key principle of research is to first establish a solid theoretical foundation and understanding of the subject field. This often determines the direction of the research. For this purpose, an extensive and relevant literature review was undertaken to gain a thorough understanding of how similar research problems were dealt with by other scholars on the same subject and to gain a scientific explanation(s) and answers to the research questions. Through the literature review, a scholar should establish a body of knowledge necessary to answer the research questions and as a baseline

against which results can be evaluated. The literature review then enabled the student to decide on a preliminary feature set, a set of outcomes to strive towards and a method of evaluating results.

This leads to the design and implementation phase. The design phase included the collection of data and the preparation of the final corpus of data and the design of the software algorithm under development. The software algorithm comprises two software programs, the first to analyse data and compute statistical features and the other to evaluate statistical results in a reproducible fashion. These statistical results are then used to determine the optimal feature set and to compare classifier performance.

1.5 Scope

This study focuses on the discrimination of speech and music audio content as well as identifying silent segments of audio. The audio segmentation and classification of only two major audio classes were considered at this stage. Twelve (12) audio features were evaluated, namely, Root Mean Square value, Short Time Energy Ratio, Zero Crossing Rate, Spectral Rolloff, Spectral Flux, Spectral Centroid, Energy Entropy, Spectral Entropy and the first 13 Mel Frequency Cepstral Coefficients (MFCC). For each of these features, the first four standardised moments (mean, variance, skewness and kurtosis) were calculated and evaluated. A sub-set of features was selected after a rigorous evaluation process. A large, community-developed, open-source implementation of the Support Vector Machine algorithm was used to evaluate and select the feature set and was also used to benchmark the neural network algorithm. This algorithm was developed from the ground up.

1.6 Limitations

The project will only consider a limited number of audio classes, of which speech is the most important. It is however very simple to adapt the algorithm to other data classes or to focus on a specific class in more detail, for instance, to discriminate a specific genre of music or to discriminate between speaker gender classes.

1.7 Contribution of the study

While this is not the first time this topic has been studied, this study provides thorough and detailed selection criteria and a benchmark process for selecting the optimal set of features for speech / non-speech discrimination. This study aims to provide the most comprehensive analysis of statistical features used for the analysis of audio signals to date and will thus serve as a benchmark for other studies in this field on which audio features and statistical variants thereof to use.

The study also compares different classifier algorithms. The algorithm for the study was implemented in a high-level software language (Python) which can easily be implemented in other projects and on various platforms with minimal changes. Part of the selection criteria focused on the performance of the algorithm, cognitive of the need to conserve processing effort. This makes it ideal to be implemented on a low power embedded or mobile platform.

The algorithm developed in this study provides a building block for quickly and efficiently creating datasets for the study of language identification and recognition for African and other indigenous languages or for analysing audio content in general.

1.8 Overview of the dissertation

The research consists of six chapters.

Chapter 1 sets the background to the project by stating an explanation of the research problem, scope and objectives as well as a general description of the research design and methodology.

Chapter 2 provides an extensive review of the literature relevant to the research problem, providing an overview of earlier work done in the field of speech / non-speech discrimination and creating the foundation for solving the research objectives. The chapter begins with a brief discussion of audio theory related to the classes of data that will be analysed, followed by an explanation of the theoretical framework for audio content analysis. The next section of the chapter provides an in-depth discussion of several audio features specific to this project as well as a summary of previous research

on audio features. The chapter also provides an understanding of the decision and learning paradigms used in machine learning and how machine learning techniques work.

Chapter 3 continues from chapter 2 and further discusses the background and theory of the classification models applied in this research. It focuses on the Support Vector Machine (SVM) supervised machine learning classification algorithm which is used for the evaluation and selection of audio features as well as a multi-layer perceptron neural network classifier model used to evaluate the final audio feature set.

Chapter 4 explains the specific software implementation of the algorithm to process the audio data to extract the relevant audio features and the classification model.

Chapter 5 discusses the evaluation of the algorithm, feature selection process, including the performance on an individual level and the benchmarking of combinations of features to select a lower-dimensional feature set without sacrificing classification accuracy.

Chapter 6 discusses the outcomes achieved and notable conclusions drawn from the study as well as future ideas and possibilities.

Chapter 2 - Audio Content Analysis: Literature Review

2.1 Introduction

This study aims to develop a speech and music discrimination algorithm as a tool to isolate parts of speech in audio recordings to assist research focusing on speech and language processing, specifically, multilingual speech processing for Southern African languages. Audio signal content can be classified into different categories, for example, speech, music, background noise and silence (low-level white noise) or a combination thereof. For the purpose of this work, only two audio classes were considered, namely pure speech and music. As a first step in the process of speech and language processing is the discrimination of speech and non-speech and it is important to understand the theory behind audio content and the mathematical description of different audio classes, and in this case, specifically, silence, speech and music in terms of and how these features can be utilised through the process of pattern recognition and automated with machine learning to develop an algorithm that can effectively and accurately discriminate between these classes of audio. For this to happen, the audio signal needs to be transformed from a continuous or sampled representation of a continuous waveform that cannot be directly compared, to a sequence of statistical time series features in a reduced feature dimension.

2.2 Audio Classes

This study focuses on the discrimination of speech and non-speech audio classes. Only two classes were considered at this stage, however silent segments of audio were dealt with as a processing step similar to the approach followed by (Bugatti, Flammini and Migliorati, 2002). Other studies that considered the same audio classes include (Saunders, 1996), (Foote, 1997), (Scheirer and Slaney, 1997), (Balabko, 1999), (Carey, Parris and Lloyd-Thomas, 1999), (Williams and Ellis, 1999), (El-Maleh *et al.*, 2000), (Bugatti, Flammini and Migliorati, 2002), (Pinquier, Rouas and E-Obrecht, 2002), (Saad *et al.*, 2002), (Tzanetakis and Cook, 2002), (Ajmera, McCowan and Bourlard, 2003), (Wang, Gao and Ying, 2003), (Beierholm and Baggenstoss, 2004), (Burred and Lerch, 2004), (Panagiotakis and Tziritas, 2005), (Giannakopoulos, Pikrakis and Theodoridis, 2006), (Ericsson, 2009), (Lavner and Ruinskiy, 2009), (Muñoz-Expósito *et al.*, 2009), (Gallardo-Antolin and Montero, 2010), (Tardón, Sammartino and Barbancho, 2010), (Markaki and Stylianou, 2011). Some studies considered other

classes as well, for instance (Zhang and Jay Kuo, 2001; Lu, Zhang and Jiang, 2002; Alexandre *et al.*, 2006, 2008) also looked at environmental noise, while (Lu, Zhang and Li, 2003; Khan and Al-Khatib, 2006) looked at a class for speech with background music. A complete list of previous research and methods can be found in *Appendix A – Summary of previous studies*

2.2.1 Speech

Speech consists of syllables that are made up of consonant clusters separated by vowels. Vowels (or voiced sounds) are high energy events due to the vocal tract being fairly unrestricted during their formation. They have a periodic component due to the glottal excitation, which can be approximated by an impulse train in the time domain and by harmonics in the frequency domain (Saunders, 1996) and which is characterised by their intensity (loudness) and frequency (pitch). Most of the energy in voiced sounds is contained in the low-frequency portion of their spectra. Approximately two-thirds of all speech is voiced. (Bachu *et al.*, 2010). The voiced speech of a typical adult male will have a fundamental frequency between 85 Hz and 180 Hz, and that of a typical adult female between 165 Hz and 255 Hz (Traunmüller and Eriksson, 1994).

Consonants, in turn, are produced by frication, where constrictions in the vocal tract lead to turbulent airflow, damped resonances and the attenuation of acoustic energy. The resulting sound has higher "noise" content, with energy distribution at somewhat higher frequencies. The speech signal is also characterised by an energy modulation peak, specifically in the syllabic range (~2-5 Hz) (Edwards and Chang, 2013). Speech is thus composed of a collection of fairly typical sound characteristics and as such, can be represented well by statistical models.

2.2.2 Music

Music is a social and emotional activity unique to humans and played an important role in human civilisation throughout history to entertain, communicate emotion and construct social cohesion. Even though there is a multitude of music genres, music is universally understood and appreciated. The ancient Greek philosophers formulated the framework of the seven liberal arts consisting of a lower dimension, trívium - grammar, dialect, rhetoric, fundamentals underlying society and higher dimension quadrivium (Pythagoras around 500 BC) - arithmetic, geometry, astronomy, music as the

laws of nature and the universe and thereby acknowledging the existence of a correlation between music and mathematics. Pythagoras defined the discipline of music or harmony, as fundamentally, an application of mathematics and geometry, evolving in time. The way humans perceive music and machines processing music is therefore not completely unrelated. Human perception and classification of music can be described using various dimensions including genres, moods (emotions) and sound characteristics. Features extracted for machine learning are aimed at pattern recognition and describe quantitative characteristics of the audio signal, such as amplitudes, energies, and spectral characteristics. These technical attributes struggle to describe the emotional component associated with music and this is a general drawback of all machine-based classification systems.

Music audio signals can be described on a low level in 5 different characteristics (Lerch, 2012):

1. Statistical signal characteristics such as amplitude distribution, including the arithmetic, geometric and harmonic mean of the audio signal as well as its variance and deviation.

2. Timbre characteristics are indicative of sound quality. These describe the spectral shape and spread of the signal. The timbre characteristics can further be distinguished by timbre quality and timbre identity. The timbre quality refers to different sounds originating from the same source such as two recordings made with the same instrument. Timbre identity enables the identification of the instrument family of two sounds with the same tone characteristics originating from different sources, for instance, two violins of different quality. The quality of vocal sound is differing from one individual to another, but in vocal music, timbre properties are usually indicative of the expression of emotion. Low-level features typically associated with timbre characteristics are Spectral Rolloff, Zero-Cross and Spectral Centroid.

3. Features related to intensity and loudness. Intensity implies a physical, measurable entity such as the magnitude of a sound while loudness refers to how a human observer perceives the magnitude of the sound. The relationship between these attributes is non-linear; a linear increase in the signal's magnitude or power will not result in a linear increase in perceived loudness. Doubling the perceived loudness corresponds to a level increase of 10 dB in intensity. Root Mean Square (RMS) is the most common intensity feature.

4. Tonal and pitch characteristics are directly related to the frequency of a signal. The relation between the fundamental frequency and the perceived pitch is non-linear, due to the non-linearity of the frequency resolution of the human cochlea. At higher frequencies, two pitches with the same perceived pitch difference will have a larger frequency difference than at lower frequencies. Tonal and pitch characteristics are often used in music theory, most notably features based on the Mel scale including Mel-frequency cepstral coefficients (MFCC).

5. Temporal characteristics such as tempo and rhythm define events in time and are often described through beat histogram analysis.

2.2.3 Silence

Silence, for real-world audio processing, is referring to a low-level random signal (white noise) with a flat power spectral density. Speech with background music or random background noise is by far the most challenging to classify accurately. Background noise is usually non-periodic and consists of high energy events that are difficult to define.

2.3 Audio Content Analysis

The theoretical framework for audio content analysis was first described by Pfeiffer, Fischer and Effelsberg (1997) and alludes to the theoretical model of audio where the content of audio signal data can be assessed in two different ways, firstly in terms of physical properties and secondly in terms of the human cognitive properties thereof. The physical properties describe the waveform characteristics of the audio signal such as amplitude, frequency and phase, while the cognitive properties pertain to how we, as humans, perceive it. The most prominent perceptive properties are pitch and timbre.

Audio content analysis and discrimination algorithms generally consist of two stages. The first stage is a supervised learning phase, based on a statistical approach. In this phase labelled training data for each class that needs to be identified is analysed by the algorithm to build a statistical model of each of the classes and to define the separation boundaries (thresholds) between the different

classes. In the second stage, the processing phase, the test audio signal is analysed and the feature data for each frame is compared against the statistical model of each class and labelled by the classification algorithm.

The audio signal stream is analysed using the following steps as illustrated in Figure 2.1:

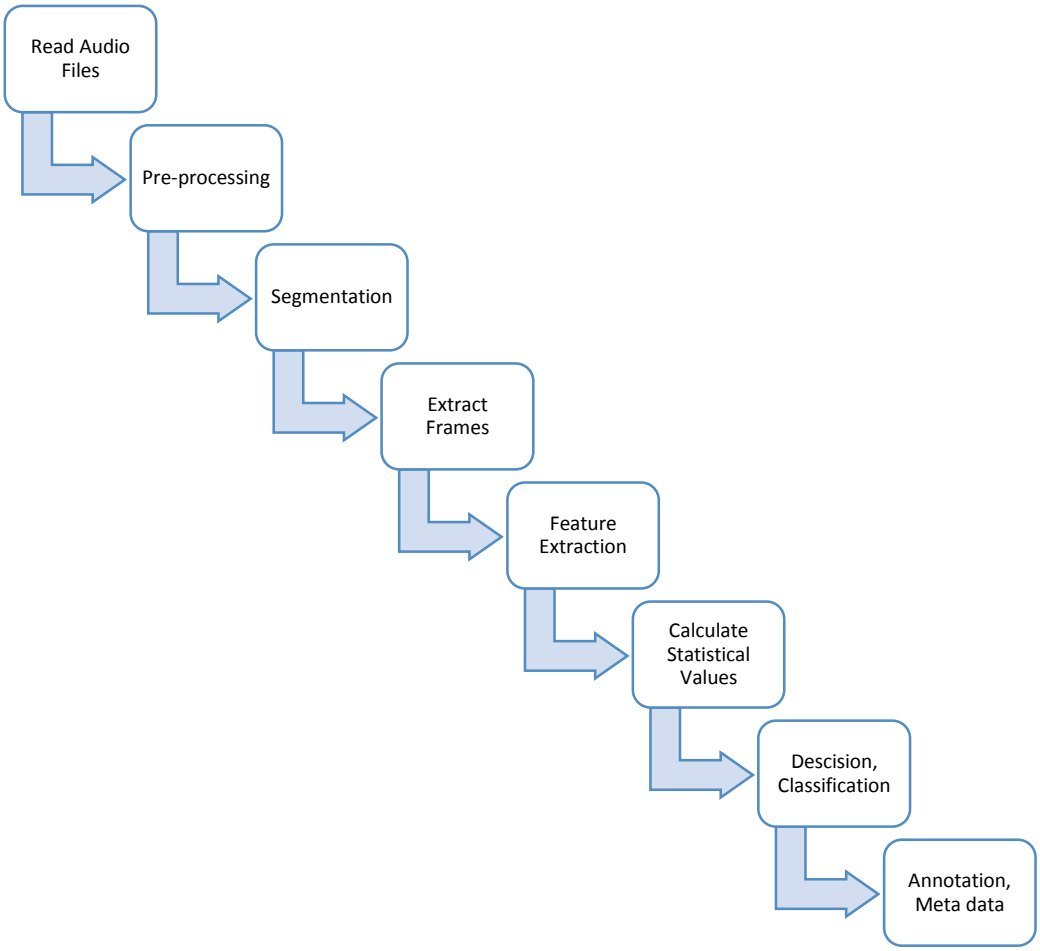

Figure 2.1: Processing stages for audio content analysis

2.3.1 Pre-processing

Pre-processing of the audio stream, including bandwidth filtering, dimensionality reduction and normalisation. As a first step, the audio files are homogenised to ensure a common format and sampling scheme. The files are converted Waveform Audio File Format, Microsoft and IBM originally developed WAV in 1991 for use within Windows 3.1. Derived from Resource Interchange File Format (RIFF) bitstream format method for storing data in indexed chunks, WAV audio format is

compressed or uncompressed audio in the loss-less linear pulse code modulation (LPCM) format. The data is then down-mixed and down-sampled to single-channel (mono) data with a 16 k sample rate. Down mixing stereo to mono is achieved by computing the arithmetic mean of all input channels according to (**2.1**).

$$x(i) = \frac{1}{Cx}\sum_{c=1}^{Cx} x_c(i) \tag{2.1}$$

Considering the Nyquist sampling theorem, this allows for audio data up to 8 kHz. A low-pass FIR filter (Hamming window) is applied to discard unwanted artefacts.

If the signal's arithmetic mean is not zero, it is indicative of a direct current (DC) offset. DC offset is removed before feature extraction by calculating the arithmetic mean of the signal and subtracting it from each sample (**2.2**).

$$x(i) = x_{DC}(i) - \frac{1}{I}\sum_{i=1}^{I} x_{DC}(i) \tag{2.2}$$

To compensate for varying audio amplitude levels in different audio recordings, the last pre-processing step is to normalise the signal level. This is done by scaling the signal so that the peak absolute value of the signal is mapped to the maximum value of the storage variable, which for 16 bits storage is 32767, thus the signal is scaled so that the largest absolute sample value is 32767.

2.3.2 Segmentation

Segmentation is dividing the audio stream into smaller segments and frames for analysis. Frames are short-term windows, generally with a duration of a few milliseconds. The optimal analysis window duration of 15 to 35 milliseconds results in a quasi-stationary representation of the audio signal which can, therefore, be processed through the short-time Fourier analysis. (Paliwal and

Wojcicki, 2008) Frames can either be overlapping (the frame step is shorter than the frame length) or non-overlapping. In this work the audio stream was divided into 1-second segments and 40-milliseconds frames, thus producing 25 frames per segment. Features derived on the frame level are referred to as short term or instantaneous features and serve as building blocks for the higher-level semantic description of the properties of an audio signal.

2.3.3 Feature Extraction

Extracting a series of features from these frames and aggregating statistical characteristics on a per-segment basis. Some features require evaluation over a longer duration, typically the length of the segment, which typically can be 1 to 10 seconds. The most common segment-based statistical features are the mean and variance of a full segment, as well as higher-order statistical properties such as skewness and kurtosis. Skewness is otherwise known as the third central moment of a signal divided by the cube of its standard deviation. It is a measure of the asymmetry of the probability density function (PDF) of the signal (**2.3**).

$$v_{SK}(n) = \frac{1}{\sigma_x^3(n).K} \sum_{i=0}^{n} (x(i) - \mu_x(n))^3 \tag{2.3}$$

For symmetric distributions, skewness = 0, while for negative value for left-skewed distributions and positive for 46-dimensional distributions. Kurtosis is the fourth central moment of a signal divided by the fourth power of its standard deviation (**2.4**).

$$v_K(n) = \frac{1}{\sigma_x^4(n).I} \sum_{i=0}^{n} (x(i) - \mu_x(n))^4 - 3 \tag{2.4}$$

Kurtosis, also called the fourth central moment of a signal, is an indication of the flatness/peakiness of a distribution compared to a Gaussian distribution (Lerch, 2012). A kurtosis value of 0 is

equivalent to a Gaussian distribution, a negative value is indicative of a flatter, wide peak shaped distribution (platykurtic), while a sharp, narrower peaked distribution is a positive value (leptokurtic).

Computationally, features can be categorised as either in the time domain (temporal) or frequency domain (spectral). Table 2.1 compares various audio features found in previous studies based on the signal domain they are in. Temporal features are usually easier to process because they do not need to be computed with a Fourier transform, however, if more than one spectral feature is employed, the Fourier transform needs to be calculated only once per frame and is then used by each feature.

Table 2.1: Comparison of audio features per signal domain

Temporal Features:	**Spectral Features:**
Energy Entropy	4Hz Modulation Energy
Root Mean Square / Short time Energy	Spectral Centroid
Zero-Crossing Ratio	Spectral Flux
Percentage Low Energy Frames / Short Time Energy Ratio	Spectral Rolloff
Modified Low Energy Ratio	Mel Frequency Cepstrum Coefficients
Dynamism	Chromatic Spectral Entropy
Loudness	Auto-correlation function
	Haar Discrete wavelet transform
	Spectral Flatness Measure

Audio instantaneous features are however also taxonomically categorised in terms of timbre, temporal, tonal and intensity/loudness characteristics. Timbre characteristics are indicative of sound quality. These describe the spectral shape and spread of the signal. Intensity and loudness related characteristics such as root mean square (RMS) values. Tonal and pitch characteristics are often used in music theory, most notably features based on the Mel scale including Mel-frequency cepstral coefficients (MFCC). Temporal characteristics, such as tempo and rhythm are important in onset detection and beat histogram analysis. (Lerch, 2012)

2.4 Audio features

Audio features can further be categorised based on their analysis method into either time-domain (temporal) or frequency domain features (spectral) (Giannakopoulos, 2015). Time-domain features are directly extracted from the raw signal samples within a specific analysis window. Time-domain features include Zero Crossing Rate, amplitude pitch and temporal features while frequency domain features include various spectral features, MFCC and entropy. Frequency domain features are derived from the magnitude of the Discrete Fourier Transform (DFT) of the audio signal. Mel-frequency cepstral coefficients are a variation of the frequency domain whereby applying the Inverse DFT on the logarithmic spectrum results in features existing in the cepstral domain. Audio features are also distinguished as either low level or high-level features. Low-level features are regarded as having no direct human interpretable meaning, whereas high-level features represent qualities that create structure in audio that humans can discern such as tempo (Lerch, 2012). Audio features regardless of which definition is applied can simply be considered a lower-dimensional representation of the original audio signal.

Audio instantaneous features are also taxonomically categorised in terms of timbre, temporal, tonal and intensity/loudness characteristics (Lerch, 2012). Many different features were used in previous studies as listed in Table 2.2. In most cases, the set of features is selected based on a hypothetical expectation of their individual strengths. It is interesting to note that only two studies explicitly compared different features to determine their suitability. Scheirer and Slaney (1997) compared the univariate discrimination performance of thirteen different temporal, spectral and cepstral

features, namely 4 Hz Modulation Energy, Percentage of Low-Energy Frames, Spectral Rolloff, Spectral Centroid, Spectral Flux, Zero Crossing Rate, Cepstrum Resynthesis Residual Magnitude and Pulse Metric (signal “rhythmicness”). The best individual features based on classification accuracy and computational overhead were then combined and compared with all features combined. In the study by Carey, Parris and Lloyd-Thomas (1999), it was hypothesised that pitch and cepstral coefficients might be a strong differentiator between speech and music classes. They selected eight features to compare, namely Cepstral Coefficients, Delta Cepstral Coefficients, Amplitude, Delta Amplitude, Pitch, Delta Pitch, Zero-Crossing Rate and Delta Zero-Crossing Rate of which Amplitude and Zero Crossing rate were the only non-spectral features but were included as they are computationally inexpensive. The authors evaluate and rank each feature individually, but do not compare them in combinations.

Table 2.2: Summary of audio features used in previous studies.

Spectral Rolloff	Spectral Centroid	Spectral Flux	Entropy	MFCC
Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Ajmera, J., McCowan, I., & Bourlard, H. (2003)	Ajmera, J., McCowan, I., & Bourlard, H. (2003)
Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Ericsson, L. (2009)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)
Barbedo, J. G. A., & Lopes, A. (2007)	Bugatti, A., Flammini, A., & Migliorati, P. (2002)	Barbedo, J. G. A., & Lopes, A. (2007)	Giannakopoulos, T. (2015)	Balabko, P. (1999)
Burred, J. J., & Lerch, A. (2004)	Burred, J. J., & Lerch, A. (2004)	Bugatti, A., Flammini, A., & Migliorati, P. (2002)	Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2006)	Bugatti, A., Flammini, A., & Migliorati, P. (2002)
Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006)	Ericsson, L. (2009)	Burred, J. J., & Lerch, A. (2004)	Pinquier, J., Rouas, J. L., & E-Obrecht, R. a. (2002)	Burred, J. J., & Lerch, A. (2004)
Giannakopoulos, T. (2015)	Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006)	Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006)	Williams, G., & Ellis, D. P. W. (1999)	Carey, M. J., Parris, E. S., & Lloyd-Thomas, H. (1999)
Lavner, Y., & Ruinskiy, D. (2009)	Giannakopoulos, T. (2015)	Giannakopoulos, T. (2015)		Ericsson, L. (2009)
Saad, E.M., El-Adawy, M.I., Abu-El-Wafa, M.E., Wahba, A.A. (2002)	Lavner, Y., & Ruinskiy, D. (2009)	Lavner, Y., & Ruinskiy, D. (2009)		Foote, J. T. (1997)
Scheirer, E.& Slaney, M., (1997)	Munoz-Exposito, J. E., Galan, S. G., Reyes, N. R., & Candeas, P. V. (2009)	Lu, L., Zhang, H.-J., & Li, S. Z. (2003)		Gallardo-Antolin, A., & Montero, J. M. (2010)
Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)	Saad, E.M., El-Adawy, M.I., Abu-El-Wafa, M.E., Wahba, A.A. (2002)	Saad, E.M., El-Adawy, M.I., Abu-El-Wafa, M.E., Wahba, A.A. (2002)		Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006)
Tardón, L. J., Sammartino, S., & Barbancho, I. (2010).	Scheirer, E.& Slaney, M., (1997)	Scheirer, E.& Slaney, M., (1997)		Giannakopoulos, T. (2015)
Tzanetakis, G., & Cook, P. (2002)	Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)	Tardón, L. J., Sammartino, S., & Barbancho, I. (2010).		Khan, M. K. S., & Al-Khatib, W. G. (2006)
	Tardón, L. J., Sammartino, S., & Barbancho, I. (2010).	Tzanetakis, G., & Cook, P. (2002)		Lavner, Y., & Ruinskiy, D. (2009)
	Tzanetakis, G., & Cook, P. (2002)			Lu, L., Zhang, H.-J., & Li, S. Z. (2003)
				Markaki, M., & Stylianou, Y. (2011).

				Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2006)
				Scheirer, E.& Slaney, M., (1997)
				Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)
				Tardón, L. J., Sammartino, S., & Barbancho, I. (2010).
				Tzanetakis, G., & Cook, P. (2002)
Zero-Crossing Rate	**RMS / Short Time Energy**	**PLEF / Short Time Energy Ratio**	**Modified Low Energy Rate**	**4 Hz Modulation Energy**
Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Ericsson, L. (2009)	Wu Chou, & Liang Gu. (2001)
Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Wang, W. Q., Gao, W., & Ying, D. W. (2003).	Pinquier, J., Rouas, J. L., & E-Obrecht, R. a. (2002)
Bachu, R. G., Kopparthi, S., Adapa, B., & Barkana, B. D. (2010).	Bachu, R. G., Kopparthi, S., Adapa, B., & Barkana, B. D. (2010).	Burred, J. J., & Lerch, A. (2004)		Scheirer, E.& Slaney, M., (1997)
Bugatti, A., Leonardi, R., & Rossi, L. (1999)	Bugatti, A., Leonardi, R., & Rossi, L. (1999)	Saad, E.M., El-Adawy, M.I., Abu-El-Wafa, M.E., Wahba, A.A. (2002)		
Bugatti, A., Flammini, A., & Migliorati, P. (2002)	Bugatti, A., Flammini, A., & Migliorati, P. (2002)	Lu, L., Jiang, H., Zhang, H.-J. (2001)		
Burred, J. J., & Lerch, A. (2004)	Burred, J. J., & Lerch, A. (2004)	Lu, L., Zhang, H.-J., & Li, S. Z. (2003)		
Carey, M. J., Parris, E. S., & Lloyd-Thomas, H. (1999)	Ericsson, L. (2009)	Scheirer, E.& Slaney, M., (1997)		
El-Maleh, K., Klein, M., Petrucci, G., & Kabal, P. (2000)	Foote, J. T. (1997)	Tzanetakis, G., & Cook, P. (2002)		
Ericsson, L. (2009)	Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006)			
Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006)	Giannakopoulos, T. (2015)			
Giannakopoulos, T. (2015)	Khan, M. K. S., & Al-Khatib, W. G. (2006)			

Khan, M. K. S., & Al-Khatib, W. G. (2006)	Lavner, Y., & Ruinskiy, D. (2009)			
Lavner, Y., & Ruinskiy, D. (2009)	Liu, Z., Wang, Y., & Chen, T. (1998)			
Lu, L., Jiang, H., Zhang, H.-J. (2001)	Panagiotakis, C., & Tziritas, G. (2005)			
Lu, L., Zhang, H.-J., & Li, S. Z. (2003)	Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2006)			
Panagiotakis, C., & Tziritas, G. (2005)	Saunders, J. (1996)			
Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2006)	Tardón, L. J., Sammartino, S., & Barbancho, I. (2010).			
Saad, E.M., El-Adawy, M.I., Abu-El-Wafa, M.E., Wahba, A.A. (2002)	Zhang, T., & Jay Kuo, C. C. (2001)			
Saunders, J. (1996)				
Scheirer, E.& Slaney, M., (1997)				
Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)				
Tardón, L. J., Sammartino, S., & Barbancho, I. (2010).				
Tzanetakis, G., & Cook, P. (2002)				
Zhang, T., & Jay Kuo, C. C. (2001)				
Dynamism	**Loudness**	**Auto-correlation function (ACF)**	**Haar Discrete wavelet transform**	**Spectral Flatness**
Ajmera, J., McCowan, I., & Bourlard, H. (2003)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Beierholm, T., & Baggenstoss, P. M. (2004)	Khan, M. K. S., & Al-Khatib, W. G. (2006)	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)
Williams, G., & Ellis, D. P. W. (1999)	Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)			Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)
	Barbedo, J. G. A., & Lopes, A. (2007)			Burred, J. J., & Lerch, A. (2004)
	Burred, J. J., & Lerch, A. (2004)			Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)

2.4.1 Silence removal

An important initial step in processing an audio signal is identifying and eliminating silence segments thereby discarding a large portion of worthless information. The method followed here distinguishes between high energy and low energy short term frames to identify silence. This is similar to the method proposed by Bugatti et al. and Giannakopoulos (Bugatti, Leonardi and Rossi, 1999), (Giannakopoulos, 2015). The criteria used in this work is where the Root Mean Square value is below 0.003% of the maximum signal energy.

2.4.2 Zero-Crossing Rate

The Zero Crossing Rate (ZCR) is a measure of how often the audio signal crosses the zero amplitude level, i.e. the amplitude of the signal changes sign per unit of time. The zero-crossing rate is an indicator of the frequency at which the energy is concentrated in the signal spectrum, thus an indication of the spectral content of the signal or the dominant frequency in the signal (Saunders, 1996), (Carey, Parris and Lloyd-Thomas, 1999), (Zhang and Jay Kuo, 2001) and (Burred and Lerch, 2004) used this feature along with short-time energy to obtain results (**2.5**).

The feature is defined as:

$$ZCR = \frac{1}{2}\sum_{n=1}^{N} |sgn(x[n]) - sgn(x[n-1])| \qquad (2.5a)$$

$$sgn(x) = \begin{cases} 1, x \geq 0 \\ -1, x < 0 \end{cases} \qquad (2.5b)$$

where $\boldsymbol{x[n]}$ is the discrete sound signal of length $\boldsymbol{N}$ measured in time and $\boldsymbol{sgn}(.)$ is a sign function.

The average zero-crossing rate is used to distinguish between voiced and unvoiced speech signals because unvoiced speech components normally have much higher ZCR values than voiced ones. The ZCR curve for speech indicates rapid variations alternating between unvoiced and voiced

components. This results in a large variance in amplitude for the speech ZCR curve. For pure speech (single voiced) it can be used as a weak estimation of half the fundamental glotteral frequency (approximately 40 Hz to 125 Hz). The zero-crossing rate for music is typically much more stable over time with a lower variation in amplitude. Higher ZCR values indicate a noisy, less periodic signal. For speech/music discrimination, the variance (2^{nd} central moment) and skewness (3^{rd} central moment) is useful, with higher variance and skewness likely indicating a speech signal. (Saunders, 1996) Due to its simplicity and low computational requirement, this feature is popular, especially for speech analysis and was also used by (Scheirer and Slaney, 1997; Bugatti, Leonardi and Rossi, 1999; El-Maleh *et al.*, 2000; Lu, Jiang and Zhang, 2001; Saad *et al.*, 2002; Tzanetakis and Cook, 2002; Bugatti, Flammini and Migliorati, 2002; Lu, Zhang and Li, 2003; Panagiotakis and Tziritas, 2005; Alexandre *et al.*, 2006, 2008; Pikrakis, Giannakopoulos and Theodoridis, 2006; Giannakopoulos, Pikrakis and Theodoridis, 2006; Khan and Al-Khatib, 2006; Lavner and Ruinskiy, 2009; Ericsson, 2009; Tardón, Sammartino and Barbancho, 2010; Bachu *et al.*, 2010; Giannakopoulos, 2015; Sigtia *et al.*, 2016).

2.4.3 Root Mean Square Energy (Short Time Energy)

The signal energy can be expressed in terms of its Root Mean Square energy value and is a measure of sound intensity. Short Time Energy (STE) is defined as the mean RMS energy of the signal within each analysis frame. Energy is the square root of power, which in turn is defined as the sum of the squares of the signal samples, averaged over the frame duration. This feature becomes computationally inefficient on large window sizes and is therefore only applied at the frame level. Bachu *et al.*, (2010) calculated the short time energy on a per-segment base using a Hamming window function, however, the choice of the window duration proved difficult, as the energy will fluctuate rapidly if it is too small and if too large, the energy will change very slowly and thus will not adequately reflect the signal properties.

The signal energy of an audio signal in one analysis frame is defined by Lerch as (**2.6**) and by Lavner and Ruinskiy (2009) and Lerch(2012) as the log of the power expressed in decibels (**2.7**).

$$E_{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} x_{(n)}{}^2} \tag{2.6}$$

where $\boldsymbol{N}$ is the number of samples within the frame and $\boldsymbol{x_{(n)}}$ is the sample value.

$$dB(E_{RMS}) = 10log_{10}\left(\frac{1}{N}\sum_{n=1}^{N} x_{(n)}{}^2\right) \tag{2.7}$$

For the same perceived loudness level, the RMS energy values of the frames for different audio classes can vary significantly. The energy contour of music tends to contain fewer and much less drastic variations than the energy contour of speech. The erratic energy contour of the speech signal is due to the alternation between voicing and frication. This feature is very useful for silence detection, but on its own, STE is a poor choice for speech/music discriminator and because it depends on the amplitude of the sample, it requires the speech and non-speech portions of the audio segments to be normalised first. Saunders found that using this as a secondary feature in conjunction with ZCR produced a marked improvement (±8%) in classification accuracy (Saunders, 1996). The STE for a segment will be very low or close to zero for silent audio passages. For music, the segment variation will be higher for speech than for music. This feature was used by (Saunders, 1996; Foote, 1997; Liu, Wang and Chen, 1998; Bugatti, Leonardi and Rossi, 1999; Zhang and Jay Kuo, 2001; Bugatti, Flammini and Migliorati, 2002; Burred and Lerch, 2004; Panagiotakis and Tziritas, 2005; Alexandre *et al.*, 2006, 2008; Pikrakis, Giannakopoulos and Theodoridis, 2006; Khan and Al-Khatib, 2006; Lavner and Ruinskiy, 2009; Ericsson, 2009; Tardón, Sammartino and Barbancho, 2010; Bachu *et al.*, 2010; Giannakopoulos, 2015). Some other features derived from the RMS Energy and STE are Loudness, Short-Time Energy Ratio (STER) and Percentage of Low Energy Frames (PLEF). Loudness is defined as an exponential function of the energy of the audio signal (**2.8**) (Alexandre *et al.*, 2006, 2008):

$$Loudness_t = RMS\ Energy_t^{0.23} \tag{2.8}$$

2.4.4 Percentage Low Energy Frames (PLEF) / Low Short Time Energy Ratio (LSTER)

This feature is derived from the STE and is defined as the ratio of frames with a mean STE less than 50% of the mean RMS energy within a segment (Scheirer and Slaney, 1997). The same feature is used by (Lu, Zhang and Li, 2003) for speech-music discrimination, but called as Low Short-Time Energy Ratio (LSTER). The value for speech is normally higher than that of music. (Saad *et al.*, 2002; Tzanetakis and Cook, 2002; Burred and Lerch, 2004; Alexandre *et al.*, 2006, 2008). The reason is that there are more quiet frames in speech as some pause between every word exists and hence the energy of the frame containing pauses is lower (Khan and Al-Khatib, 2006). According to Saunders (1996), the energy distribution for speech is more left-skewed than that of music.

2.4.5 Modified Low Energy Ratio

The feature Modified Low Energy Ratio (MLER) is introduced by (Wang, Gao and Ying, 2003) as a variation to the Percentage Low Energy Frames. The feature exploits the fact that music shows little variation in the energy contour of the waveform, while speech shows large variations between voicing and frication. MLER is defined as the proportion of frames with RMS power less than a variable threshold within one second. It is suggested by the authors that the threshold should be in the interval [0.05%, 0.12%] for best performance (**2.9**). This feature was also used by (Ericsson, 2009).

$$MLER = \frac{1}{2N}\sum_{n=1}^{N}\left|sgn\big(lowthres - E(n)\big) + 1\right| \qquad (2.9a)$$

$$lowthres = \delta.\frac{\sum_{n=1}^{N} E(n)}{N} \qquad (2.9b)$$

where N is the total number of frames in the segment, E(n) is the Short Time Energy of the nth frame, δ is a control coefficient, which is suggested in the interval [0.05, 0.12].

2.4.6 Energy Entropy and Chromatic Spectral Entropy

The term entropy was originally used by von Boltzmann in the 1870s in the field of thermodynamics to describe the statistical mechanics that quantifies the average disorder in an isolated system. Shannon's 1948 paper on the mathematical theory of information provides the foundation of the field of information theory in the context of communication systems and stated inter alia that entropy is a measure of the uncertainty or disorder in a given distribution and thus a more stochastic signal contains higher entropy. The information in a signal can be interpreted as the inverse of entropy and the negative logarithm of its probability. The application of entropy speech/music discrimination is based on the assumption that the speech spectrum is more stochastic than music and has a higher entropy value (Pinquier, Rouas and E-Obrecht, 2002).

Giannakopoulos differentiate between energy entropy and spectral entropy, with the former simply the entropy of the normalised energies (STE) of the individual frames Giannakopoulos, (2015). To determine the spectral entropy, calculate the entropy of the normalised Power Spectral Density (PSD) of the signal (**2.10**).

$$H_{PSE} = -\sum_{i=1}^{n} p_i \, lnp_i \qquad (2.10a)$$

here $p_i$is the normalised PSD:

$$p_i = \frac{P(w_i)}{\sum P\,(w_i)} \qquad (2.10b)$$

$$P(w_i) = \frac{1}{N}|X(w_i)|^2 \qquad (2.10c)$$

In Pikrakis, Giannakopoulos and Theodoridis (2006), the authors use a feature called Chromatic Entropy which is a variation of Spectral Entropy. The spectrum is first mapped to the Mel scale and then divided into twelve sub-bands (bins), with centre frequencies that coincide with the frequencies of the chromatic scale. The energy E_i of the i-th sub-band, $i = 0, \ldots, N-1$, is then normalized by the total energy of all the sub-bands, yielding $n_i = \frac{E_i}{\sum_{i=0}^{L-1} E_i}$, $i = 0, \ldots, N-1$

The entropy of the normalized spectral energy is now calculated (**2.11**) as per Pikrakis, Giannakopoulos and Theodoridis, (2006).

$$H = -\sum_{i=0}^{N-1} n_i \times log_2(n_i) \tag{2.11}$$

where $\boldsymbol{n_i}$ is the normalized energy of sub-band $\boldsymbol{i}$ and $\boldsymbol{N}$ is the number of sub-bands.

Ajmera, McCowan and Bourlard (2003) show that the average entropy for music is usually higher than for speech since the values of probabilities in music will be more uniformly distributed than in speech, thus resulting in a higher value for entropy (Pinquier, Rouas and E-Obrecht, 2002).

2.4.7 Spectral Rolloff

Spectral Roll-off (SR) is a representation of the spectral shape of an audio signal. It is defined as the frequency bin below which 95% of the energy in the power spectral distribution is concentrated according to (Scheirer and Slaney, 1997). However, some papers refer to the 85th percentile, e.g. (Reeder, 2001), (Burred and Lerch, 2004) and (Lerch, 2012) or the 90th percentile (Giannakopoulos, 2015) for each K where (**2.12**) applies.

$$f(k) = \sum_{k=0}^{i} |X[k]| \leq 0.85 \sum_{k=0}^{N/2-1} |X[k]| \tag{2.12}$$

Then, the Spectral Roll-off frequency is$f(K)$, where $|X[k]|$represents the magnitude of bin number k, and $f(k)$ represents the centre frequency of that bin. Voiced speech has a high proportion of energy contained in the high-frequency range of the spectrum, where most of the energy for unvoiced speech and music is contained in lower bands. As a result, the spectral roll-off value is higher values for music and unvoiced speech, and lower for voiced speech (Saad *et al.*, 2002). This feature was also used by Tzanetakis & Cook (2002); Alexandre et al. (2006, 2008); Giannakopoulos

et al. (2006); Barbedo & Lopes (2007); Lavner & Ruinskiy (2009); Tardón et al. (2010); Giannakopoulos (2015); Sigtia et al. (2016).

2.4.8 Spectral Flux

Spectral Flux (**2.13**) or Delta Spectrum Magnitude feature is a measure of the rate at which the spectral shape changes, or fluctuates. It is calculated by summing the squared differences of magnitude spectra of two neighbouring frames.

$$F = \sum_{k=1}^{\frac{N}{2}} (|X_r[k]| - |X_{r-1}[k]|)^2 \tag{2.13}$$

where $\boldsymbol{N}$ is the number of FFT points and $\boldsymbol{X_r[k]}$ is the STFT of frame $\boldsymbol{r}$ at bin $\boldsymbol{k}$.

Lu, Zhang and Li (2003) found this feature to be especially useful for detecting some strong

periodicity environment sounds such as tone signals, from music signals. Burred and Lerch (2004) found this a useful feature for speech/music separation, yielding higher values for music than speech, as speech signals exhibit more drastic frame-to-frame changes than music. Speech alternates between periods of transition (consonant-vowel boundaries) and periods of relative stasis (vowels), where music typically has a more constant rate of change (Khan and Al-Khatib, 2006) (Scheirer and Slaney, 1997).

This feature was also used by (Tzanetakis and Cook, 2002; Bugatti, Flammini and Migliorati, 2002; Saad *et al.*, 2002; Burred and Lerch, 2004; Alexandre *et al.*, 2006, 2008; Barbedo and Lopes, 2006; Giannakopoulos, Pikrakis and Theodoridis, 2006; Lavner and Ruinskiy, 2009; Tardón, Sammartino and Barbancho, 2010; Giannakopoulos, 2015).

2.4.9 Spectral Centroid

This feature describes the spectral shape of the audio signal, i.e. indicates the “balancing point” of the spectral power distribution. This feature is correlated with the psychoacoustic features'

sharpness and brightness which is related to the high-frequency content of the spectrum. There are several definitions of the Spectral Centroid (SC) feature in previous work. In (Ericsson, 2009) it is defined as a weighted mean of the frequencies in the FFT transform of the signal (**2.14**).

$$SC = \frac{\sum_{k=1}^{N/2} f(k)|X[k]|}{\sum_{k=1}^{N/2}|X[k]|} \tag{2.14}$$

where $\boldsymbol{N}$is the number of FFT points, $|\boldsymbol{X}[\boldsymbol{k}]|$represents the magnitude of bin number $\boldsymbol{k}$, and $\boldsymbol{f}(\boldsymbol{k})$ represents the centre frequency of that bin.

4 Hz Modulation Energy and Spectral Centroid are also used to separate voiced speech from noisy sound signals (Saunders, 1996). The spectral centroid for music is typically higher than that for speech. (Scheirer and Slaney, 1997; Saad *et al.*, 2002). This feature was also used by (Tzanetakis and Cook, 2002; Bugatti, Flammini and Migliorati, 2002; Burred and Lerch, 2004; Alexandre *et al.*, 2006, 2008; Giannakopoulos, Pikrakis and Theodoridis, 2006; Lavner and Ruinskiy, 2009; Muñoz-Expósito *et al.*, 2009; Tardón, Sammartino and Barbancho, 2010; Giannakopoulos, 2015; Sigtia *et al.*, 2016)

2.4.10 Mel Frequency Cepstral Coefficients (MFCC)

The human ear detects small changes in pitch at low frequencies much better than at high frequencies. The cochlea of the human inner ear is a fluid-filled, coiled tube that is lined lengthwise with approximately 3,500 sensory cells, similar to a row of tiny hairs, which resonates to different sound frequencies. The cochlea is also separated length-wise by two membranes. One of these, the basilar membrane, is the base of the sensory cells and responds to sound frequencies in a non-linear way along its length. The analogy of this is a continuous array of band-pass filters with exponential frequency response, along the length of the membrane which effectively separates sound into spectral components (Rothmann, 2018). Figure 2.2 illustrates the similarity between the human perception of sound and the non-linear characteristic of the inner ear to the path of a digital signal through a non-linear filterbank and an input to a classifier model.

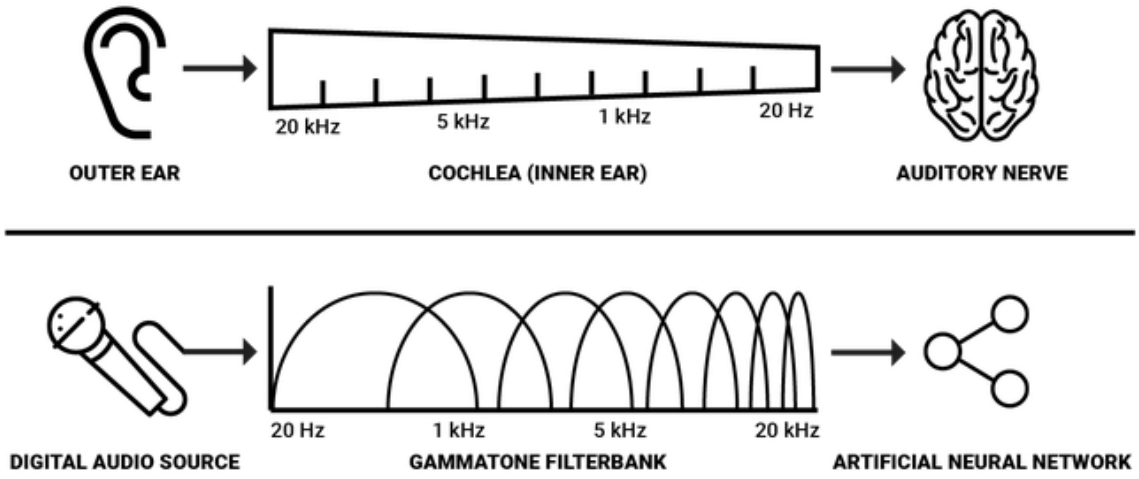

Figure 2.2: A simplified comparison between human hearing and its digital equivalent (Rothmann, 2018)

To simulate the response of the human ear, a signal source cannot just be sampled over a linear frequency scale, rather a non-linear (gamma tone) scale, called the Mel scale is used. The Mel scale relates the perceived frequency, or pitch, of a pure tone to its actual measured frequency. Incorporating this scale makes our features match more closely what humans perceive. The Mel scale emphasises the lower frequency ranges as these contain more useful information.

The formula for converting from frequency to Mel scale is given in (**2.15**)

$$M(f) = 2595\, log_{10}\left(1 + \frac{f}{700}\right) \tag{2.15}$$

Filter banks using the Mel scale are used to compute a particular parameterisation of the cepstrum, known as Mel-Frequency Cepstral Coefficients (MFCCs). A filter bank is an array of overlapping triangular bandpass filters that cover a desired portion of the frequency spectrum, usually from zero up to the Nyquist frequency. On a Mel scale, the distances between the centre frequency f_c and the low frequency (f_l) / high frequency (f_h), i.e. ($f_c - f_l$) and ($f_h - f_c$) are the same as the distance between the centre frequencies of successive filters, mimicking the non-linear response of the human ear, by being more discriminative at lower frequencies and less discriminative at higher

frequencies. The filter bank is applied to map the signal power spectrumobtained from short-time Fourier transform (STFT) to Mel bins, as shown in Figure 2.3.

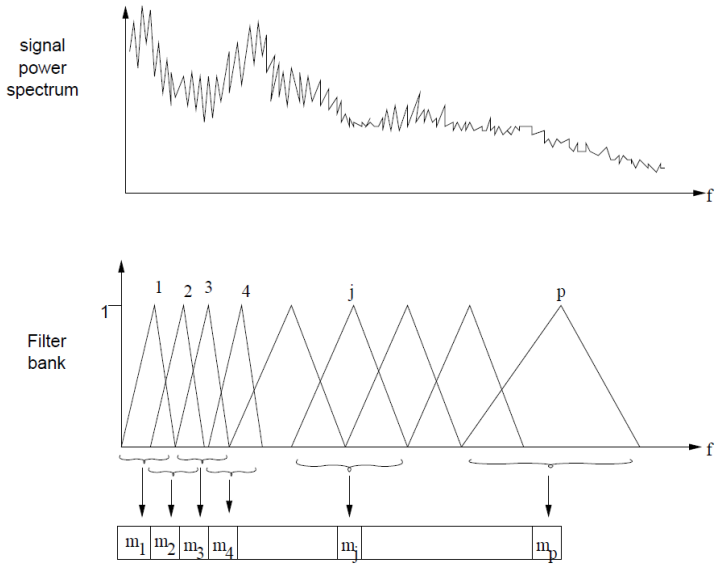

Figure 2.3: Triangular filter bank using the Mel scale (Bishop, 2006)

MFCCs are commonly derived as follows:

- Take the fast Fourier transform (FFT) of each frame of the audio signal.
- Map the powers of the spectrum onto Mel scale bins, by multiplication of the FFT of each frame with each filter in the triangular filter bank. Each bin now contains a weighted sum representing the spectral magnitude in that filter bank channel.
- Take the logarithm of the powers at each of the Mel scale bins.
- Take the Discrete Cosine transform (DCT) of the list of Mel log powers, as if it were a signal.
- The MFCCs are the amplitudes of the resulting spectrum.

The lowest 13 coefficients of the Mel-cepstrum are widely used as a feature in automatic speech recognition. Often the lowest cepstral coefficient is replaced by the frame energy. This feature was evaluated in the research of (Foote 1997.), (Scheirer and Slaney, 1997) and (Ericsson, 2009). The statistical mean and variance of MFCC might not be discriminative enough for distinguishing between the different audio classes and kurtosis and skewness must also be considered (Pikrakis,

Giannakopoulos and Theodoridis, 2006; Gallardo-Antolin and Montero, 2010). MFCC is computationally very expensive, however, it can be justified since 4 Hz Modulation energy is a by-product of MFCC.

2.4.11 4 Hz modulation Energy

Dudley formulated a "modulation theory" of speech, whereby speech sounds are seen as a sum of carrier signals produced by the vocal cords and the vocal tract (Dudley, 1940). The amplitude modulation (AM) component relates to the fluctuations of the temporal envelope of the speech signal. These fluctuations are due to alternating periods of 46-dimensional silence and higher energy events, corresponding to the 4 Hz syllabic rate. The frequency modulation (FM) components are mainly due to the primary glottal frequency and the harmonics thereof, the amplitude and frequency of which change slowly as a consequence of the dynamic changes of the vocal tract during phonation (Varnet et al., 2017). Speech signals thus have a distinct rhythmic rate of energy pulsation due to amplitude modulation, corresponding to the average syllable rate (around 4 Hz, depending on speaker and language spoken) and the rate of music energy spectrum variation is determined by the beat rate, around 0.7 Hz – 2 Hz. (Scheirer and Slaney, 1997; Edwards and Chang, 2013).

This feature can be calculated using different methods. The first is a by-product of the MFCC feature. Unless the MFCC feature is also required, this method is unnecessarily complex and computationally expensive. The alternative method is using the Hilbert transform for envelope detection. In a simpler experiment, to avoid the MFCC calculation process, (Pinquier, Rouas and E-Obrecht, 2002), determined the 4 Hz Modulation energy directly from the short term energy of the signal.

In (Scheirer and Slaney, 1997) and (Pinquier, Rouas and E-Obrecht, 2002), the authors calculated the energy for each MFCC bin. This energy was then filtered with an FIR bandpass filter, centred on 4 Hz. The energy is summed for all channels and normalized by the mean energy on the frame.

This method first produces an analytical signal of the original signal by either extracting the signal envelope by using half-wave rectification that is then filtered with a low pass filter with a cut-off at around 50 Hz or the Hilbert Transform. An analytic signal (**2.16**) is a form of envelope detection

where the magnitude and phase information is extracted from a signal. The next step is to apply a Fast Fourier transform to the analytical signal waveform. It is important that the frequency resolution is low enough (~1 Hz). The analytical signal $s_A(t)$ of signal (t) , where $j\hat{s}(t)$ is the Hilbert Transform of the signal $s(t)$:

$$s_A(t) = s(t) + j\hat{s}(t) \tag{2.16}$$

The Analytic Signal is complex-valued, therefore it can be expressed in exponential notation (**2.17**) where $A(t)$ is the instantaneous amplitude (envelope) and $\varphi(t)$ is the instantaneous phase:

$$s_A(t) = A(t)e^{j\varphi(t)} \tag{2.17}$$

A note on discrete signal energy vs power in time and frequency domain:

An energy signal is a signal with finite energy and zero average power, while a power signal is a signal with infinite energy but finite average power. Energy signals are time-limited while power signals can exist over infinite time.

Non-periodic signals are energy signals while power signals are periodic.

The power of an energy signal tends to zero and the energy of a power signal is approaching infinite.

A discrete-time energy signal is defined as one for which $0 < E < \infty$ and a discrete-time power signal is defined as $0 < P < \infty$. A discrete-time signal can be neither an energy signal nor a power signal, but it cannot be both. Since audio signals are mostly non-periodic, they can be regarded as energy signals. The energy in the time domain for a discrete-time signal is shown in (**2.18**). However, the glottal excitation frequency portion of speech may be regarded as periodic and thus regarded as a discrete-time power signal. (**2.19**)

$$E = \sum_{n=0}^{N-1} |x(n)|^2 \tag{2.18}$$

$$P = \lim_{N\to\infty} \left(\frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2 \right) = \frac{1}{N} \int_{k=0}^{N-1} |X[k]|^2 \tag{2.19}$$

The energy in the frequency domain for the same discrete-time signal is equal to the DFT of the time domain discrete-time signal. This is Parseval's relation, which implies, inter alia, that the total energy may be determined either summing over all time-domain samples or by integrating within the frequency domain over all frequencies.

2.5 Summary of previous research on audio features

The paper by Scheirer and Slaney (1997) used thirteen audio features, to train different types of multidimensional classifiers, including a Gaussian MAP estimator and the nearest neighbour classifier and was considered the most extensive and influential study to date and is widely cited (n > 1293). This research aims to build on this and complete an even more comprehensive study of the most common features used in audio content analysis. Table 2.3 contains a summary of prior research on audio content analysis where the audio features used were explicitly mentioned. The references are listed alphabetically and the table summarises the purpose and main applications of the research as well as the classification method that was used. It is interesting to note that while there is some similarity in audio features used, only a few studies mention the use of higher-order statistical moments for analysis. Skewness and kurtosis is only mentioned by Bugatti, Leonardi and Rossi (1999), Lavner and Ruinskiy (2009), Muñoz-Expósito et al. (2009), Gallardo-Antolin and Montero (2010) and Sigtia et al., (2016).

Table 2.3: Abbreviated summary of previous research on speech / music discrimination

Reference	Main Applications	Features	Classification method
(Ajmera, McCowan and Bourlard, 2003)	Automatic transcription of broadcast news	Entropy measure and "dynamism" estimated at the output of a multilayer perceptron (MLP) trained to emit posterior probabilities of phones. MLP input: 13 first cepstra of a 12th-order perceptual linear prediction filter.	2-state HMM with minimum duration constraints (threshold-free, unsupervised, no training).
(Alexandre et al., 2006)	Speech/music classification algorithm for hearing aids.	Spectral centroid, spectral rolloff, spectral flux, ZCR, High Zero Crossing Rate Ratio, short-time energy, low short time energy ratio (LSTER), MFCC, voice-to- white, activity level, loudness, Spectral Flatness Measure (SFM)	Fisher linear discriminant, three-layer MLP neural network
(Alexandre *et al.*, 2008)	Development of an automatic sound classifier for digital hearing aids with automatic self-adaptation.	14 Features: Spectral Centroid, Spectral Roll-Off, Voice2White, Spectral Flux, Zero Crossing Rate, Short-Time Energy, Percentage of Low Energy Frames, High Zero Crossing Rate Ratio, Low Short-Time Energy Rate, Spectral Flatness Measure, Mel Frequency Cepstral Coefficients, Loudness, Spectral Crest Factor, Bandwidth	Two-stage Mean Squared Error (MSE) linear discriminant classifier, first one discriminates the input sound into either speech or non-speech, the second layer classifies it into either speech in quiet or speech in noise.
(Bachu *et al.*, 2010)	Separation of voiced and unvoiced speech	ZCR and Energy	Not stated
(Balabko, 1999)	Speech/music discrimination for automatic speech recognition systems	MFCC. Used 6th band (466-600 Hz) and 20th band (1510-1732 Hz) out of 40 bands	Gaussian Classifier
(Barbedo and Lopes, 2006)	Automatic genre classification of musical signals.	Bandwidth, spectral roll-off, spectral flux, and loudness. Three summary features are extracted: mean, variance, and main peak prevalence	Pairwise Euclidean distance four-layer hierarchical structure search tree and linear discriminant analysis (LDA)
(Beierholm and Baggenstoss, 2004)	Class-specific features discriminating between speech and music.	Auto-correlation function (ACF)	Class-specific density functions were estimated using Gaussian mixture HMMs. 130 speech samples and 75 music samples were used in the density estimation
(Bugatti, Flammini and Migliorati, 2002)	"Table of Content description" of a multimedia document	ZCR-based features, spectral flux, short-time energy, cepstrum coefficients, spectral centroids, the ratio of the high-frequency power spectrum, a measure based on the syllabic frequency	Multivariate Gaussian classifier, neural network (MLP)

(Bugatti, Leonardi and Rossi, 1999)	Video indexing approach based only on audio classification.	ZCR variance, skewness (3rd moment), Difference between the number of ZCR samples which are above and below the mean and Short-time energy	Multivariate Gaussian classifier
(Burred and Lerch, 2004)	Audio classification (speech/ music/background noise), music classification into genres	Statistical measures of the short-time frame features: ZCR, spectral centroid/roll-off/flux, first 5 MFCCs, audio spectrum centroid/flatness, harmonic ratio, beat strength, rhythmic regularity, RMS energy, time envelope, low energy rate, loudness, others	KNN classifier, 3- component GMM classifier
(Carey, Parris and Lloyd-Thomas, 1999)	Proof of concept for pitch and amplitude features	Mel Frequency Cepstral Coefficients, Delta Cepstral Coefficients, Amplitude, Delta Amplitude, Pitch, Delta Pitch, Zero-Crossing Rate and Delta Zero-Crossing Rate	Gaussian Mixture Model using the difference in the log-likelihood
(Wu Chou and Liang Gu, 2001)	Signing signal detection in speech/music discrimination applied to applications of audio indexing.	4 Hz modulation energy, harmonic coefficient and 4 Hz modulation value of the harmonic coefficient	Gaussian Mixture Model
(El-Maleh *et al.*, 2000)	Automatic coding and content-based audio/video retrieval	Line spectral frequencies (LSF), differential LSF, measures based on the ZCR of high-pass filtered signal and linear prediction zero-crossing ratio (LP-ZCR)	KNN classifier and quadratic Gaussian classifier (QCG)
(Ericsson 2009)	Thesis – Discrimination and annotation of speech and music for radio broadcasts.	Variance, standard deviation and the derivative and the standard deviation of the RMS amplitude, Zero Crossing-Rate (ZCR), Mel Frequency Cepstrum Coefficients (MFCC), Spectral Centroid (SC), Pulse Clarity (PC) and Modified Low Energy Ratio (MLER)	Compare feature histograms per class. Final classification algorithm not stated.
(Foote, 1997)	Audio search engine. Retrieving audio documents by acoustic similarity	12 MFCC, Short-time energy	Template matching of histograms, created using a tree-based, vector quantizer, trained to maximize mutual information. Create histogram templates of classes to compare unknown samples to base on Euclidian and Cosine distance measures.
(Gallardo-Antolin and Montero, 2010)	Automatically classifying collections of audio files in three acoustic classes: speech, instrumental music and song (music with singing voice).	Mean, variance, kurtosis and skewness of 12 MFCC. Polynomial-Fit Histogram Equalization (PHEQ) of the MFCC	GMM-based classifier
(Giannakopoulos, Pikrakis and Theodoridis, 2006)	Speech/music discriminator for radio recordings.	The first and second moment of Spectral Centroid, Spectral Flux, Spectral Rolloff, Zero Crossing Rate, Frame Energy and four Mel-frequency cepstral coefficients (MFCCs)	Bayesian Network (BN) that combines the outputs of nine individual k-Nearest Neighbour classifiers.

Giannakopoulos (2015)	Python Audio Analysis library is to provide a wide range of audio analysis functionalities	Zero-Crossing Rate, Energy, Entropy of Normalised Energy, Spectral Centroid, Spectral Spread (The second central moment of the spectrum.), Spectral Entropy, Spectral Flux, Spectral Rolloff, 12 MFCCs, Chroma Vector (12-element) and Chroma Deviation	Support vector machines and the k-Nearest Neighbour classifier
(Graves, Mohamed & Hinton, 2013)	Speech recognition.	Map directly from acoustic to phonetic sequences. No feature extraction required.	Recurrent neural networks (RNNs). All networks were trained using stochastic gradient descent, with learning rate 10−4, momentum 0.9 and random initial weights are drawn uniformly from [−0.1, 0.1].
(Khan and Al-Khatib, 2006)	Evaluation of features and classifiers for speech and music classification.	Haar Discrete wavelet transform, the variance of MFCC, RMS of lowpass filtered signal, delta ZCR. Used fuzzy C-means clustering to select viable feature.	Multi-Layer Perceptron (MLP) Neural Networks, radial basis functions (RBF) Neural Networks, and Hidden Markov Model (HMM)
(Lavner and Ruinskiy, 2009)	Consumer audio application of segmentation of audio signals into speech or music	Mean and Standard deviation of Short-time energy, Zero Crossing rate, Band Energy Ratio, Autocorrelation Coefficient, MFCC, Spectral Rolloff, Spectral Centroid. Spectral Flux and Spectrum Spread. The skewness of ZCR and the difference magnitude between consecutive analysis frames. Low Short Time Energy	Three-stage sieve-like decision tree approach, applying both Bayesian and rule-based methods.
(Lobo and Loizou, 2003)	Algorithm for voiced-unvoiced speech discrimination in noise.	Gabor coefficient atomic decomposition	Radial Basis function MLP neural network
(Liu, Wang and Chen, 1998)	Analysis of audio for the scene classification of TV programs	The mean and standard deviation of RMS value, Silence ratio, volume std, volume dynamic range, mean and std of pitch difference, speech, noise ratios, frequency centroid, bandwidth, energy in 4 sub-bands	A feed-forward neural network using the one-class-in-one-network (OCON) structure
(Lu, Jiang and Zhang, 2001)	Audio content analysis in video parsing	High zero-crossing rate ratio (HZCRR), low short-time energy ratio (LSTER), linear spectral pairs, band periodicity, noise-frame ratio (NFR)	3-step classification: 1. KNN and linear spectral pairs-vector quantization (LSP-VQ) for speech/non- speech discrimination. 2. Heuristic rules for non-speech classification into music/background noise/silence. 3. Speaker segmentation

(Lu, Zhang and Li, 2003)	Content-based audio classification and segmentation using support vector machines (SVM)	8 MFCC coefficients, zero-crossing rate (ZCR), short-time energy (STE), sub-band power distribution, brightness, bandwidth, spectrum flux (SF), band periodicity (BP), and noise frame ratio (NFR).	Compared the performance of SVM, K-Nearest Neighbour (KNN), and Gaussian Mixture Model (GMM). Audio is classified into five classes. They are silence, music, background sound, pure speech, and non-pure speech which include speech with music and speech with noise. Used rule-based pre- and post-classification
(Markaki and Stylianou, 2011)	Discrimination of speech and non-speech for speaker segmentation/recognition and speech transcription	MFCC and Modulation spectrum envelope for 65 frequency sub-bands was detected by a magnitude square operator. Higher-order generalization of singular value decomposition (HOSVD) for dimensionality reduction.	SVM classifier with median filter smoothing
(Muñoz-Expósito *et al.*, 2009)	An intelligent audio coding system	Warped LPC-based spectral centroid. Used the mean, variance, and skewness of the feature vector.	3-component GMM, with or without fuzzy rules-based system. SVM with a fuzzy rule-based system
(Panagiotakis and Tziritas, 2005)	Web-based streaming audio content characterisation	Mean and variance of RMS value and ZCR as an indication of mean frequency	Rule-based Gaussian likelihood ratio test.
(Pikrakis, Giannakopoulos and Theodoridis, 2006)	Discrimination scheme for radio recordings	Energy, zero-crossing rate, spectral entropy and the first two Mel- Frequency Cepstral Coefficients (MFCCs). ML	Variable Duration Hidden Markov Model (VDHMM) and a Bayesian Network (BN).
(Pinquier, Rouas and E-Obrecht, 2002).	To describe and index an audio document. Keywords and speakers detection.	4 Hz modulation energy, entropy modulation, number of “stationary” segments and segment duration.	Bayesian maximum likelihood classifier
(Saad *et al.*, 2002)	Automatic classification of audio signals	Percentage of low energy frames, spectral roll-off, spectral centroid, spectral flux, ZCR	Rule-based classification
(Saunders, 1996)	Automatic monitoring of FM radio channels in real-time	Short-time energy, statistical parameters of the ZCR	Multivariate Gaussian classifier

(Scheirer and Slaney, 1997)	Speech/music discrimination for automatic speech recognition	13 temporal, spectral and cepstral features (e.g., 4 Hz modulation energy, % of low energy frames, spectral roll-off, spectral centroid, spectral flux, ZCR, cepstrum-based feature, "rhythmicness"), the variance of features across 1 sec. Pulse Metric	Gaussian mixture model (GMM), K nearest neighbour (KNN), K-D trees, multidimensional Gaussian MAP estimator
(Sigtia *et al.*, 2016)	Automatic Environmental Sound Recognition (AESR) algorithm for low power IoT devices.	MFCCs, Spectral centroid, Spectral flatness, Spectral roll-off, Spectral kurtosis and Zero crossing rate	Compared Gaussian mixture models (GMMs), support vector machines (SVMs) and deep neural networks (DNNs) in terms of their performance and their computational cost.
(Taniguchi, Tohyama & Shirai, 2008)	Speech, music and mixed sound classification method based on sinusoidal trajectories.	Sinusoidal trajectories and temporal features extracted thereof.	Gaussian mixture models
(Tardón, Sammartino & Barbancho, 2010)	Efficient music-speech discriminator for large audio data sets	The mean and standard deviation of RMS, ZCR, Cepstrum Residuals, Spectral Flux, DFT Magnitude, 5 MFCC's, Volume dynamic ratio were used in the final classification. Silence ratio, Spectral Centroid, Spectral Rolloff, Bandwidth, frame and segment energy, fundamental frequency and Salience of pitch was also evaluated.	Simple Gaussian model
(Tzanetakis and Cook, 2002)	Genre categorisation for audio	Mean and Std Deviation of Spectral Centroid, Spectral-Rolloff, Spectral Flux, Zero Crossing ratio as well as Percentage Low Energy frames, MFCC and rhythm features derived from Discrete Wavelet Transform	Gaussian classifier
(Wang, Gao and Ying, 2003)	An experimental approach to discriminate speech and music	Modified Low Energy ratio	Bayes MAP classifier. Rule-based post-decision filter
(Williams and Ellis, 1999)	Segmentation of speech versus non-speech in automatic speech recognition tasks	Mean per-frame entropy and average probability "dynamism", background-label energy ratio, phone distribution match—all derived from posterior probabilities of phones in hybrid connectionist-HMM framework	Gaussian likelihood ratio test

2.6 Decision and Learning Paradigms

The application of machine learning problems can be approached in two different ways: Frequentism and Bayesianism. Fundamentally, the disagreement between frequentists and Bayesians concerns the definition of probability. For frequentists, probabilities are fundamentally related to the frequencies of events. While for Bayesians, probabilities are fundamentally related to our knowledge about an event.

2.6.1 Frequentist Learning

Frequentist learning relies on a frequency of events to derive a conclusion, all culminating in one universal true model with parameters θ_0. In frequentist learning, these parameters are selected through a process of optimisation such as the likelihood function.

Maximum likelihood is the most widely used estimator in Frequentist learning. In the Frequentist learning paradigm, we try to avoid prior beliefs, bias and subjectivity, instead we believe that there exists only one true model, with parameters θ_0. The belief in this model is enforced by statistical data. The estimated parameters based on the training data, that maximise the likelihood, are denoted $\hat{\theta}$, where $\hat{\theta} = argmax_{\theta} p(X|\theta)$. For independent and identically distributed data from $p(x|\theta_0)$, the maximum likelihood distribution minimises the Kullback-Liebler divergence (**2.19**). The first term of the Kullback-Leibler divergence represents the information (inverse entropy) in the true model and the second term is the information in the chosen model.

$$\hat{\theta} = argmin_{\theta} \int log \frac{p(x|\theta_0)}{p(x|\theta)} p(x|\theta_0) \tag{2.19a}$$

$$\hat{\theta} = argmin_{\theta} \int p(x|\theta_0) log p(x|\theta_0) dx - \int p(x|\theta_0) log p(x|\theta) dx \tag{2.19b}$$

According to the law of large numbers, the estimated parameters will diverge towards the true model if the data set is sufficiently large, $\hat{\theta}n \xrightarrow{\Rightarrow} \theta_0$. The maximum likelihood estimate (MLE) of θ is obtained by taking the derivative of the log-likelihood, $log_p p(y|X, \theta, \sigma)$ and adjusting the parameters (θ, σ) in order to maximise the likelihood.

2.6.2 Bayesian Learning

Bayesian learning relies on the paradigm that there exists some prior observed knowledge component that is expressed as a relevant probability distribution and we apply the Bayes rule to compute a corresponding posterior distribution, given the prior data. In Bayesian learning $\theta$is a random variable wherein ML, θ is a known parameter and the randomness is in the data. Where maximum likelihood is regarded as a problem of optimisation, Bayesian learning is a problem of integration. Conjugate analysis plays an important role in solving posterior probabilities in Bayesian learning. By using the notion of conjugate priors we can establish the functional form of the posterior distribution to be the same as that of the prior distribution, thereby simplifying the Bayesian analysis. For example, the prior mean of a Gaussian is another Gaussian distribution and using the mathematical property that the area under a Gaussian integrates to one, reduces the complexity of the calculation. The conditional probability in the case of Frequentist learning is defined as an expected frequency of occurrence over a large number of experiments (**2.20**).

$$P(X|Y) = p(y|X, \theta_0) \tag{2.20}$$

While in Bayesian learning the conditional probability is related to a degree of belief and is defined as (**2.21**)

$$P(X|Y) = \int p(y|X, \theta, \sigma)\, p(\theta|D) \tag{2.21a}$$

where

$$p(y|X, \theta, \sigma) = (2\pi\sigma)^{\frac{-1}{2}} . e^{\frac{-1}{2\sigma^2}(y-X\theta)^T(y-X\theta)} \tag{2.21b}$$

Remember that the sum of probabilities in a Gaussian distribution sums to one:

$$\int p(y|X,\theta,\sigma) = 1 \tag{2.22}$$

then:

$$\int p(y|X,\theta,\sigma) = \int (2\pi\sigma)^{\frac{-1}{2}} . e^{\frac{-1}{2\sigma^2}(y-X\theta)^T(y-X\theta)} = 1 \tag{2.23a}$$

where

$$\int e^{\frac{-1}{2\sigma^2}(y-X\theta)^T(y-X\theta)} = (2\pi\sigma)^{\frac{1}{2}} \tag{2.23b}$$

Assume a class model C_k with prior distribution $p(C_k)$ and an observed data set, x, we can use Bayes Theorem to find the posterior probabilities $P(x|C_k)$ corresponding to a value of x. (**2.24**)

$$P(x|C_k) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{2.24}$$

Bayes rule relates what we know about a model before (prior) and after (posterior) having seen the data. The difference between the prior and posterior distributions for the model can be interpreted as a `machine learning' effect. The aim of using the Bayes Rule is to maximise the class conditional probability $P(x|C_k)$ to find $argmax\ \{P(x|C_k)P(C_k)\}$. The probability that a feature vector x belongs to the class C_k is $P(C_k|x)$, and it is often referred to as a posteriori probability. The classification of the vector is done according to posterior probabilities or decision risks calculated from the probabilities using Bayes Rule.

In the Bayesian paradigm, we place higher confidence in the data that is closely related to known data but it relies on exploring new data in areas of high uncertainty, as this either affirms the

confidence of existing data or provides the opportunity to explore new lucrative data. Both scenarios improve the quality of our predictions (exploration versus exploitation).

2.7 Classification Models

In this study, an MLP neural network was built and compared with the Python Scikit-Learn library (Pedregosa, Weiss and Brucher, 2011) implementation of the SVM classifier. The MLP neural network was written from scratch in Python and the library implementation of the SVM classifier served as a benchmark to compare the performance of the neural network and to help with the selection of the optimal feature set. The multilayer perceptron was previously used by Ajmera, McCowan and Bourlard (2003) to estimate the posterior probabilities of the speech phonetic classes, as a precursor to an HMM classifier. Alexandre *et al.* (2006) compared a Fisher linear discriminant classification algorithm and a neural network. In discriminating speech from non-speech data, the MLP obtained better results while discriminating between pure speech and noisy speech, the results were almost equal. Bugatti et al. compared an MLP classifier with 8 input vectors, including variants of Spectral Flux, Short Time Energy, Spectral Centroid and MFCC features to a simpler configuration of Zero Crossing Ratio and a Bayesian classifier (Bugatti, Flammini and Migliorati, 2002). The latter produced comparable results in the case of pure music or speech but performed poorly in mixed class and noisy environments compared to the MLP. The authors concluded that the higher computational complexity of the MLP was a worthwhile trade-off for superior classification performance. Khan and Al-Khatib (2006) compared three classifiers; MLP and Radial Basis Function Neural Network (RBFNN), and Hidden Markov Model (HMM) applied to various sets of features. The RBFNN delivered a sporadic performance in their application while both, MLP networks and HMMs have given good results. The authors pointed out that a disadvantage of using HMMs is that it requires long training and testing time as compared to MLP and that HMMs need to be trained for each audio class separately, which requires more memory space. MLP is trained only once for all the audio classes simultaneously and the synaptic weights are stored once.

In a fairly recent and salient paper, Sigtia *et al.*, (2016) compared Gaussian mixture models (GMMs), support vector machines (SVMs) and Feed-Forward Deep Neural Networks (DNNs) and Recurrent

Neural Networks (RNNs) in terms of their performance as a function of their computational cost. The features used were MFCCs, Spectral Centroid, Spectral Flatness, Spectral Rolloff and Zero Crossing Rate. They suggest that GMMs have a low computational cost and reasonable performance. SVMs with linear and sigmoid kernels yield similar performance compared to GMMs, but their computational cost is overall higher. The neural networks consistently outperform both the GMMs and the SVMs, although they are computationally most expensive. The authors allude to the fact that the computational cost of DNNs can be controlled by limiting the number of hidden units and the number of layers.

In Lu, Zhang and Li, (2003) the performance of SVM, K-Nearest Neighbor (KNN), and Gaussian Mixture Model (GMM) was compared. The authors claimed that the performance of the SVM method is much higher than that of using KNN and GMM classifiers. Markaki and Stylianou (Markaki and Stylianou, 2011) compared frame-based and segment-based SVM classifiers and found that using a median filter for the output smoothing on the frame-based SVM classification produced the best results. The authors (Muñoz-Expósito *et al.*, 2009) compared the performance of an SVM-based classifier for speech/music discrimination to the GMM classifier with a fuzzy rules post-decision stage that achieved an improvement of about 6% compared to using only the SVM-based classifier. Both performed better than the GMM classifier.

2.8 Summary

This chapter provides a detailed literature study of research on audio content analysis and audio feature extraction used in previous studies, specifically to discriminate between different audio classes. The prelude of the chapter explained the rationale and aim of this study. In section 2.2, the properties of the data classes are examined and this is followed by a brief introduction to audio content analysis. Section 2.4 is a discussion of the various audio features utilised in this study the process of pattern recognition to develop an algorithm that can effectively and accurately discriminate between these classes of audio. Section 2.5 provides a summary of previous research on audio features. Section 2.6 is a discussion of the different paradigms followed in machine learning decision making, explaining the differences in the classifiers based on a popular method used in each approach. For the frequentist machine learning paradigm, the maximum likelihood estimator

is discussed in comparison to Bayesian learning and classifiers based on the Bayesian rule approach. This sets the background for the discussion of various classification models used in previous studies. It is interesting to note that there were many different classifier models and learning approaches used in previous studies and that no clear advantage or general preference in the choice of classifier method was observed across the different previous studies in this literature review. Instead, there appears to be a consensus that the choice of classifier model is a trade-off between higher computational complexity versus computational speed and simplicity at the possible sacrifice of superior classification performance. In the next chapter, the development of a Neural Network-based classifier for discriminating between speech and music audio classes is described. Classical neural networks use maximum likelihood estimation to optimise network weight and bias parameters, thus making it an example of frequentist learning. This model is compared to an SVM model to benchmark classifier performance as well as for the selection of an optimal audio feature set.

Chapter 3 - Classification Model for Audio Content Analysis

3.1 Introduction

The study of speech and language processing requires preparing homogeneous data sets based on real-world data such as digital television broadcasts and internet podcasts. The same requirement exists for audio classification, audio segmentation, and music information retrieval tasks. The preparation of datasets for this research, especially the manual extraction of speech or music data from mixed media recordings, is a laborious and time-consuming exercise, slowing down the progress of the research. To automate this process, audio content analysis (ACA) is applied. Lavner and Ruinskiy describe audio content analysis as a field of research within machine learning, also called computer audition or machine listening (Lavner and Ruinskiy, 2009). This process entails the extraction of information from audio signals through statistical modelling of data and the application of probability and decision theory with the aim of classification of sound categories. It combines knowledge and methods from a variety of disciplines including signal processing, machine learning, sound perception (psychoacoustics) and cognitive musicology (Scheirer, 2000), (Ericsson, 2009), (Tzanetakis and Cook, 2002). Humans orientate themselves quickly with musical sounds and effortlessly distinguish between speech and music and even a combination of sounds. For a machine learning process, however, it is a very different scenario, as music audio and environmental noise obfuscate speech data and vice versa (Scheirer, 2000).

Audio content analysis and discrimination algorithms generally consist of two stages. The first stage is a supervised learning phase, based on a statistical approach. In this phase labelled training data for each class that needs to be identified is analysed by the algorithm to build a statistical model of each of the classes based on the extracted feature set and to define the separation boundaries (thresholds) between the different classes. In the second stage, the processing phase, the test audio signal is analysed and the feature data for each segment is compared against the statistical model of each class and labelled by the classification algorithm according to the highest probability class affinity.

The choice of audio features must reflect the significant characteristics of each class of audio signals and it influences the accuracy of audio classification greatly, consequently, it is important to further reduce the feature space by ranking the importance of these features, both individually and in combinations to compare how they contribute to the overall process. The next step of the process is to evaluate and benchmark the classification algorithm presented here, implemented in a high-level programming language in this case, Python, with another established and mature open-source classification algorithm and compare the difference in performance.

3.2 Feature extraction

In this study, several of the different audio features suggested in previous studies were compared, comprising both time-domain (temporal) and frequency domain (spectral) features and focused on separating pure speech and music classes as well as removing silent segments in a pre-processing step. Unlike previous studies, the features are first evaluated and ranked according to individual and collective suitability to discriminate between the two audio classes. Various feature selection methods were used to evaluate both individual and combinations of features, culminating in the selection of a final set of features to be used in the classification task. The final feature set was compared on two different classification algorithms, a feed-forward multi-layer perceptron (MLP) neural network classifier and a support vector machine (SVM) classifier as a baseline.

3.3 Support Vector Machine

The SVM classifier is a supervised machine learning classification algorithm, first introduced in the 1960s but only became popular in the 1990s. The reason was that it was initially only regarded as a linear, binary classifier, and yes, SVM is excellently suited to linearly separable data in two classes. Support Vector Machines are based on the concept of decision planes that define decision boundaries. A simplified representation of the elements of an SVM is shown in Figure 3.1, indicating the relation between the data points, support vectors and the separation plane. The two data classes are separated by the optimal hyperplane. Although it is possible to define multiple decision boundaries that will possibly separate data classes, the SVM algorithm attempts to find the optimal decision boundary between two data classes, by establishing the decision boundary that maximizes the distance from the nearest data points of all the classes. The nearest data points of each class to

the decision boundary, that maximise the distance between the decision boundary and these points are called support vectors. Support vectors are the most difficult to classify and have a direct bearing on the optimum location of the decision boundaries.

The decision boundary is referred to as the maximum margin classifier or the maximum margin hyperplane if the data is in higher dimensions. The SVM classifier also scales well to higher dimensional and non-linearly separable data when using a kernel method.

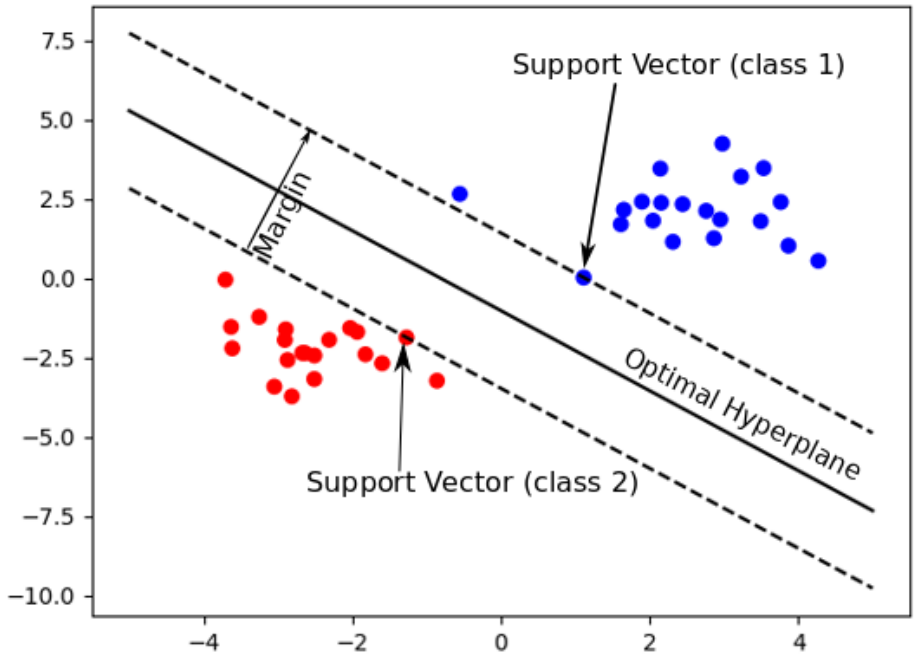

Figure 3.1: SVM margin and hyper-plane

Kernel models first transform the input feature space to achieve non-linearity by increasing the dimensionality. In the case of the Gaussian RBF kernel, this equates to virtually infinite-dimensional space. The biggest disadvantage of the SVM classifier is that kernelised SVMs require the computation of a distance function between each point in the dataset, the computational scaling with the number of samples N is $O[N_{features} * N_{samples}^3]$ or at best $O[N_{features} * N_{samples}^2]$ for

efficient implementations. For large numbers of training samples, this computational cost can be inefficient or prohibitive. Since the calculation of support vectors and associated hyper-plane is not derived from a probability model, SVM is regarded as a frequentist learning perspective.

3.4 Neural Network Model construction

Classification performance is defined by two characteristics, accuracy and the processing effort required to achieve it. The objective is to ensure that the selected features deliver optimal classification performance. While it should always be the goal to maximise the classification accuracy, a large feature set may be undesirable as it implies a processing performance penalty. In some applications such as the Internet of Things (IoT), processing power is a restriction, and a slight reduction in accuracy might be a viable sacrifice to reduce the size of the feature set.

3.4.1 Perceptron

Neural networks as the name suggest, mimic the synaptic function of animal brain neurons. McCulloch and Pitts (1943) formulated a mathematical model of neurons as a set of weighted inputs (w_i) that corresponds to synapses, an adder function that sums these inputs and an activation (or threshold) function that decides the output state, i.e. the neuron fires or not (**3.1**). This mathematical model of a single neuron is represented as a block diagram in Figure 3.2.

The mathematical model of neurons, where $x_i = x_1, x_2, \dots, x_m$ as a set of input nodes, $w_i = w_1, w_2, \dots, w_m$ as the corresponding weights and h is the output of the adder function, can be represented as follows:

$$h = \sum_{i=1}^{m} w_i \, x_i \tag{3.1}$$

The McCulloch and Pitts (1943) model also provide for an additional bias input to provide an outcome for the situation where all the inputs are simultaneously zero, in which case the weighted inputs are all negated and causing the output to depend purely on the activation function threshold.

The bias input is usually set to the value -1. The McCulloch and Pitts neuron model is thus a binary threshold device and a network of such neurons is capable of complex decision making, provided the input weighting is correctly set. A network of neurons is also called a perceptron.

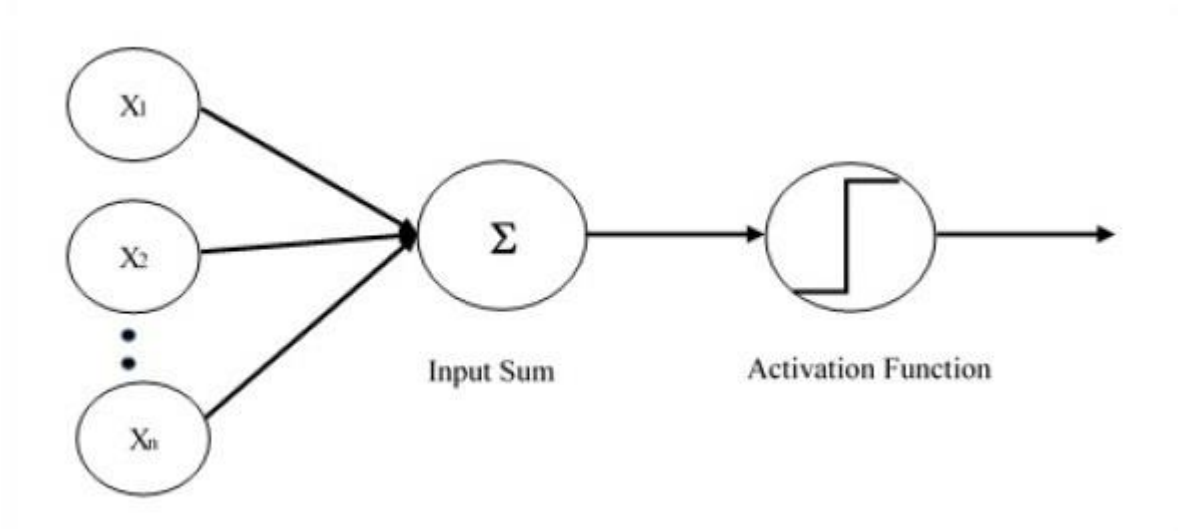

Figure 3.2: Single layer perceptron

In supervised learning, the weighting of the inputs and threshold values must be tuned such that the output of the perceptron must closely resemble the output target of training data. This is an iterative process and during each iteration, the weight of each neuron is adjusted to minimise the error function and the magnitude of the adjustment is controlled by the learning rate parameter (η). The value of η determines how fast the network learns while still providing dampening to prevent instability.

The single-layer perceptron is a linear classifier and is therefore limited to solutions where groups of data can be separated by a straight line or hyper-plane. The most famous example of the perceptron's inability to solve problems with non-linear vectors is the Boolean XOR problem. Multilayer neural networks are created by inserting additional weight and adder layers to the perceptron and this enables us to solve non-linear separation problems. These additional layers are also called hidden layers because it is not possible to examine or adjust their values directly. This

multilayer structure is shown in Figure 3.3, illustrating the interconection of perceptrons with the inclusion of hidden layers.

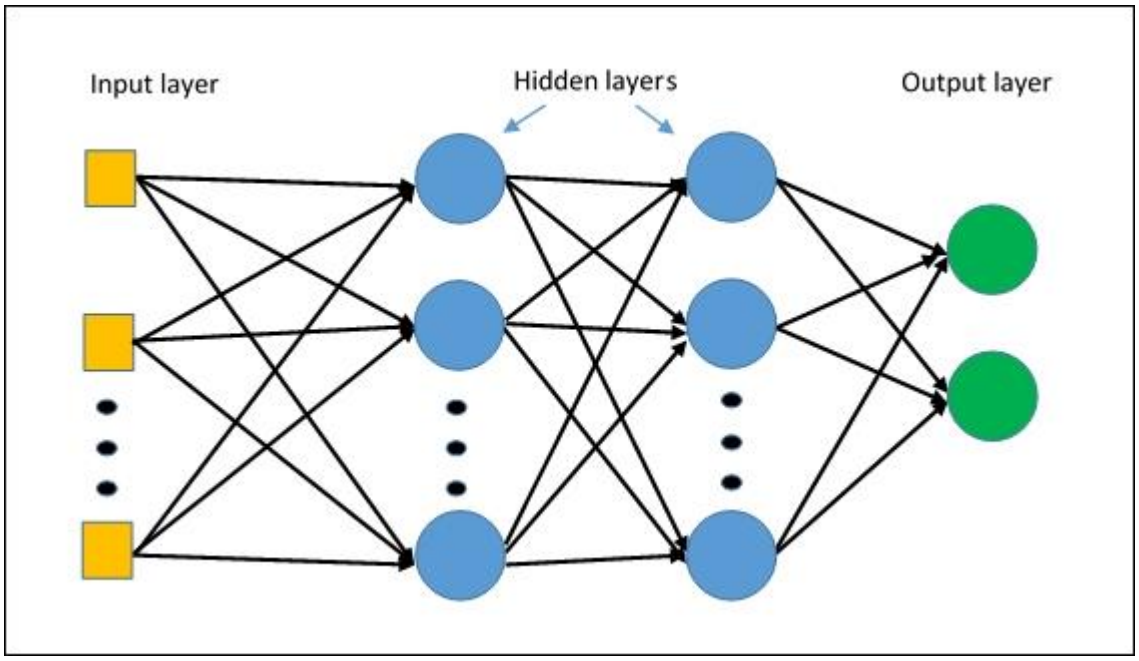

Figure 3.3: Structure of a Multi-Layer Perceptron

For the perceptron the error function: $E = t - y$ is used, however with the MLP you need to use a sum-of-squares error function to avoid making the mistake of inner layer errors cancelling out due to mismatched signs (**3.2**).

$$E(t, y) = \frac{1}{2}\sum_{k=1}^{n}(t_k - y_k)^2 \tag{3.2}$$

In an MLP the weight update function is not that simple since you cannot directly determine which layer is responsible for which portion of the total error. Instead, an optimisation algorithm is used to minimise the error function.

3.4.2 Activation Function

There are a few choices for the activation function of an MLP network. In this study, the sigmoid activation function was used. The sigmoid function resembles a binary logistic regression classifier solution (**3.3**). The advantage of using the sigmoid curve is that it allows for differentiation, which greatly simplifies the back-propagation training of multi-layer networks (**3.4**).

$$P(class = 1) = \sigma = \frac{1}{1 + e^{-z}} \tag{3.3}$$

The derivative of the sigmoid function is:

$$\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big) \tag{3.4}$$

This function has saturation points at (0;1) with a linear transition between these values at the decision boundary as shown in Figure 3.4. The derivative of the sigmoid function is also shown in this figure. The smooth and continuous output between (0;1) can be interpreted as a probability range, with the decision boundary being:

$$P \geq 0.5 \; class{=}1$$

$$P{<}0.5, \; class{=}0$$

Since this is a binary classification problem, applying a moving average filter can improve the classification result accuracy on flip point decisions.

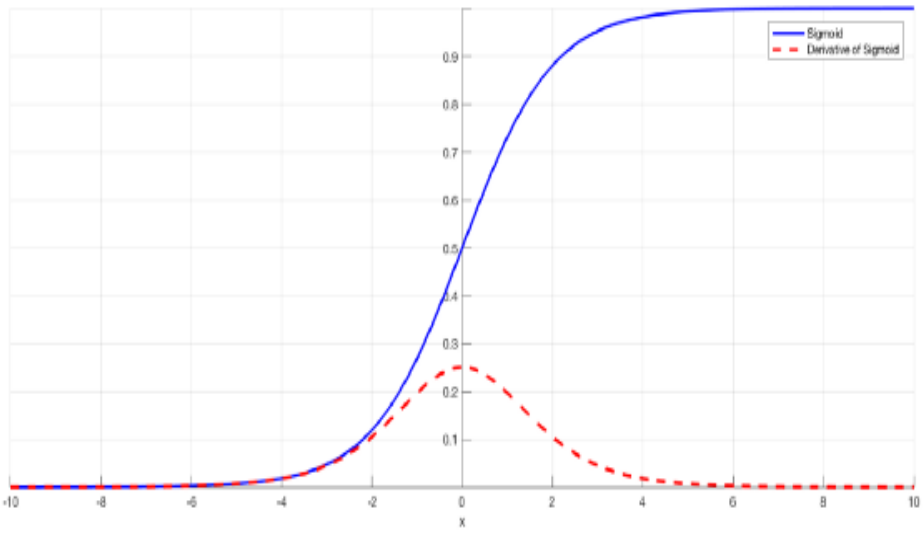

Figure 3.4: Sigmoid function and its derivative

In the case of multiclass logistic regression, with *K* classes, the predicted probabilities are determined by using the softmax function (**3.5**).

$$P(class = 1) = \sigma(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{3.5}$$

Although the mean squared error cost function can be used with the softmax activation function, cross-entropy is the preferred loss function for classification NN with softmax activation in the last layer. It is given by the function (**3.6**):

$$C = -\frac{1}{k}\sum_{k=1}^{K}\left[y_k \log(\sigma(z_k)) + (1 - y_k)\log(1 - \sigma(z_{k}))\right] \tag{3.6}$$

Another popular activation function is the Rectified Linear Unit (ReLU) activation function (**3.7**). It is used in almost all convolutional neural networks or deep learning.

$$R(z) = \max(0, z) \tag{3.7}$$

As its name implies, the ReLU function represents a response similar to a half-bridge rectifier, with all negative values leading to an output of zero and the output equal to the positive values. Figure 3.5 visually compares the ReLU and Sigmoid activation functions, with the sigmoid function having a linear transition between saturation points at (0;1) and an intersection at the midpoint where the input is zero.

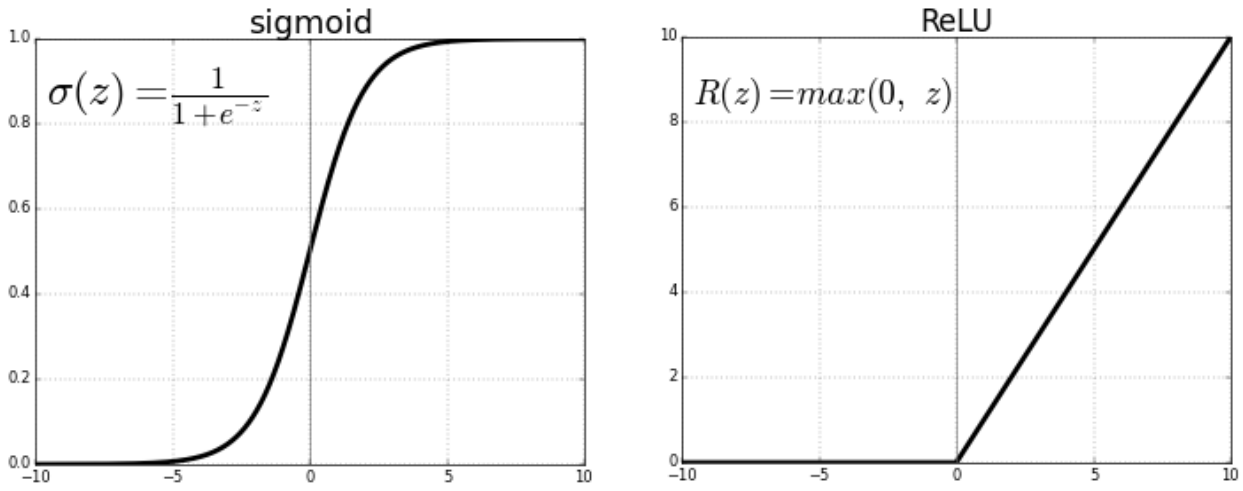

Figure 3.5: ReLU vs Sigmoid

3.4.3 Optimisation / minimisation

Like most machine learning algorithms, the primary challenge of neural networks is an optimisation problem. Solving this optimisation problem is a function of updating and tuning the Neural Network Model's parameters in a direction as to minimise an associated cost function. The cost function can contain potentially many minima, but usually, only one that provides the lowest possible value of the function, known as the global minimum. A critical part of the optimisation process is to ensure that the global minimum is resolved and to avoid getting trapped in local minima. The cost function depends on the biases and synaptic weights of the layers of the neural network and the goal is to

optimise these adaptive parameters to minimise the cost function. The simplest optimisation algorithm is the Gradient descent method which is a recursive process of finding the local minima by evaluating at each step the direction where the gradient is the steepest and converging in that direction (**3.8**).

$$\theta_{i+1} := \theta_i - \alpha \frac{d}{d\theta_i} J(\theta_0, \theta_1) \tag{3.8}$$

If the learning rate parameter, α is small, gradient descent might take a long period to converge, but if it is too large, it can fail to converge to the local minima and can even diverge. The derivative part of the function does help to mostly avoid this by reducing the step size as it approaches the minima. The evaluation of these error derivatives with respect to the layer weights (ω) through all the layers of the MLP is accomplished by using a version of the chain rule of differentiation, thus back-propagation of the error. This method requires significantly more training iterations than the simple perceptron and the training iteration requirement increases as the number of hidden layers increases. The optimisation continues until either the loss error decreases below a set threshold or the reduction rate of the error does not reduce significantly or a predetermined maximum amount of epochs has been reached. These conditions are guided by diminishing returns and limits on computing time. In cases where the derivative of the cost function can include multiple local minima, an optimisation algorithm is used called Newton's method, which is a good method of approximating the roots of higher-order polynomial functions.

Newton's method is a second-order algorithm and it makes use of the Hessian matrix (**3.9**). Newton's method for unconstrained minimisation of $f(x)$, where f is convex and is twice continuously differentiable, is expressed as:

$$x_{k+1} = x_k - \alpha \nabla^2 f(x_k)^{-1} \nabla f(x_k) \tag{3.9a}$$

$$= -\alpha H_k^{-1} \nabla f(x_k) \tag{3.9b}$$

where $\nabla^2 f(x_k)^{-1}$ is the inverse Hessian and $\nabla f(x_k)$ is the gradient of the current $f(x)$ and α is the gradient step size, also called the training rate or learning rate.

The Hessian matrix is composed of the second partial derivatives of the cost function. Due to the complexity of evaluating the Hessian matrix and calculating its inverse, this method is computationally expensive. To mitigate this problem, variable metric methods such as the quasi-newton method are used to approximate Newton's method in a way that is more computationally efficient. Instead of calculating the Hessian directly and then evaluating its inverse, the inverse Hessian can be approximated by another matrix using only the first partial derivatives of the cost function. In this study, the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm is used to perform the optimisation process. BFGS is a quasi-Newton method algorithm for iteratively solving nonlinear optimisations. The steps for optimising the back-propagation parameters using the BFGS algorithm are as follows (Vandenberghe, 2019):

Given a starting point $x_0 \in dom\, f$, $f(x)$ denotes the objective function to be minimized and an estimated initial Hessian $H_0 = 0$, for k = 0, 1, ... :

i. Compute quasi-Newton direction:

$$\Delta x_k = -H_k^{-1} \nabla f(x_k) \tag{3.10}$$

ii. Compute the step size:

$$\alpha_k = \arg\min f(x_k + \alpha \Delta x_k) \tag{3.11}$$

iii. Update the estimate:

$$x_{k+1} = x_k + \alpha_k \Delta x_k \tag{3.12}$$

iv. Find the iteration difference:

$$\delta = x_{k+1} - x_k \tag{3.13}$$

v. And the gradient difference:

$$\gamma \;\; = \nabla f(x_{k+1}) - \nabla f(x_k) \tag{3.14}$$

vi. Compute the updated Hessian matrix:

$$H_{k+1} = H_k + \frac{\gamma . \gamma^T}{\gamma^T \delta} - \frac{H_k \delta {H_k}^T \delta^T}{\delta^T H_k \delta} \tag{3.15}$$

vii. Find the inverse of the update Hessian:

$$H_{k+1}^{-1} = H_k^{-1} + \frac{(\delta^T \gamma + \gamma^T H_k^{-1} \gamma)(\delta \delta^T)}{(\delta^T \gamma)^2} - \frac{H_k^{-1} \gamma \delta^T + \delta \gamma^T H_k^{-1}}{\gamma^T H_k^{-1}} \tag{3.16}$$

3.5 Summary

This chapter provides an overview of the classification models that were used in this study, a support vector machine (SVM) classifier (Section 3.3) as a baseline, but also used in evaluating and selecting the optimal feature set and a feed-forward multi-layer perceptron (MLP) neural network classifier. Section 3.4 describes the components of the MLP neural network, including the function of the perceptron, the different options for activation functions and finally the process of optimising (section 3.4.3) the parameters of the multi-layer perceptron (MLP), specifically the optimisation using the quasi-Newton method used in the algorithm for this study. In the next chapter, this theory is applied in the development of the algorithm and implementation in software.

Chapter 4 - Development of a neural network-based speech/non-speech discrimination algorithm

4.1 Introduction

This chapter will discuss the steps taken to collect a dataset for training and testing purposes, develop an algorithm for processing the audio files in the corpus, extract audio features from these files and generate statistical abstraction of the raw feature data for analysis. The performance of the audio features is then evaluated to determine the best performing set of features.

4.2 Corpus

For the initial training and testing dataset, a corpus of audio files was compiled representing a balanced mixture of both classes of data. For the speech class, part of the LibriSpeech ASR corpus was used (Korvas *et al.*, 2014). The total duration of the speech data is 233 minutes, 48 seconds. The music class data comprises random samples of various musical genres. The duration of data for this class is 231 minutes and 7 seconds. The audio data were down-sampled and down-mixed to single channel, 16 bit and 16 kHz sample rate and then normalised before splitting into 1-second segments with a 250 ms segment shift. These segments were then further divided into 40 ms frames with a 20 ms offset. For each of these frames, a scalar value for each of the audio features is extracted. The first four statistical moments are then calculated for each feature on the accumulated frame data for the duration of the segment. For four of the features, Chromatic Spectral Entropy, Energy Entropy, Percentage Low Energy Frames and Modified Low Energy Ratio, the values are not calculated on a per-frame basis, but rather on the whole segment at once. The duration of this corpus is compared to those of similar studies in Table 4.1 below. The duration of the corpus content varies significantly between the previous studies, from as little as 20 minutes (El-Maleh *et al.*, 2000) to more than 34 hours in the case of Lavner and Ruinskiy, (2009). In this study, the total duration of the corpus content is 7 hours and 45 minutes, which is more than the average of similar previous studies. The size requirement of the dataset for supervised machine learning is a fundamental question. The amount of data needed should be enough to reasonably approximate the unknown underlying mapping function from inputs to outputs and to reasonably measure the performance

of the model on testing data. It is also important that the corpus is a balanced representation of the various data classes as skewed training data results in a poor approximation.

Table 4.1: Comparison of audio corpus duration

Reference to previous studies	Audio Corpus Duration
This study	7 hour 44 minutes 47 seconds
(Ajmera, McCowan and Bourlard, 2003)	40 minutes
(Alexandre *et al.*, 2006)	2 hours 2min 33sec
(Alexandre *et al.*, 2008)	2 hours 2min 33sec
(Balabko, 1999)	24 minutes
(Barbedo and Lopes, 2006)	> 20 hours
(Beierholm and Baggenstoss, 2004)	41 minutes 20 seconds
(Bugatti, Flammini and Migliorati, 2002)	30 minutes
(Bugatti, Leonardi and Rossi, 1999)	Not stated
(Burred and Lerch, 2004)	75 minutes
(Carey, Parris and Lloyd-Thomas, 1999)	20 hours
(Wu Chou and Liang Gu, 2001)	26 minutes
(El-Maleh *et al.*, 2000)	20 minutes
(Foote, 1997)	77 minutes 28 seconds
(Gallardo-Antolin and Montero, 2010)	2440 audio files, duration not specified
(Giannakopoulos, Pikrakis and Theodoridis, 2006)	3 hours
(Khan and Al-Khatib, 2006)	5 hours 37 minutes
(Lavner and Ruinskiy, 2009)	> 34 hours

(Liu, Wang and Chen, 1998)	70 seconds
(Lu, Jiang and Zhang, 2001)	6 hours
(Lu, Zhang and Li, 2003)	4 hours
(Markaki and Stylianou, 2011)	6 hours
(Muñoz-Expósito *et al.*, 2009)	2 hours
(Panagiotakis and Tziritas, 2005)	3 hour 52 minutes
(Pikrakis, Giannakopoulos and Theodoridis, 2006)	5 hours 40 minutes
(Pinquier, Rouas and E-Obrecht, 2002)	33 minutes 19 seconds
(Saunders, 1996)	2 hours
(Scheirer and Slaney, 1997)	40 minutes
(Sigtia *et al.*, 2016)	~7 hours
(Tzanetakis and Cook, 2002)	6 hours 15 minutes
(Wang, Gao and Ying, 2003)	5 hours
(Williams and Ellis, 1999)	40 minutes

4.3 Algorithm Development

For the development of the algorithm, the Python programming language was used. Python is a very popular programming language for many well-known reasons, notably, its ease of use and wide support from an active community of users (Cass, 2018), (Voskoglou, 2017). A general critique of Python has always been that since it is an interpreted programming language, it tends to be slow compared to other compiled programming languages. With the implementation of the Python just-in-time compilation of time-critical functions, usage of the Cython C optimised static compiler and the use of multi-processing, this drawback has been mostly negated (Behnel *et al.*, 2011).

Python is however very well suited towards machine learning and deep-learning, specifically with the inclusion of various add on libraries. There are higher-level libraries specific geared towards deep learning such as TensorFlow, Theano and PyTorch and some expansion libraries like Numpy and Scikit Learn. It is curious to note that by default python does not cater for an array data type. For this reason, the Numpy library is used which facilitates the efficient calculation of matrix mathematics, a crucial part of neural network computation.

4.3.1 Structure of the algorithm

The structure of the algorithm is explained in a flow diagram (Figure 4.1). The program consists of two major parts, the one part being is tasked with reading and analysing the audio files, extracting the features and storing the statistical data in storage media, while the other part of the program then reads the statistical data and trains the neural network classifier by optimising the adaptive parameters, to minimise the loss function or to perform the classification function, comparing the result of the neural network classifier to the SVM benchmark classifier.

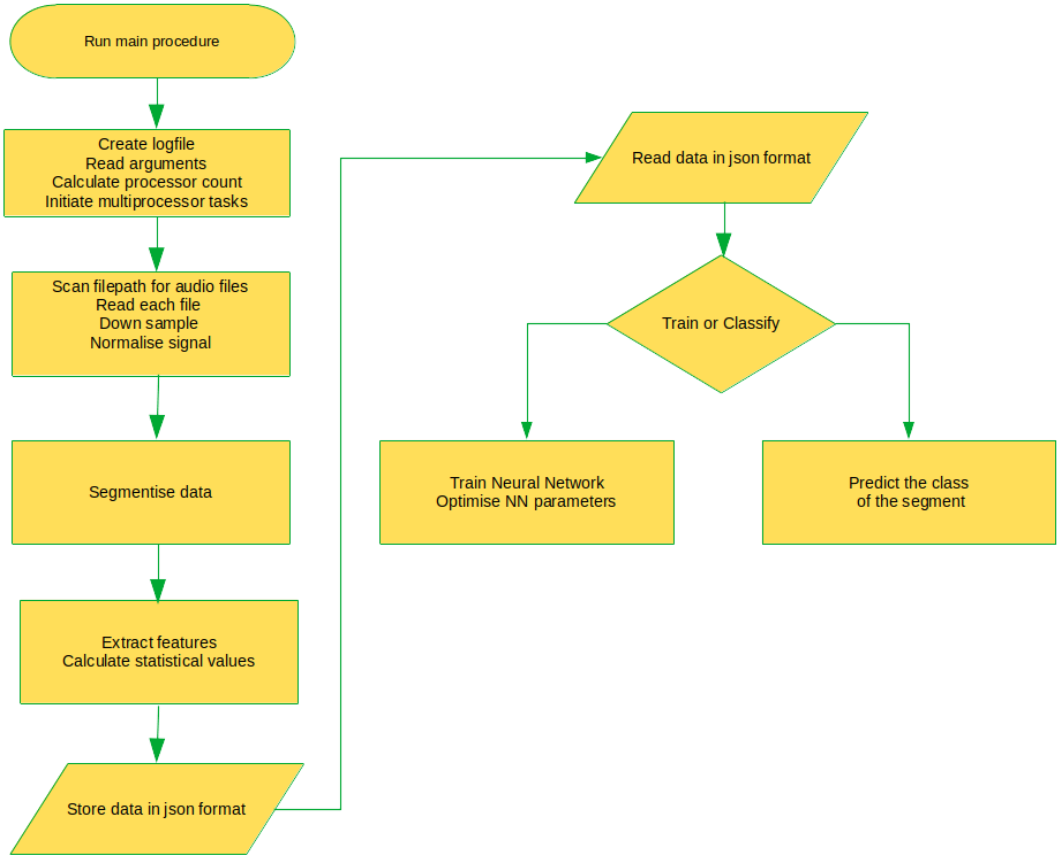

Figure 4.1: Structure of the algorithm

4.4 Signal pre-processing

The signal pre-processing stage consists of resampling the input signal to 16 kHz, converting the input signal from stereo to mono by adding together half the value of each signal and then normalising the signal level by removing any offset from zero mean and rescaling the maximum amplitude to 16 bits as discussed in section 2.3.1.

The following code excerpts show how the various signal preprocessing steps were implemented in Python. Firstly, resampling of the audio signal if the sample rate is not 16 kHz as part of homogenising the signal:

```
with wave.open(wfile,'rb') as wf:
    nchannels, sampwidth, srate, nframes, comptype, compname = wf.getparams()
    signal = np.frombuffer(wf.readframes(-1), dtype='int16')
    # Resample if sample rate is higher or lower than desired
    if srate != samplerate:
```

```
        n_frames = wf.getnframes()
        signal = wf.readframes(n_frames)
        cutoff_freq = (samplerate/2)*0.9
        # including lowpass filter
        signal = firfilter(signal, cutoff_freq, srate)
        scale_factor = math.ceil(srate/samplerate)
        signal = sp.signal.decimate(signal, scale_factor,
                                    n=None, type='fir', zero_phase=True)
        samplerate = (srate/scale_factor)
```

Convert the signal from stereo to mono:

```
# If signal stereo, convert to mono first
if nchannels == 2:
    lc = signal[0::2]
    rc = signal[1::2]
    signal = lc / 2 + rc / 2
```

Normalise the signal:

```
def normalise_signal(sig):
    """
    DC Removal - Remove any DC offset
    This is achieved by subtracting the arithmetic mean from each sample
    Find the maximum sample value of all of the samples and using the max(abs())
    Because the maximum amplitude we can get with 16 bits is 32767
    We scale the stream so that the largest sample value to 32767
    largest * scale = 32767
    To solve the scale factor use the formula: scale multiplier = 32767/largest
    Now that we have the scale factor, we multiply all the sound samples by
    that amount

    :param sig (numpy array of int):  audio file in .wav format
    :return sig (numpy array of int): normalised audio file
    """
    sig = sig - sum(sig)/len(sig)       # Remove DC offset
    numbits = 16
    max_val = float((2**(numbits-1))-1)      # Calculate max value for given
                                               storage bits
    scale = float(max_val)/(max(abs(sig)))  # Calculate scaling factor
    sig = scale * sig                       # Scale signal to max value
    return(sig)
```

The segmentation step divides the normalised signal content of an audio file into segments of a predetermined duration and an overlapping shift length. This step produces an array of sample amplitude values spanning 1000 ms in duration and each offset by 250 ms. In the feature extraction step, these segments are then further subdivided into 40 ms frames, with a 20 ms offset, producing

a total of 40 frames per segment, each containing 640 sample values. This means that every second of audio data contains 4 segments and 160 frames.

4.5 Feature extraction

For each of these frames, a scalar value for each of the audio features is extracted. The first four statistical moments are then calculated for each feature on the accumulated frame data for the duration of the segment. For four of the features, Chromatic Spectral Entropy, Energy Entropy, Percentage Low Energy Frames and Modified Low Energy Ratio, the values are not calculated on a per-frame basis, but rather on the whole segment at once. The feature extraction for both classes was completed in 4 hours and 18 minutes processing parallel on a single CPU with 4 cores (Intel i7-4800MQ). The following code excerpts show how the feature extraction was implemented in Python.

4.5.1 Zero-Crossing Rate

```
def feat_zcr(frame):
    """
    Calculate the Zero Crossing Rate value of the frame
    :param frame (numpy array of int): the array of samples in a frame.
    Each row is a frame (640 samples @ 25ms).
    :return zcr (float): The average zero crossings in the frame
    """
    signs = np.int8(np.sign(frame))       # find sign of each array element
    signs[signs == -1] = 0
    signs_min_one = np.append(signs[1:],1)   # move sequence one place on
    crossings = signs_min_one^signs          # Bitwise XOR
    zcr = float(len(np.where(crossings)[0]))/len(frame)
    return(zcr)
```

4.5.2 Signal energy and Short Time Energy

```
def feat_ste(frame):
    """
    Calculate the Short Time Energy (in decibels) value of the frame
    :param frame (numpy array of int): the array of samples in a frame.
     Each row is a frame (640 samples @ 25ms)
    :return frame_energy (float): the RMS energy of the frame
    :return ste (float): the Short Time Energy (logarithm of RMS energy)
    of the frame
    """
    frame_energy = (feat_rms_e(frame))
    if frame_energy > 0:
        ste = 10*float(math.log10(frame_energy))
    else:
```

```
        ste = 0
    return(frame_energy, ste)


def feat_rms_e(frame):
    """
    Calculate the RMS Energy value of the frame
    :param frame (numpy array of int): the array of samples in a frame.
    :return rms_e (float): the RMS energy of the frame
    """
    rms_e = math.sqrt(feat_sig_energy(frame))
    return(rms_e)


def feat_sig_energy(frame):
    """
    # Calculate the signal energy of the frame
    # Since audio signals are mostly non-periodic, we regard it as an
      energy signal.
    :param frame (numpy array of int): the array of samples in a frame.
    :return sig energy (float): the signal energy of the frame
    """
    dim = len(frame)
    if dim > 0:
        sig_energy = float(sum(np.square(frame)))
        sig_energy = np.where(sig_energy == 0, np.finfo(float).eps,
                              sig_energy)/dim
        # if sig_energy = zero, set it to a VERY small number.
        # required for log operation in calling function
        return(sig_energy)
    else:
        return(0)
```

4.5.3 Percentage Low Energy Frames / Modified Low Energy Ratio

```
# Percentage Low Energy Frames
def feat_plef_mler(seg, ste):
    """
    Calculate the Percentage Low Energy Frames and Modified Low Energy Ratio
    of the segment.
    Percentage Low Energy Frames is defined by the percentage of frames with
    RMS power less than 50%-70%
    of the mean RMS power within a one second window. The value for speech
    is normally higher than that of music.
    Modified Low Energy Ratio - MLER is defined as the proportion of frames
    with RMS power less than a
    variable threshold within one second. The threshold should be in the
    interval [0.05%, 0.12%] for best
    performance.
    Music shows little variation in energy contour of the waveform, while
    speech shows large variations
    between voicing and frication.
    :param seg (numpy array of int): the array of samples in a segment.
    Each row is a segment (16000 samples @ 25ms)
    :param ste (numpy array of float): the array of Short Time Energy samples
```

```
    in the segment
    :return plef (float): the Percentage Low Energy Frames of the segment.
    :return mler (float): the Modified Low Energy Ratio of the segment.
    """
    mler = 0
    plef = 0
    plef_ratio = 0.5 #0.7
    dim = len(ste)
    seg_rms, seg_ste = feat_ste(seg)
    # PLEF
    threshold = plef_ratio*seg_ste              # Calculate the energy threshold
    [a, ] = np.nonzero(np.array(ste) <= threshold)   # Return the indices of
                                                     # the elements that are
                                                     # non-zero
    if len(a) > 0:
        plef = float(len(a)) / (float(dim))     # Calculate the ratio of elements
                                                 # with energy less than
                                                 # the threshold
    else:
        plef = 0.0
    # MLER:
    level = 0.12
    thres_high = seg_ste*(1+level)
    thres_low = seg_ste*(1-level)
    # Return the indices of the elements that are between the lower and
    # upper threshold
    [b, ] = np.nonzero(np.logical_and(thres_low < np.array(ste),
                      np.array(ste) < thres_high))
    if len(b) > 0:
        mler = float(len(b)) / (float(dim)) # Calculate the ratio of
                                            # elements with energy between
                                            # the threshold levels
    else:
        mler = 0.0
    return(plef, mler)
```

4.5.4 4 Hz Modulation Energy (Hilbert Transform Method)

```
# 4Hz Modulation Energy
def feat_4hz(frame):
    """
    Calculate the 4Hz Modulation Energy of the frame using the Hilbert Transform
    :param frame (numpy array of int): the array of samples in a segment.
     Each row is a segment (16000 samples @ 25ms)
    :return fhz_energy (float): the 4 Hz Modulation energy of the frame.
    """
    hilbert_signal = hilbert(frame)       # Take Hilbert transform of signal
                                          # using the Python SciPy library)
    amplitude_envelope = np.abs(hilbert_signal)
    # Apply 4Hz bandpass filter
    lowpass_envelope = pass_band_firfilter(amplitude_envelope, freq=4,
                                           fs=samplerate)
    # Calculate the periodogram of the envelope
    fhz_periodogram = powspec(lowpass_envelope, len(lowpass_envelope))
```

```
    # Calculate 4Hz energy
    energy = np.sum(powspec(frame, len(frame))) # this stores the total energy
                                                # in each frame
    # if sig_energy = zero, set it to a VERY small number.
    energy = np.where(energy == 0, np.finfo(float).eps, energy)
    mean_energy = np.mean(energy)
    fhz_energy = np.sum(fhz_periodogram)
    # find the ratio of the 4 Hz energy to the mean energy of the frame
    fhz_energy = 10 * np.log10(fhz_energy / mean_energy)

    return(fhz_energy)
```

4.5.5 Entropy features

```
# Chromatic Spectral Entropy
def feat_spec_entropy(Xf):
    """
    Computes the Spectral Entropy value of the segment
    :param Xf (numpy array of int): an N x 1 array of the power
     spectrum of samples in a frame.
    :return se (float): the Spectral Entropy value of the segment
    """
    s = Xf/(np.sum(Xf))
    s = np.where(s == 0, np.finfo(float).eps, s)    # if s = zero, set it to a
                                                    # VERY small number.
    se = -np.sum(s*np.log2(s))
    return(se)

# Energy Entropy
def feat_energy_entropy(seg, sub_energy):
    """
    Calculate the Energy Entrophy value of the segment
    :param seg (numpy array of int): the array of samples in a segment.
    Each row is a segment (16000 samples @ 25ms)
    :param sub_energy (): normalised energies (STE) of the individual frames
    :return entropy (float) the Energy Entrophy value of the segment
    """
    seg_energy = feat_rms_e(seg)
    ent_energy = [e / seg_energy for e in sub_energy]
    ent_energy = np.where(ent_energy == 0, np.finfo(float).eps, ent_energy)
    entropy = -np.sum(ent_energy*np.log2(ent_energy))
    return(entropy)
```

4.5.6 Spectral Centriod

```
# Spectral Centroid
def feat_spec_centroid(Xf):
    """
    Calculate the Spectral Centriod value of the frame. Defined as a
    weighted mean of the frequencies in the power spectrum
    :param Xf (numpy array of int): an N x 1 array array of the power
    spectrum of samples in a frame.
    :return sc (float): the Spectral Centroid value of the frame
```

```
    """
    Xt = 0.0
    # Array of freq samples between 0 and Nyquest freq,
    # spaced at nFFT/2+1 increments
    ind = (np.arange(0, len(Xf + 1)) * (samplerate/(2.0 * len(Xf))))
    Xf = (Xf / Xf.max())  # Normalise Xf
    if np.sum(Xf) != 0:
        sc = (np.sum(ind * Xf) / np.sum(Xf))/(samplerate / 2.0)
        # Spread:
        ss = np.sqrt(np.sum(((ind - sc) ** 2) * Xf) / np.sum(Xf))
    return(sc)
```

4.5.7 Spectral Rolloff

```
# Spectral Roll-off
def feat_spec_rolloff(Xf):
    """
    Calculate the Spectral Rolloff value of the frame. This is the number
    of frequency bins with a cumulative energy
    in the power spectral distribution is concentrated below a threshold
    percentage n%.
    :param Xf (numpy array of int): an N x 1 array array of the power spectrum
    of samples in a frame.
    :return (float): the Spectral Rolloff value of the frame
    """
    limit = 0.90    # the threshold energy %
    total_energy = np.sum(Xf)  # this stores the total energy in each frame
    dim = len(Xf)
    threshold = limit*total_energy
    cumulative_energy = np.cumsum(Xf)  # cumulative energy in the frame
    [a, ] = np.nonzero(cumulative_energy > threshold)
    if len(a) > 0:
        sr = np.float64(a[0]) / (float(dim))
    else:
        sr = 0.0
    return (sr)
```

4.5.8 Spectral Flux

```
# Spectral Flux
def feat_spec_flux(Xf, Xp):
    """
    Calculate the Spectral Flux value of the frame
    :param Xf (numpy array of int): an N x 1 array of the power spectrum
    of samples in a frame.
    :param Xp (numpy array of int): an N x 1 array of the power spectrum
    of previous frame.
    :return sf (float): the Spectral Flux value of the frame
    """
    sumX = np.sum(Xf)           # stores the total energy in the frame
    sumPrevX = np.sum(Xp)       # stores the total energy in the previous frame
    # compute the spectral flux as the sum of square distances
    if sumPrevX != 0:
```

```
        sf = np.sum((Xf / sumX - Xp / sumPrevX) ** 2)
    else:
        sf = 0.0
    return(sf)
```

4.5.9 Mel Frequency Cepstral Coefficients (MFCC)

4.5.9.1 *Mel Filter Bank*

```
# Mel Filter Bank
def hz2mel(hz):
    """Convert a value in Hertz to Mels
    :param hz: a value in Hz. This can also be a numpy array, conversion
    proceeds element-wise.
    :returns: a value in Mels. If an array was passed in, an identical sized
    array is returned.
    """
    return 2595 * np.log10(1+hz/700.0)


def mel2hz(mel):
    """Convert a value in Mels to Hertz
    :param mel: a value in Mels. This can also be a numpy array, conversion
    proceeds element-wise.
    :returns: a value in Hertz. If an array was passed in, an identical sized
    array is returned.
    """
    return 700*(10**(mel/2595.0)-1)


def mel_filter(nFFT=512):
    """
    Compute the Mel filter bank. Filter banks using the Mel scale are used to
    compute Mel-Frequency Cepstral
    Coefficients (MFCCs). The filter bank is an array of overlapping triangular
    bandpass filters.
    :param nFFT(int): the FFT length (bins) to use for dividing the window into
    equal bins.
    :return fbank (numpy array of float): an D x N array of values representing
    the dimensions of the mel frequency filter bank. (D = 40 default)
    """
    mf = 40                  # number of mel filters
    fmax = samplerate/2      # maximum frequency (Nyquist frequency)
    fmin = 2                 # minimum frequency
    mel_l = hz2mel(fmin)     # lower mel freq (2595 * np.log10(1+hz/700.0))
    mel_h = hz2mel(fmax)     # upper mel freq (2595 * np.log10(1+hz/700.0))
    mel_f = np.zeros(mf+2)   # array of zeros, length is the number of mel
                             # filters + 2
    freq = np.zeros(mf+2)    # array of zeros, length is the number of mel
                             # filters + 2
    fft_bins = np.zeros(mf+2)    # array of zeros, length is the number of mel
                                 # filters + 2
    mel_increment=(mel_h - mel_l)/(mf+1)
    for i in range(0,mf+2):
        mel_f[i] = mel_l + (i* mel_increment)  # find the centres of the filter
```

```
                                                # bank elements in mel values
        freq[i] = mel2hz(mel_f[i])              # find the centres of the filter
                            # bank elements in Hz. (Hz = 700*(10**(mel/2595.0)-1))
        fft_bins[i] = np.floor((nFFT+1)*freq[i]/samplerate)
    fbank = np.zeros([mf,int(nFFT/2+1)]) # D x N array of zeros.
                                         # D = number of mel filters and
                                         # N = nFFT/2 + 1
    # Calculate filter bank parameters
    for j in range(0,mf):
        for i in range(int(fft_bins[j]),int(fft_bins[j+1])):
            fbank[j,i] = (i - fft_bins[j])/(fft_bins[j+1]-fft_bins[j])
        for i in range(int(fft_bins[j+1]),int(fft_bins[j+2])):
            fbank[j,i] = (fft_bins[j+2]-i)/(fft_bins[j+2]-fft_bins[j+1])
    return(fbank)
```

Figure 4.2 illustrates the Mel filter bank output. Filter banks using the Mel scale are used to compute Mel-Frequency Cepstral Coefficients (MFCCs). The Mel-scale simulate the non-linear human hearing perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. The filter bank is an array of overlapping triangular bandpass filters, used to map frequency bins from short-time Fourier transform (STFT) to Mel bins.

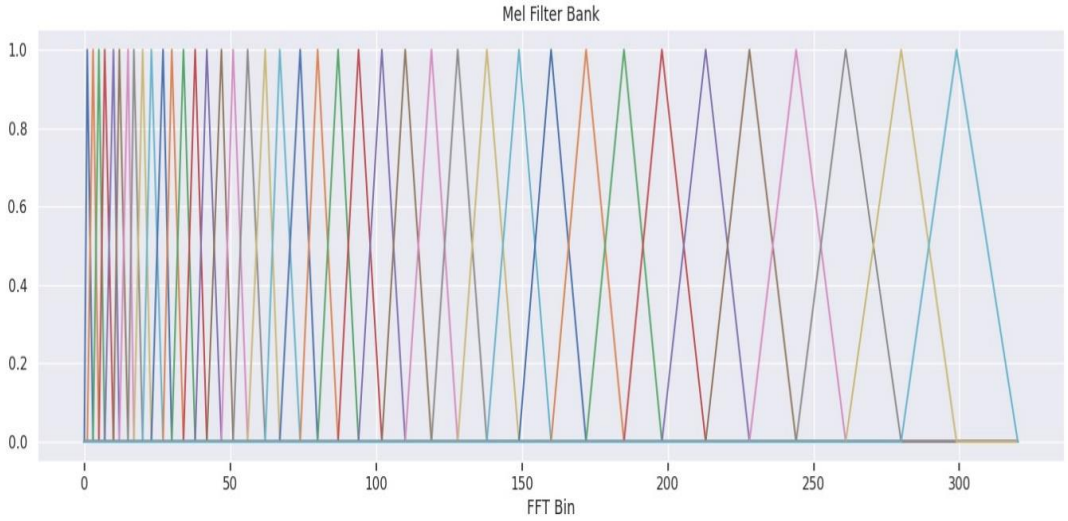

Figure 4.2: Mel Filter Bank

4.5.9.2 *Mel Frequency Cepstral Coefficients and 4Hz Modulation Energy*

```
# Mel Frequency Cepstral Coefficients (MFCC)
def feat_mfcc(fbank, Xf):
    """
    Calculate the mfcc of the frame and the 4Hz Modulation Energy
    Steps:
    1. Divide the signal into short frames (already done).
    2. Take FFT of frame
    3. For each frame calculate the estimate of the power spectrum.
    4. Apply the mel filterbank to the power spectra, sum the energy in
```

```
        each filterbank.
    5. Take the logarithm of all filterbank energies.
    6. Take the Discrete Cosine Transform (DCT) of the log filterbank energies.
    7. Keep DCT coefficients 1-12, discard the rest.
    :param fbank (numpy array of float): an D x N array of values representing
    the dimensions of the mel frequency filter bank. (D = 40 default)
    :param XF (numpy array of int): an N x 1 array of the power spectrum of
    samples in a frame.
    :return mfcc (numpy array of float): array containing the MFCC coefficients
    of each frame.
    :return fhz_me (float): scalar containing the 4Hz Modulation Energy result
    of each frame
    """
    mfcc = []
    power_spectrum = Xf
    # compute the filterbank energies:
    energy_0 = np.sum(power_spectrum)       # this stores the total energy
                                            # in each frame
    # if energy_0 = zero, set it to a VERY small number.
    energy_0 = np.where(energy_0 == 0,np.finfo(float).eps,energy_0)
    energy = np.dot(power_spectrum,fbank.T)  # dot product of power spectrum
                                             # and filter bank transposed
    energy = np.where(energy == 0,np.finfo(float).eps,energy)
    #calculating the cepstral coefficients
    log_energy = np.log(energy)
    mfcc = dct(log_energy, norm='ortho')[0:ncep] # taking the DCT of the log
                                                 # of the energy coefficients.
                                                 # keeping only the first ncep
                                                 # coefficients
                                                 #(ncep=13 default)
    # replace first cepstral coefficient with log of frame energy
    mfcc[0] = 10*np.log10(energy_0)
    # Calculate 4Hz Modulation Energy:
    # apply FIR bandpass filter to find energy at 4Hz
    fhz_energy = pass_band_firfilter(energy, freq=4, fs=samplerate)
    mean_energy = np.mean(energy)
    fhz_energy = np.sum(fhz_energy)
    fhz_me = 10*np.log10(fhz_energy/mean_energy) # find the ratio of the 4 Hz
                                    # energy to the mean energy of the frame
    return(mfcc, fhz_me)
```

4.5.9.3 *Magnitude Spectrum*

```
def magspec(fraction,nFFT=512):
    """Compute the magnitude spectrum of each frame or fraction.
     If fraction is an NxD matrix, output will be NxnFFT.
    :param fraction: the array of frames. Each row is a frame.
    :param nFFT: the FFT length to use. If nFFT > frame_len,
    the frames are zero-padded.
    :returns: If fraction is an NxD matrix, output will be NxnFFT.
    Each row will be the magnitude spectrum of the corresponding frame.
    """
    complex_spec = np.fft.rfft(fraction,nFFT)
    ms = np.absolute(complex_spec)
    return(ms)
```

4.5.9.4 *Power Spectrum*

```
    # Calculate power spectrum compute the periodogram
def powspec(fraction,nFFT=512):
    """Compute the power spectrum of each frame or fraction.
    If fraction is an NxD matrix, output will be NxnFFT.
    :param fraction: the array of frames. Each row is a frame.
    :param nFFT: the FFT length to use. If nFFT > frame len,
    the frames are zero-padded.
    :returns: If fraction is an NxD matrix, output will be NxnFFT.
    Each row will be the power spectrum of the corresponding frame.
    """
    tp = (sum(magspec(fraction,nFFT))**2)/nFFT
    tq = sum((magspec(fraction,len(fraction))**2)/len(fraction))

    if len(fraction) > 0:
        ps = 1.0/(len(fraction)*samplerate)*
                np.square((magspec(fraction,nFFT)**2)/nFFT)
    else:
        ps = 0
    return(ps)
```

4.5.9.5 *Bandpass Filter*

```
# 4Hz FIR bandpass filter
def pass_band_firfilter(src, freq=4.0, fs=samplerate):
    """
    :param src (numpy array of float): the signal input to the filter
    :param freq (float): centre frequency of the filter
    :param fs (int): sampling frequency
    :return (numpy array of )
    """
    order = 500
    nyq = 0.5 * fs      # Find the Nyquist frequency of the signal.
    lowcut = (freq -2)/nyq
    highcut = (freq + 2)/nyq
    # calculate the filter parameters. Using Python SciPy library
    coefficients = bandpass_firwin(order + 1, lowcut, highcut, fs=fs)
    a = 1
    # Apply filter to input signal twice
    firstpass = sp.signal.lfilter(coefficients, a, src)
    secondpass = sp.signal.lfilter(coefficients, a, firstpass[::-1])[::-1]
    return(secondpass)
```

4.6 Classifier functions

The following code excerpts show how the classifier models were implemented in Python.

4.6.1 Support Vector Machine

```
def svm_classifier(trainX,trainY,testX,testY):
    #------------------------------------------------------------------------
    # Train & Test Support Vector Machine
    #------------------------------------------------------------------------
    #
    """
    Support Vector Machine classifier implementation of Python Sklearn library
    :param trainX (array of float): NxD array of training samples.
    N samples x D features
    :param trainY (array of float): NxD array of annotation of training data.
    N samples x D data classes
    :param testX (array of float): NxD array of test samples.
    N samples x D features
    :param testY (array of float): NxD array of annotation of test data.
    N samples x D data classes
    """
    t1 = time.time()
    clf = svm.SVC(verbose=True, kernel='rbf')
    clf.fit(trainX, trainY)
    delta_T = time.time() - t1
    print('\nModel training time: ' + str(delta_T))
    print('SVM Score: '+ str(clf.score(trainX, trainY)))
    y_pred = clf.predict(testX)
    print(metrics.classification_report(np.array(testY), y_pred))
```

4.6.2 Feed Forward Neural Network

```
def neural_net(trainX,trainY,testX,testY):
    """
    Train and predict with a MLP Neural Network
    :param trainX (array of float): NxD array of training samples.
    N samples x D features
    :param trainY (array of float): NxD array of annotation of training data.
    N samples x D data classes
    :param testX (array of float): NxD array of test samples.
    N samples x D features
    :param testY (array of float): NxD array of annotation of test data.
    N samples x D data classes
    :return score (float): NxD array of precision, recall, F-measure
    and support for each class.
    :return ypred (array of float): NxD array of predicted results of test data.
    N predictions x D data classes
    """
    # Globalise Initialise neural network parameters to enable writeback
    global W1
    global W2
    global J
    global J_test
```

```
    global hidden_layer
    global input_layer
    global output_layer

    # Set layer width
    hw = ceil(trainX.shape[1]/6)
    input_layer = trainX.shape[1]
    output_layer = trainY.shape[1]
    hidden_layer = (hw if hw > 4 else 5)  # Hidden layer width to be the same
                                          # dim as input layer, but not less than 5

    # Weights (initial parameters)
    W1 = np.random.randn(input_layer, hidden_layer)
    W2 = np.random.randn(hidden_layer, output_layer)
    J = []
    J_test = []

    # Train neural network
    t1 = time.time()
    train(trainX, trainY)
    print('Shape X training data: ' + str(trainX.shape))
    print('Shape Y training data: ' + str(trainY.shape))
    delta_T = time.time() - t1
    print('Model training data: ' + str(len(trainX)))
    print('Model training time: ' + str(delta_T))
    print('Model testing data: ' + str(len(testX)))

    # Predict neural network
    r,y_pred = predict(testX)
    score = metrics.precision_recall_fscore_support(np.array(testY), y_pred)
    return (score, y_pred)

def nn_setparams(params):
    """
    Update the NN parameters
    :param params (float): Array of new parameters to be written back
    :return:
    """
    global W1
    global W2
    global hidden_layer
    global input_layer
    global output_layer
    W1_start = 0
    W1_end = hidden_layer * input_layer
    W1 = np.reshape(params[W1_start:W1_end], (input_layer, hidden_layer))
    W2_end = W1_end + hidden_layer * output_layer
    W2 = np.reshape(params[W1_end:W2_end], (hidden_layer, output_layer))
    return ()


def sigmoid(z):
    """
    Apply sigmoid activation function.
    :param z (array of float): NxD array N the weighted values x D layer
    width at the activation node
    :return sig (array of float): NxD array N the decision result (0:1) x D
```

```
    layer width at the activation node
    """
    sig = 1 / (1 + np.exp(-z))
    return (sig)



def nn_forward(X):
    """
    Propogate input values forward though network
    :param X (array of float): NxD array of samples. N samples x D features
    :return yH (array of float): NxD array of forward results of NN.
    N output values x D data classes
    """
    z2 = np.dot(X, W1)  # Calculate first layer weighted inputs
    a2 = sigmoid(z2)    # Calculate first layer activation output
    z3 = np.dot(a2, W2) # Calculate second layer weighted inputs
    yH = sigmoid(z3)    # Calculate second layer activation output
    return (yH)


def cost_function(X, y)
    """
    Compute cost for given X,y, use weights already stored in class.
    :param X (ndarray): Input NxD array of test samples. N samples x D features
    :param y: (ndarray): NxD array of true results for samples.
    N samples x D features
    :return: (float or ndarray) Norm of the matrix or vector
    """
    y_hat = nn_forward(X)
    J = 0.5 * np.sum((y - y_hat) ** 2) / X.shape[0] + (Lambda /
       (2 * X.shape[0])) * (np.sum(W1 ** 2) + np.sum(W2 ** 2))
    return np.linalg.norm(J)

def cost_function_prime(X,y):
    """
    Compute derivative with respect to W1 and W2 for a given X and y:
    :param X (ndarray): Input NxD array of test samples. N samples x D features
    :param y: (ndarray): NxD array of true results for samples.
    N samples x D features
    :return the back probagation derivatives for each layer
    """
    y_hat = nn_forward(X)
    z2 = np.dot(X, W1)
    a2 = sigmoid(z2)
    z3 = np.dot(a2, W2)
    # Calculate the L2 delta for each class per sample
    delta3 = np.multiply(-(y - y_hat), sigmoid_prime(z3))
    # Calculate the back prob derivative for W2
    dJ_dW2 = np.dot(a2.T, delta3) + Lambda*W2
    # Calculate the L1 delta for each class per sample
    delta2 = np.dot(delta3, W2.T) * sigmoid_prime(z2)
    # Calculate the back prob derivative for W1
    dJ_dW1 = np.dot(X.T, delta2) + Lambda*W1
    return (dJ_dW1, dJ_dW2)


def compute_gradients(X, y):
```

```
    """
    Compute the gradient function
    :param X: ndarray Input NxD array of test samples. N samples x D features
    :param y: ndarray NxD array of true results for samples.
    N samples x D features
    :return: Return a contiguous flattened array of the gradient at each node
    """
    dJdW1, dJdW2 = cost_function_prime(X, y)
    gr = np.concatenate((dJdW1.ravel(), dJdW2.ravel()))
    return (gr)

def process_cf(params, X, y):
    """
    Process the cost function method
    :param params: flattened array of the weight parameters at each node
    :param X: Input NxD array of test samples. N samples x D features
    :param y: NxD array of true results for samples. N samples x D features
    :return cost (float): magnitude of the cost
    :return grad: flattened array of the gradient at each node
    """
    nn_setparams(params)
    cost = cost_function(X, y)
    grad = compute_gradients(X, y)
    return (cost, grad)
```

The algorithm also includes various small functions used to evaluate and rank combinations of the audio features, there include Principal Component Analysis (PCA) for exploratory dimensionality reduction analysis, univariate feature selection, Random Forest Regression, forward and backward Sequential Feature Selection and the highest loading factors of the PCA components.

4.7 Summary

Chapter 4 describe the implementation of the algorithm in the Python programming language. Section 4.2 describes the composition of the data corpus for training and testing purposes, and section 4.3 illustrates the structure and flow of the algorithm. Software code excerpts are provided to show the functionality of algorithm functions, including the signal pre-processing steps (section 4.4), various functions for extracting audio features from the data (section 4.5) and the code excerpts for the SVM and MLP neural network classifier models in section 4.6.

Chapter 5 - Experimental Evaluation of model for audio features for speech/non-speech discrimination

5.1 Feature selection

Starting with a large and diverse set of features, it is not known how much each feature would contribute to the overall classification performance individually and if the selected feature or combination of features is well suited for the classification task at hand. Some features may be effective for one class, but not very good for another class and some might deliver moderately to well, yet not exceptional results in all classes. Applying several of these "all-rounder" features in neural networks is useful and improves both classifier accuracy and recall. Another unknown factor was if a certain combination of features might reduce overall performance. Having too many irrelevant features in your data can decrease the accuracy of the model. The performance of the algorithm is indirectly proportional to the dimension of the feature space, it is, therefore, important to keep it as small as possible. There are important benefits of performing feature selection before modelling your data. Firstly, it reduces overfitting, less redundant data means less opportunity to make decisions based on noise and less misleading data improves modelling accuracy, while also reducing the training time. It is important to perform feature selection on a different dataset than you train the model on, to prevent over fitment of your training data.

The selection of feature combinations chosen empirically through exhaustive experimentation is not practical due to the large number of features extracted. With a total of over 21 features computed on the frame level with 4 statistical parameters each and computed 4 times per segment, the resultant feature space is quite large. Considering a total feature space of 88 features, it leads to 3.094850098×10^{26} possible combinations of features, clearly impossible to attempt to evaluate through brute force methods.

To gain a better understanding of the feature set, exploratory dimensionality reduction analysis was performed with Principal Component Analysis (PCA). The cumulative explained variance ratio (Figure 5.1) indicates that it is not a case of only a few features that contribute massively but rather that all or most features contribute meaningfully to the overall result. 21.86% (n=19) of the features

are responsible for explaining one standard deviation (>68.27%) of the data and 64.77% (n=57) of the features describe the data to 2 standard deviations (>95.45%).

Having calculated the principal components for the feature set, it is interesting to now also produce the loadings plot for each of the principal components, for example, Figure 5.2 show the loadings factors, these are the correlated features that contribute the most differentially towards the first (main) principal component. Some features are positively correlated, for instance in this plot, the variance of the Spectral Centroid (sc_var) and the variance of the Spectral Rolloff (sr_var) feature show a strong positive correlation and some features are negatively correlated.

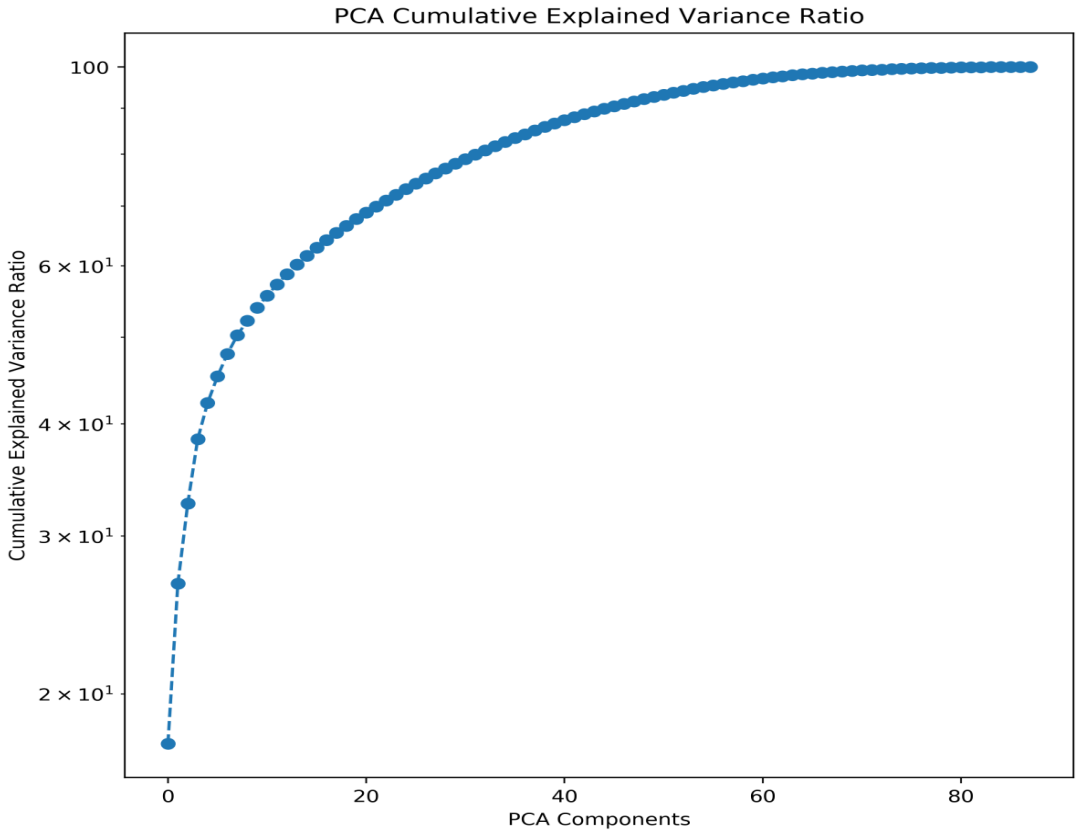

Figure 5.1: PCA Cumulative Explained Variance Ratio.

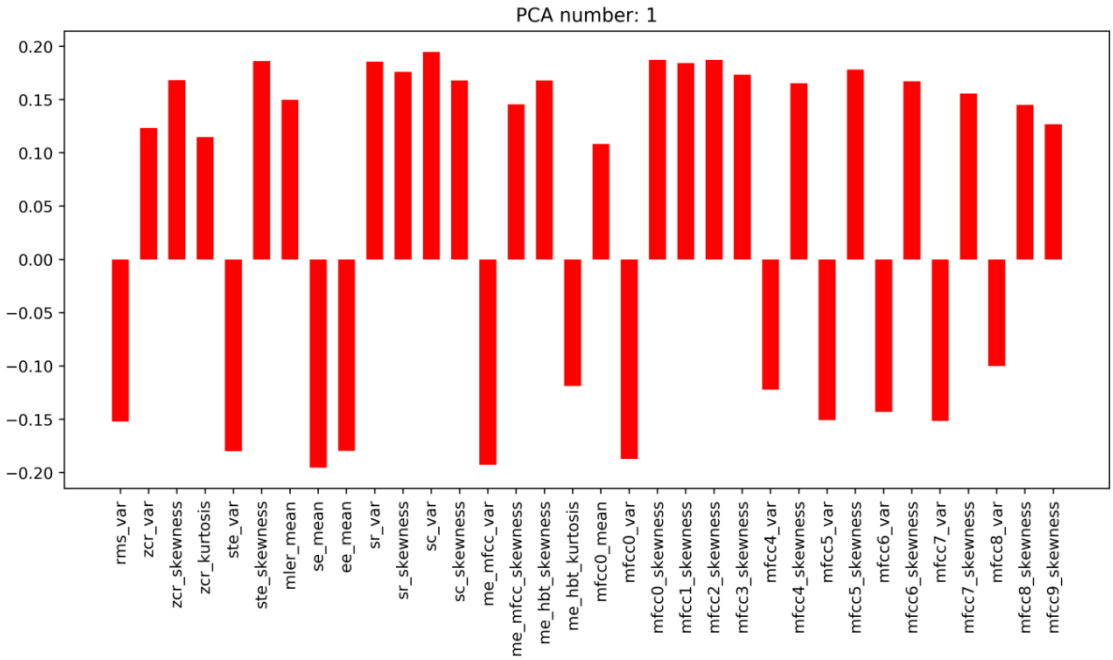

Figure 5.2: Loadings Plot for the first principle component.

Several feature selection techniques were employed to reduce an initial d-dimensional feature space to a k-dimensional feature subspace where k < d, i.e., the optimal feature set. As a first step, each feature was listed in descending order on mean accuracy (MA) and compared to the mean accuracy obtained by all the features (n=88) combined. The training and testing data were generated by selecting two independent data sets by random sampling. The mean accuracy for the SVM classifier was used to rank the features. The Scikit-learn Univariate feature selection algorithm was also used to verify the results obtained by manual selection. It works by selecting the best features based on univariate statistical tests. The result of the Univariate feature selection test is represented in **Error! Reference source not found.**, ranking features in descending order of effectiveness. The mean and variance of the Spectral Flux feature ranked highest in this test. The features that scored in the top 50% for the Univariate (n=44) were selected to represent this test against the other feature selection methods.

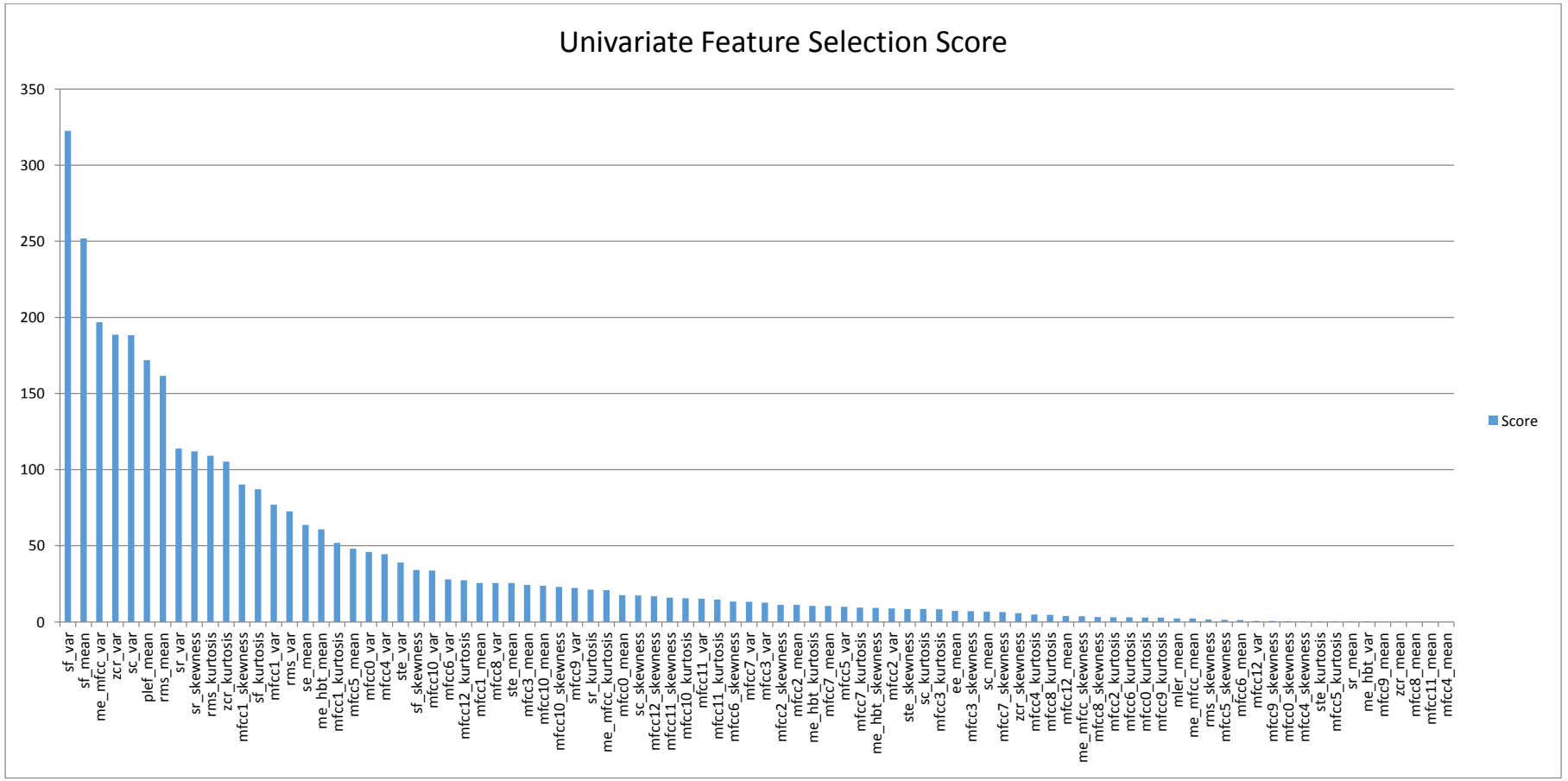

Figure 5.3: Feature ranking with Univariate selection method

The next method of feature selection considered was Random Forest Regression (RFR). Random forests are not only used as a classification and regression tool but can also be used as a method for feature selection by applying properties of either mean decrease Gini impurity or information gain/entropy. Random forests are an ensemble of several decision trees. Every node in the decision tree aims to divide the dataset using the properties of a single feature assigned to the node. The optimal condition is based on a property called impurity. In the case of classifier trees, the property is typically either Gini impurity or information gain/entropy and for regression trees it is variance. It can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure. An alternative method of calculating the feature importance in Random Forest Regression is called Mean Decrease Accuracy and in this method, the feature importance calculation is based on the mean decrease in the accuracy. The feature set is randomly shuffled and a feature gets selected to evaluate what impact that feature exerts on the model's performance. The features with the greatest positive impact on the performance are the most important.

Another method evaluated was the forward and backward Sequential Feature Selection (SFS) was used to extract further candidates for optimal feature sets. Forward Sequential Feature Selection, is a greedy search algorithm that considers the full d-dimensional feature set as input and starts with an empty subset of the original feature set. First, the best performing single feature is added and then a specified number of features are recursively included in the subset based on criteria of which feature maximises the result when used in combination with the existing features in the subset. This method is still very computationally expensive, with $\sum_{n=k}^{d} \frac{n(n+1)}{2}$ combinations to evaluate. Each combination also required a recursive SVM optimisation step. It can be seen that a relatively small subset of features will provide close to optimal results (Figure 5.5). Conversely, with the backward Sequential Feature Selection algorithm, the feature subset is initially equal to the entire d-dimensional feature space and the features that reduce the overall classifier performance the least is removed recursively, until the dimension of the subset = k.

5.2 Selection results

The classification result of the full feature space (n=88) was used as the baseline for comparison with the results of various feature reduction schemes. In Table 5.1 the classification result of each feature is listed in descending order on mean accuracy (MA) and compared to the result obtained by all the features combined. For each measurement, the mean accuracy was calculated using the SVM classifier on the data corpus using five-fold cross-validation. The training and testing data were generated by selecting two independent data sets by random sampling. The metric used for the baseline as well as each subsequent measurement was the mean accuracy achieved with the classifier for each of the cross-validation iterations. This metric was also used to rank all the individual features. The top-ranked feature, based on an individual evaluation, is a Short Time Energy variance, with a mean accuracy of 93.2%. The feature with the lowest individual performance, MFCC 9 skewness, had a mean accuracy of 51.38% on the SVM classifier, which indicates that each of the features in the set contributes positively to the overall result. Although the MFCC 9 skewness feature scored lowest individually, it was among the biggest contributors to the first PCA based on the loadings elements. This is a significant finding, supporting the case for feature selection analysis.

A number of feature selection techniques were employed to determine the optimal feature set. The results for the different selection methods are summarised in Table 5.2. The overall mean accuracy for the full feature set (n=88) was 99.80%. Selecting the features that had an individual mean accuracy higher than 90% (n=5), which comprised the variances of the Short-Time Energy Ratio, MFCC 0 and MFCC 1 as well as the means of the Modified Low Energy Ratio and the Spectral Entropy, yielded a mean accuracy of 96.40%. The ensemble of features with individual mean accuracies higher than 80% (n=19) yielded a combined accuracy of 99.24%, those with individual mean accuracies higher than 70% (n=31) yielded a combined accuracy of 99.80%, and finally, the features with a MA > 60% (n=54) also yielded a combined accuracy of 99.80%. Various sequences of the top N features were then selected as possible candidates. Although all features contribute to a better overall result, the mean accuracy peaked at n=13 features (MA = 99.22%), suggesting that it might be a good trade-off of accuracy and low complexity. The next sequence was the significant loadings of the first five principal components (n=70) which achieved a similar mean accuracy (99.94%) to

the best combination of features in the Univariate (n=44) evaluation. This was followed by the best mean average score for the Random Forest Regression with a mean decrease of Gini impurity index score method (n=25, MA=99.92%) and the permutation importance by a mean decrease of accuracy. This method resulted in similar accuracy (MA=99.80%) and feature count (n=25). The feature ranking with Random Forest Regression method is shown in Figure 5.4. In this test, the features are again ranked in descending order of their score. The variance of the Short-Time Energy feature ranked highest by a significant margin, followed by the mean of the Modified Low Energy Ratio feature. These two features also delivered the highest mean accuracy in the evaluation of individual features, but the difference in the margin between them seems contradictory to the PCA cumulative explained variance ratio which indicated that no single feature dominated the variance in the feature space, but rather that all features contribute positively and suggesting that a combination of features would be optimal, as proven in the final analysis.

The final feature selection method evaluated was the Sequential Feature Selection. The best effort forward Sequential Feature Selection evaluation (n=44) provided slightly better accuracy (MA=99.95%) than the best effort Backward Sequential Feature Selection evaluation (n=59, MA=99.92%) but importantly with fewer features. The Forward Sequential Feature Selection subset (n=44) was selected for the final evaluation of the performance of the MLP NN classifier model. This dataset delivered the best results in the preliminary feature selection tests and apart from delivering a higher mean accuracy (99.95%) than the full feature set, it utilises only half the number of features, i.e. provided a 50% reduction in dimensionality. The feature ranking with the Forward Sequential Feature Selection method is shown in Figure 5.5 and it indicates the use of an increasing number of features based on the recursive addition of features that maximises the score of feature space. The score increases rapidly with the inclusion of only a few features and peaks at a combination of 44 features before slightly decreasing up to the point where all features are included. Figure 5.6 shows the ranking of features using the Backward Sequential Feature Selector method and conversely, how a decreasing number of features influences the selection score. It can be seen that by reducing the feature space, the score initially increases slightly and it only decreases significantly with a small number of features remaining.

An immediate concern with such high accuracy is overfitting of the data. This was repudiated by repeating the test several times and combining k-fold cross-validation with data hold-out to ensure that the testing phase is performed on unseen data.

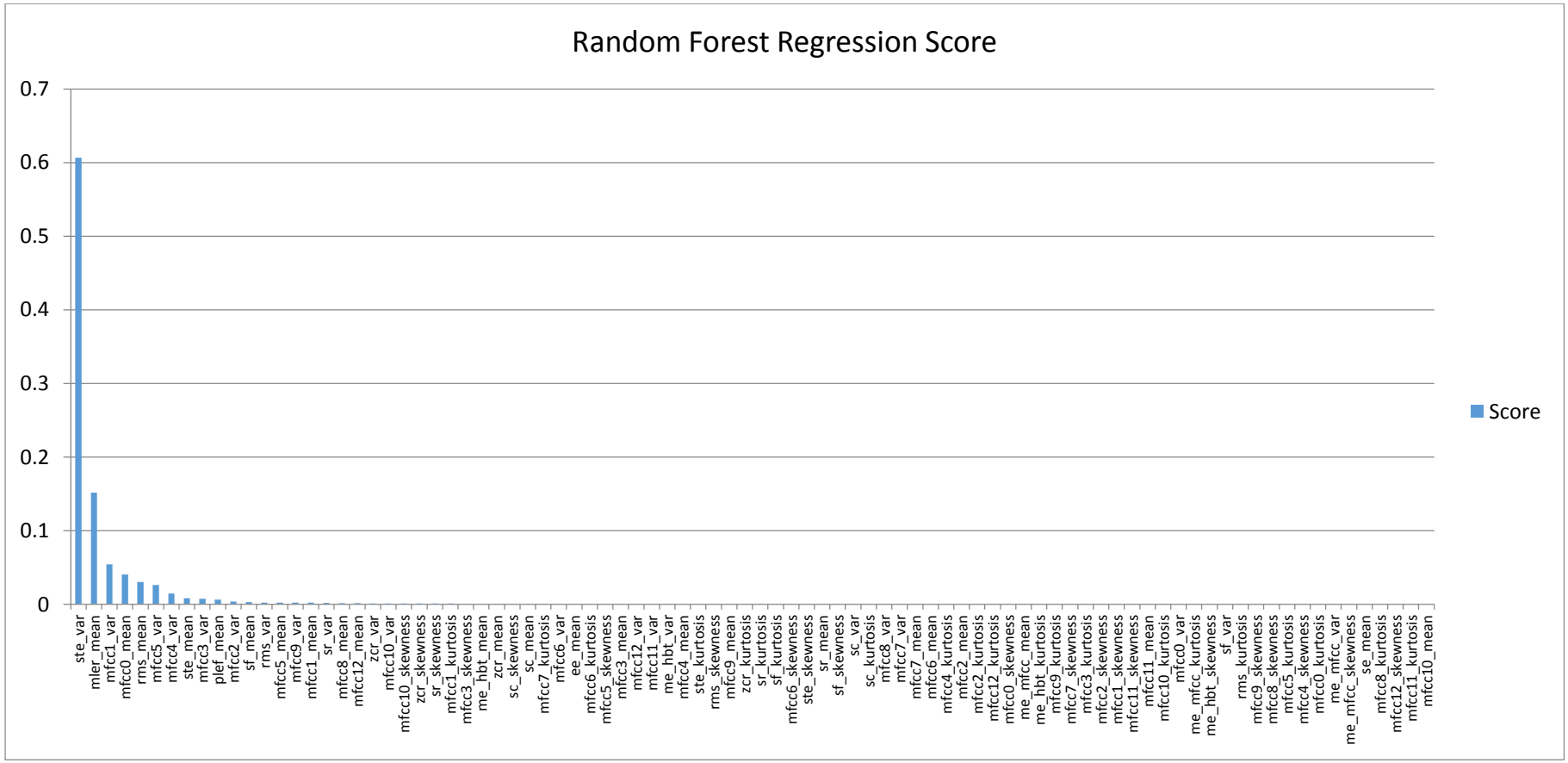

Figure 5.4: Feature ranking with Random Forest Regression method

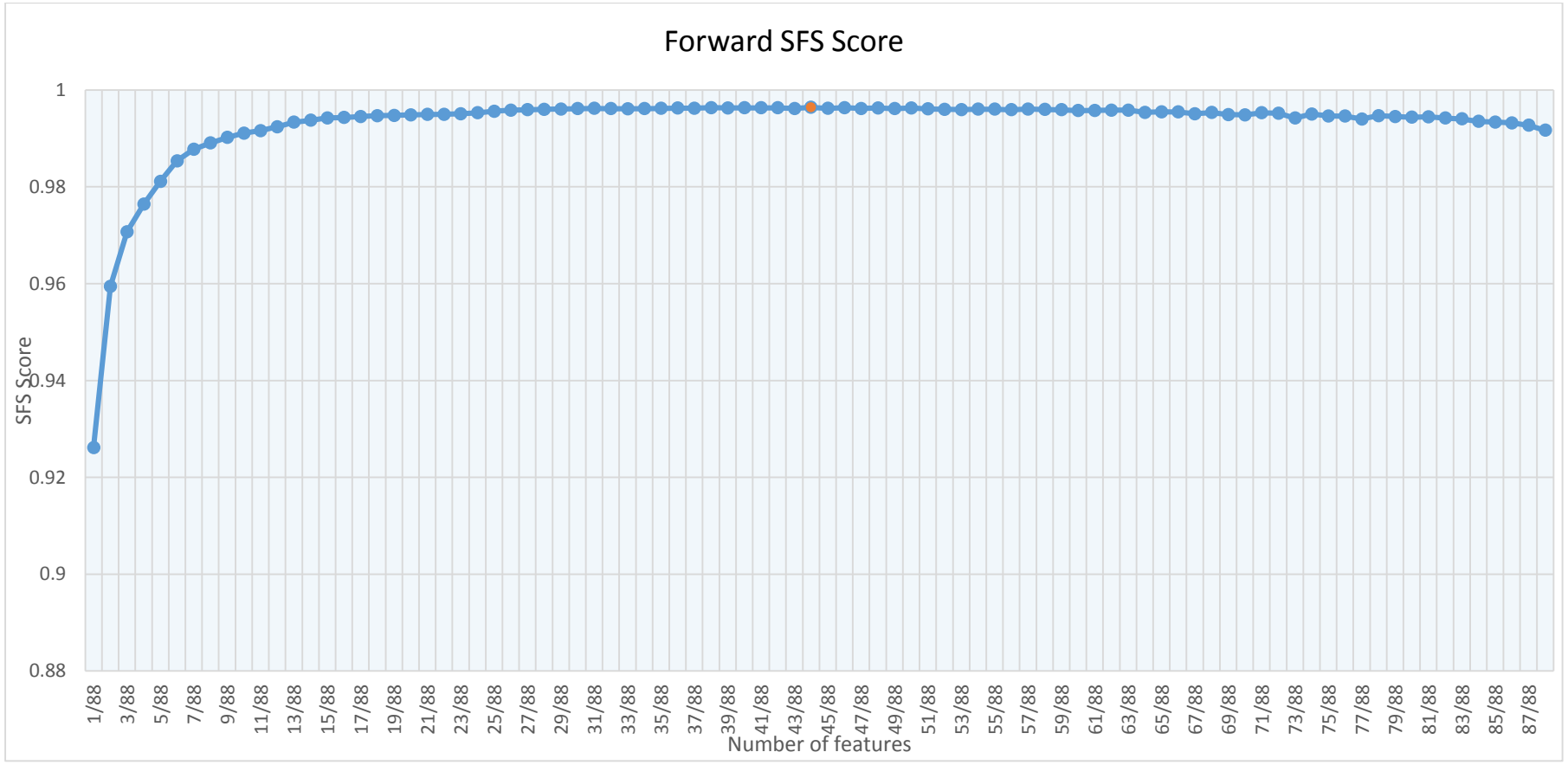

Figure 5.5: Feature ranking with Forward Sequential Feature Selector method

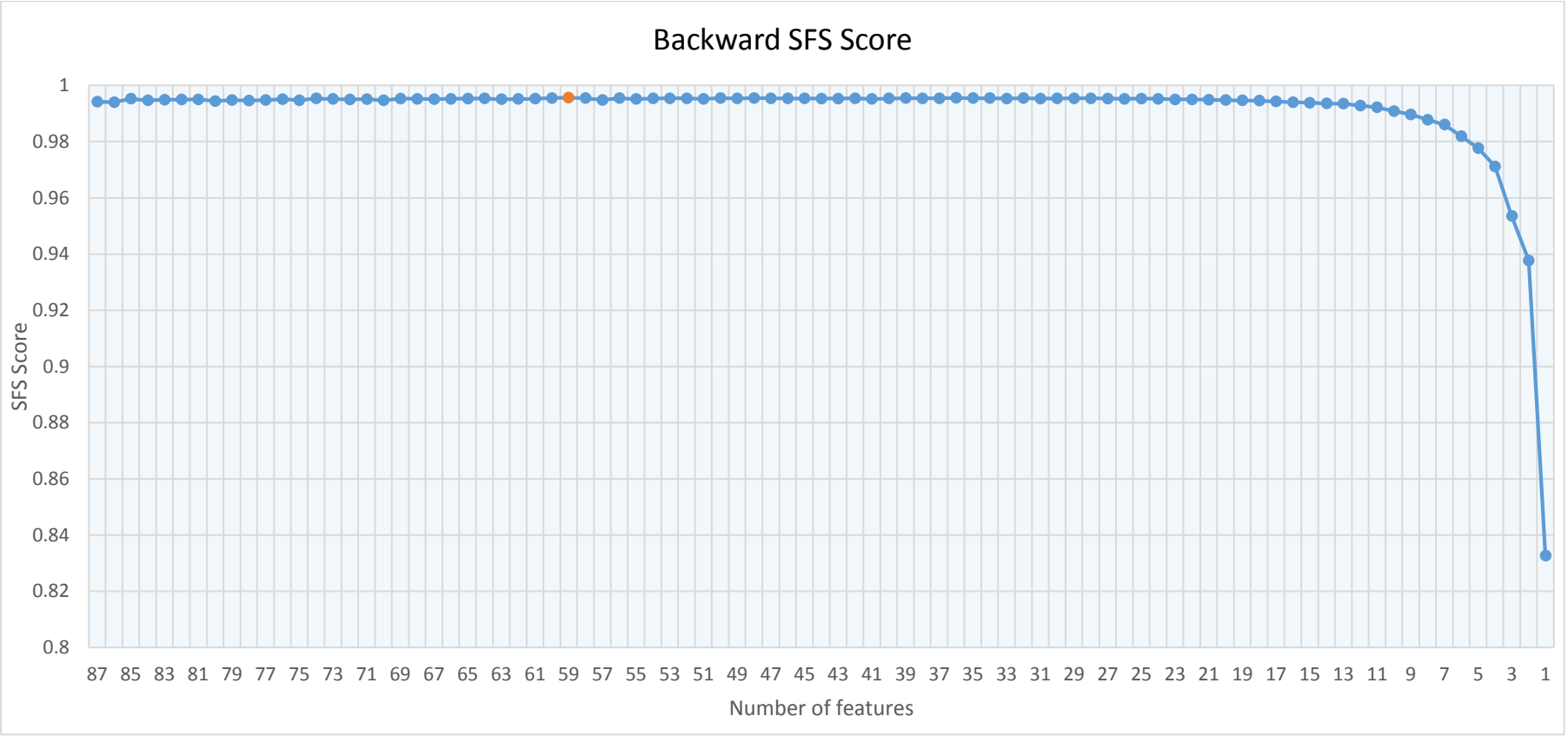

Figure 5.6: Feature ranking with Backward Sequential Feature Selector method

Table 5.1: Classification results for all features combined versus each individual feature.

	Neural Network			Support Vector Machine			
Feature	precision	recall	f1-score	precision	recall	f1-score	mean accuracy
all features	0.99	0.99	0.99	1	1	1	99.80%
ste_var	0.9	0.92	0.9	0.93	0.93	0.93	93.20%
mler_mean	0.92	0.92	0.92	0.92	0.92	0.92	93.04%
mfcc0_var	0.91	0.89	0.9	0.91	0.91	0.91	91.40%
mfcc1_var	0.9	0.91	0.9	0.91	0.91	0.91	90.72%
se_mean	0.9	0.89	0.89	0.9	0.9	0.9	90.46%
mfcc3_var	0.89	0.88	0.88	0.89	0.89	0.89	89.82%
plef_mean	0.88	0.84	0.84	0.91	0.89	0.89	88.92%
mfcc2_var	0.88	0.89	0.89	0.9	0.9	0.89	88.72%
me_hbt_var	0.87	0.87	0.87	0.88	0.88	0.88	88.36%
mfcc5_var	0.86	0.89	0.88	0.88	0.88	0.88	88.00%
mfcc4_var	0.86	0.87	0.86	0.88	0.88	0.88	87.80%
ste_mean	0.85	0.86	0.86	0.87	0.86	0.86	86.70%
mfcc0_mean	0.86	0.85	0.86	0.87	0.86	0.86	85.78%
mfcc6_var	0.85	0.85	0.85	0.86	0.86	0.86	85.60%
sc_var	0.85	0.83	0.82	0.86	0.85	0.85	85.10%
me_mfcc_var	0.84	0.81	0.8	0.86	0.84	0.84	84.64%
sr_var	0.81	0.82	0.8	0.84	0.84	0.84	84.24%
rms_mean	0.83	0.82	0.82	0.85	0.82	0.82	81.38%
zcr_var	0.8	0.79	0.78	0.81	0.8	0.8	80.08%
mfcc7_var	0.79	0.81	0.8	0.8	0.8	0.8	79.48%
me_mfcc_mean	0.78	0.78	0.78	0.79	0.78	0.78	78.92%

sc_mean	0.78	0.77	0.78	0.79	0.79	0.79	78.36%
mfcc7_mean	0.77	0.77	0.77	0.77	0.77	0.77	77.34%
mfcc8_var	0.77	0.78	0.77	0.78	0.77	0.77	77.10%
zcr_skewness	0.76	0.76	0.76	0.76	0.76	0.76	76.94%
me_hbt_kurtosis	0.74	0.76	0.75	0.75	0.75	0.75	75.92%
sr_mean	0.75	0.75	0.75	0.77	0.76	0.76	75.52%
mfcc9_var	0.75	0.74	0.75	0.75	0.75	0.75	75.12%
mfcc5_mean	0.75	0.74	0.75	0.75	0.75	0.75	74.22%
mfcc6_mean	0.74	0.68	0.71	0.73	0.73	0.73	72.78%
me_hbt_skewness	0.72	0.72	0.72	0.71	0.71	0.71	71.10%
sf_var	0.62	0.63	0.62	0.72	0.7	0.69	69.56%
mfcc10_var	0.7	0.69	0.69	0.69	0.69	0.69	69.56%
mfcc1_skewness	0.68	0.68	0.68	0.7	0.69	0.68	69.06%
zcr_kurtosis	0.69	0.7	0.68	0.69	0.68	0.67	67.90%
mfcc11_var	0.66	0.67	0.66	0.67	0.67	0.67	67.26%
me_mfcc_skewness	0.66	0.65	0.66	0.66	0.66	0.66	65.64%
mfcc4_mean	0.65	0.63	0.64	0.68	0.67	0.66	65.48%
mfcc8_mean	0.66	0.66	0.66	0.66	0.66	0.66	65.38%
sc_skewness	0.64	0.64	0.64	0.64	0.64	0.64	65.10%
mfcc10_mean	0.65	0.64	0.64	0.65	0.65	0.65	64.98%
mfcc12_var	0.64	0.63	0.63	0.64	0.64	0.64	63.96%
sf_mean	0.52	0.48	0.49	0.66	0.63	0.61	63.70%
mfcc1_mean	0.62	0.64	0.63	0.64	0.64	0.64	63.70%
ste_skewness	0.62	0.61	0.62	0.62	0.62	0.62	62.98%
sr_skewness	0.59	0.55	0.57	0.64	0.61	0.6	62.22%
rms_skewness	0.62	0.62	0.62	0.62	0.61	0.61	61.60%

mfcc0_skewness	0.61	0.6	0.6	0.61	0.61	0.61	61.48%
mfcc1_kurtosis	0.62	0.6	0.6	0.62	0.61	0.61	61.20%
me_mfcc_kurtosis	0.62	0.62	0.61	0.62	0.62	0.62	60.80%
sf_skewness	0.57	0.69	0.63	0.59	0.59	0.59	60.76%
mfcc11_mean	0.59	0.6	0.6	0.6	0.6	0.59	60.18%
mfcc3_mean	0.59	0.48	0.53	0.62	0.6	0.58	60.10%
sc_kurtosis	0.59	0.6	0.59	0.6	0.6	0.6	60.06%
rms_kurtosis	0.56	0.54	0.55	0.61	0.6	0.58	59.54%
zcr_mean	0.61	0.58	0.59	0.61	0.61	0.61	59.52%
rms_var	0.52	0.52	0.5	0.61	0.59	0.57	59.46%
mfcc12_mean	0.58	0.56	0.57	0.59	0.59	0.58	59.06%
ee_mean	0.59	0.57	0.58	0.6	0.59	0.58	58.70%
me_hbt_mean	0.53	0.47	0.5	0.58	0.57	0.56	58.32%
mfcc4_skewness	0.57	0.58	0.58	0.58	0.58	0.58	58.10%
ste_kurtosis	0.51	0.95	0.66	0.58	0.58	0.58	58.04%
mfcc0_kurtosis	0.53	0.48	0.51	0.58	0.56	0.52	57.42%
mfcc2_kurtosis	0.56	0.57	0.57	0.57	0.56	0.56	56.44%
mfcc5_skewness	0.55	0.56	0.55	0.55	0.55	0.55	54.96%
mfcc3_skewness	0.55	0.55	0.55	0.56	0.55	0.53	54.90%
mfcc2_mean	0.54	0.61	0.56	0.58	0.54	0.48	54.74%
mfcc9_mean	0.53	0.46	0.49	0.53	0.53	0.53	54.68%
sf_kurtosis	0.55	0.55	0.54	0.56	0.55	0.53	54.52%
sr_kurtosis	0.52	0.51	0.51	0.54	0.54	0.54	54.46%
mfcc7_skewness	0.52	0.58	0.55	0.54	0.53	0.48	53.86%
mfcc5_kurtosis	0.52	0.54	0.51	0.54	0.53	0.5	53.82%
mfcc6_skewness	0.5	0.54	0.51	0.54	0.53	0.51	53.68%

mfcc12_skewness	0.52	0.62	0.57	0.54	0.54	0.53	53.60%
mfcc12_kurtosis	0.53	0.53	0.5	0.53	0.53	0.52	53.50%
mfcc11_kurtosis	0.52	0.53	0.51	0.54	0.52	0.46	53.28%
mfcc6_kurtosis	0.25	0.5	0.33	0.52	0.52	0.52	53.16%
mfcc11_skewness	0.52	0.62	0.56	0.54	0.53	0.52	53.14%
mfcc4_kurtosis	0.5	0.55	0.49	0.52	0.52	0.52	53.12%
mfcc8_skewness	0.52	0.48	0.5	0.52	0.52	0.51	53.06%
mfcc2_skewness	0.51	0.5	0.51	0.53	0.53	0.5	52.74%
mfcc3_kurtosis	0.51	0.53	0.42	0.52	0.52	0.49	52.74%
mfcc10_kurtosis	0.53	0.51	0.52	0.53	0.53	0.53	52.72%
mfcc9_kurtosis	0.25	0.39	0.3	0.52	0.52	0.48	52.56%
mfcc8_kurtosis	0.48	0.34	0.33	0.51	0.51	0.51	52.44%
mfcc10_skewness	0.51	0.55	0.52	0.52	0.52	0.51	52.22%
mfcc7_kurtosis	0.54	0.17	0.25	0.55	0.52	0.45	51.50%
mfcc9_skewness	0.5	0.49	0.49	0.25	0.5	0.33	51.38%

Table 5.2: Classification results for top-ranked features combinations

	Neural Network			Support Vector Machine			SVM
Feature	precision	recall	f1-score	precision	recall	f1-score	mean accuracy
All features (n=88)	0.99	0.99	0.99	1	1	1	99.80%
MA > 0.6 (n=54)	0.99	0.99	0.99	1	1	1	99.80%
MA > 0.7 (n=31)	0.99	0.99	0.99	1	1	1	99.80%
MA > 0.8 (n=19)	0.99	0.99	0.99	0.99	0.99	0.99	99.24%
MA > 0.9 (n=5)	0.95	0.96	0.96	0.96	0.96	0.96	96.40%
MA (n= top 6)	0.96	0.97	0.97	0.97	0.97	0.97	97.36%
MA (n=top 7)	0.96	0.98	0.97	0.98	0.98	0.98	97.70%
MA (n=top 8)	0.97	0.97	0.97	0.98	0.98	0.98	97.92%
MA (n=top 9)	0.97	0.97	0.97	0.98	0.98	0.98	97.90%
MA (n=top 10)	0.98	0.98	0.98	0.98	0.98	0.98	98.36%
MA (n=top 11)	0.98	0.98	0.98	0.98	0.98	0.98	98.62%
MA (n=top 12)	0.99	0.99	0.99	0.99	0.99	0.99	99.08%
MA (n=top 13)	0.99	0.98	0.98	0.99	0.99	0.99	99.22%
MA (n=top 14)	0.99	0.99	0.99	0.99	0.99	0.99	99.06%
MA (n=top 15)	0.99	0.98	0.99	0.99	0.99	0.99	99.00%
MA (n=top 16)	0.99	0.99	0.99	0.99	0.99	0.99	99.16%
MA (n=top 17)	0.99	0.99	0.99	0.99	0.99	0.99	99.14%
MA (n=top 18)	0.99	0.99	0.99	0.99	0.99	0.99	99.12%
MA (n=top 19)	0.99	0.99	0.99	0.99	0.99	0.99	99.20%
MA (n=top 20)	0.99	0.99	0.99	0.99	0.99	0.99	99.24%

PCA Loadings 1-5	0.99	0.99	0.99	1	1	1	99.94%
RFR impurity (n=25)	0.99	0.99	0.99	1	1	1	99.92%
RFR accuracy (n=25)	0.99	0.99	0.99	1	1	1	99.80%
Univariate Top 50% (n=44)	0.99	0.99	0.99	0.99	0.99	0.99	99.94%
Forward SFS Top 25	0.99	0.99	0.99	1	1	1	99.88%
Forward SFS Best MA (n=44)	0.99	0.99	0.99	1	1	1	99.95%
Backward SFS Top 25	0.99	0.99	0.99	1	1	1	99.92%
Backward SFS Best MA (n=59)	0.99	0.99	0.99	1	1	1	99.92%

A number of feature selection techniques were employed to determine the optimal feature set as discussed in chapter 5. Each feature was evaluated against both the MLP neural network classifier and the SVM classifier. The training and testing data was generated by selecting two independent data sets by random sampling. The mean accuracy for the SVM classifier was used to rank the features. It was expected that the MLP neural network would be at a disadvantage when evaluating only one feature at a time since the hidden layers and layer width remained optimised for a high dimensional feature set. The result, however, was that the MLP neural network performed well and achieved similar results to the SVM classifier. The combined feature set delivers excellent results, with a total mean accuracy of 99.8%. In Table 5.2 the results of the Univariate selection process, Random Forrest Regression (RFR), forward and backward Sequential Feature Selection (SFS) processes are shown, compared to combinations of top-n individual features. The Forward Sequential Feature Selection subset (n=44) was selected for the final evaluation, as it had a higher mean accuracy (99.95%) than the full feature set, it utilises only half the number of features. This leads to a reduction in the optimisation task for the MLP Neural Network.

The subset consisted of the following collection of features for the final evaluation:

- **RMS Mean,**
- **ZCR Mean, ZCR Variance, ZCR Skewness, ZCR Kurtosis,**
- **STE Variance, STE Skewness,**
- **PLEF Mean,**
- **MLER Mean,**
- **SR Mean,**
- **SF Mean,**
- **SC Mean, SC Kurtosis,**
- **ME MFCC Mean,**
- **ME_Hilbert Variance, ME Hilbert Skewness,**
- **MFCC_0 Mean, MFCC_0 Variance, MFCC_0 Skewness, MFCC_0 Kurtosis,**
- **MFCC_1 Mean, MFCC_1 Variance,**
- **MFCC_2 Mean, MFCC_2 Variance,**

- **MFCC_3 Variance,**
- **MFCC_4 Variance, MFCC_4 Skewness, MFCC_4 Kurtosis,**
- **MFCC_5 Mean, MFCC_5 Variance,**
- **MFCC_6 Mean, MFCC_6 Variance, MFCC_6 Skewness,**
- **MFCC_7 Mean, MFCC_7 Kurtosis,**
- **MFCC_8 Skewness,**
- **MFCC_9 Mean,**
- **MFCC_10 Mean, MFCC_10 Variance, MFCC_10 Skewness, MFCC_10 Kurtosis,**
- **MFCC_11 Variance,**
- **MFCC_12 Mean, MFCC_12 Variance**

It is significant that three of the features in this set, MFCC_7 Kurtosis, MFCC_10 Skewness, MFCC_10 Kurtosis were among the six lowest-performing features on an individual basis, barely achieving over 50 % mean accuracy.

The designated feature set was evaluated with the MLP Neural Network classifier using 5-fold cross-validation, which avoids overlapping test sets. The Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm was used to perform the back-propagation optimisation of the MLP parameters. BFGS is a quasi-Newton method algorithm for iteratively solving nonlinear optimisations. The BFGS algorithm ($O(n^2)$) is computationally less complex than the Newton method ($O(n^3)$) and copes well with a large number of variables. The $O(n^2)$ complexity is still exorbitantly high if the sample size becomes large and therefore the training size for each of the folds was limited to 5000 samples (approximately 10%) and the remaining samples were assigned to the test set. The classification results of each run were aggregated and the performance of the classification results is summarised in the confusion matrix in Figure 5.7. The speech class is denoted “0” or “Negative” and the music class as “1” or “positive”. The confusion matrix consists of four basic characteristics that are used to define the measurement metrics of the classifier. These four characteristics are:

1. TN (True Negative): TN represents the number of correctly classified speech audio segments.

2. TP (True Positive): TP represents the number of audio segments that have been properly classified as music.
3. FP (False Positive): FP represents the number of speech audio segments misclassified as music. FP is also known as a Type I error.
4. FN (False Negative): FN represents the number of music audio segments misclassified as speech. FN is also known as a Type II error.

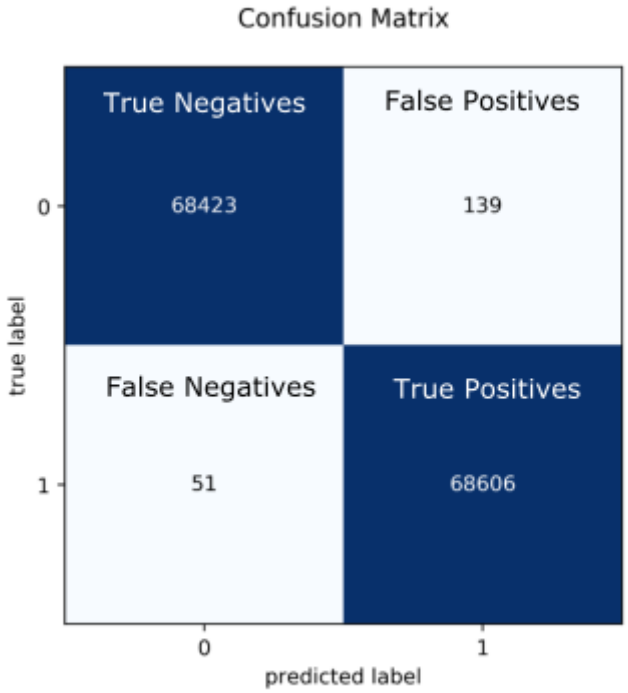

Figure 5.7: Confusion Matrix for MLP classifier

5.3 Validating the results of the classification model

Building a predictive model and using accuracy as the only metric of performance does not guarantee that it is a good model. A model must also be reproducible and robust. The purpose of validating the model is to instil trust in the result and confidence to use the model as part of a production system.

The model was evaluated using stratified K-fold cross-validation. Stratification aims to ensure that each fold is representative of all strata of the data. This aim is achieved by ensuring that the data corpus was balanced, thus representing each major class equally and sampling the testing and training data randomly based on a normal distribution of samples. 5-fold cross-validation was used in which the model is evaluated 5 times each time using a randomly selected test set consisting of approximately 18% of the samples in the total corpus. The performances results across the 5 iterations are then averaged to obtain the final estimate of the model performance.

The confusion matrix presented in Figure 5.7 compares the real training data and the mean prediction results of the classifier model over the 5-folds of the cross-validation and the statistical metrics are summarised in Table 5.3.

Table 5.3: **Evaluation Results**

True Positive (TP):	68423
False Positive (FP):	51
True Negative (TN):	68606
False Negative (FN):	139
Accuracy:	0.9986153521013854
Error:	0.0013846478986146234
Precision:	0.9992551917516137
Recall:	0.9979726379043785
F1 Score:	0.9986135030211039

The statistical metrics highlighted are accuracy, precision, recall, and F1 score, which are calculated based on the TP, TN, FP, and FN stated in the confusion matrix (Figure 5.7)

The Accuracy metric of an algorithm(**5.1**) is represented as the ratio of correctly classified audio segments (TP+TN) to the total number of audio segments (TP+TN+FP+FN).

$$Accuracy = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN})} \tag{5.1}$$

The Precision metric of an algorithm (**5.2**) is represented as the ratio of correctly classified audio segments (TP) to the total audio segments predicted to be of a certain class (TP+FP).

$$Precision = \frac{(\mathrm{TP})}{(\mathrm{TP} + \mathrm{FP})} \tag{5.2}$$

The Recall metric (**5.3**) is defined as the ratio of correctly classified audio segments (TP) divided by the total number of audio segments of a certain class.

$$Recall = \frac{(\mathrm{TP})}{(\mathrm{TP} + \mathrm{FN})} \tag{5.3}$$

The F1 score (**5.4**) is also known as the F Measure. The F1 score states the equilibrium between the precision and the recall.

$$F1\ Score = \frac{2 * \mathrm{precision} * \mathrm{recall}}{\mathrm{precision} + \ \mathrm{recall}} \tag{5.4}$$

In this study, the classification model had an overall accuracy of 99.86 % with an equally impressive precision (99.93%) and recall (99.79%).

With such high numbers, it is important to check if overfitting of the classifier did not occur. The way to check this is by plotting the training error versus the test error. Looking at the cost function error graph in Figure 5.8, it can be observed that the model produces similar results for both the training set and the test set and that the cost function error for the test set does not start to diverge from that of the training set. The latter is indicative of overfitting when the model is trained recursively. It can be inferred that it is unlikely that overfitting occurs in the model.

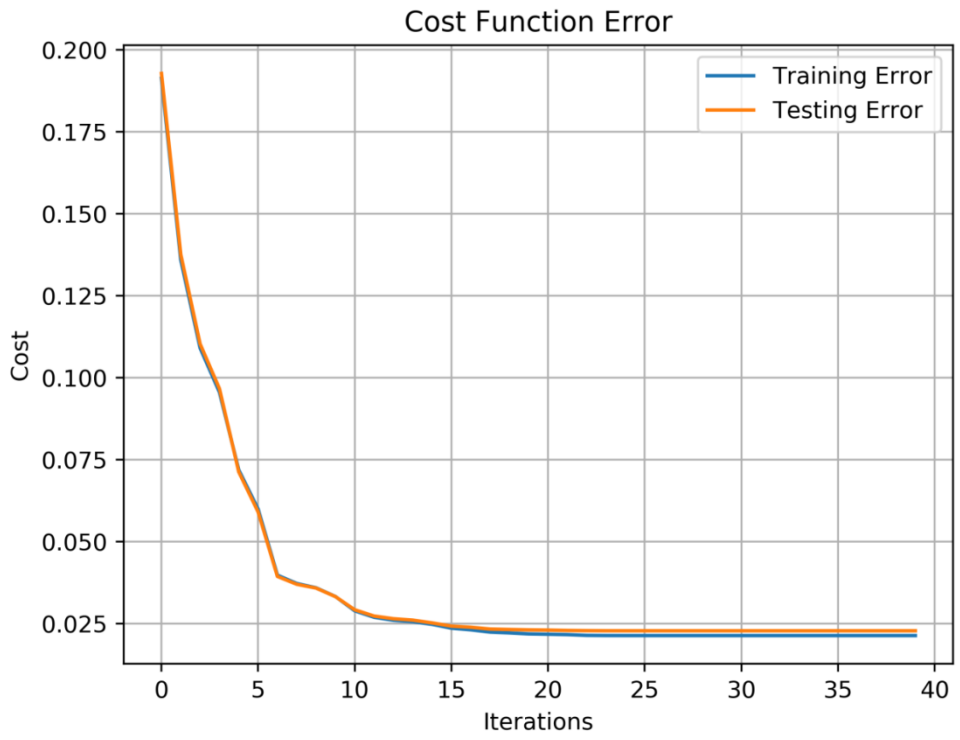

Figure 5.8: Training vs Testing cost function error

Chapter 6 - Conclusion

6.1 Summary

This research dealt with the problem of speech / non-speech discrimination and the development of a feed-forward neural network to classify an audio signal into speech and non-speech parts. In Chapter 1, the background information that leads up to this research was explained, along with the purpose and methodology of this research. The research objectives were stated in section 1.3. A literature review was presented in chapter 2, where related work was discussed and background theory offered to place this study in the context of literature. In section 2.2 audio theory, in general, was discussed, followed by a description of the process of audio content analysis in section 2.3, including descriptions of the various audio features that was evaluated in this study. Section 2.4 provided a theoretical description of the audio features used in this research followed by section 2.5 with a summary of previous research in speech/music discrimination.

Section 2.6 and 2.7 introduced the theoretical aspects of the classification process, commencing with a discussion of decision and learning paradigms and ending with an overview of classification models used in previous and related publications. Chapter 3 builds on this topic to describe the classification model used in this research, with a specific focus on Support Vector Machine and Artificial Neural Network algorithms. The chapter was concluded with a brief discussion of the mathematical optimisation process required to update and tune the Neural Network Model's parameters.

Chapter 4 described the compilation of the dataset for this study in section 4.2 and the development of the algorithm in section 4.3. Sections 4.4 to 4.6 provide selected extracts of code to show the function of important parts of the algorithm. Chapter 5 presents a comprehensive description of the evaluation and selection of audio features for application in speech-music discrimination algorithms. This process has been distinctly lacking in previous studies. In section 5.2, the experimental evaluation of the algorithm was presented, followed by a validation of the classification model in 5.3.

These chapters, particularly chapters 4 and 5 set out to solve the research objectives as stated in section 1.3 Firstly, various low-level statistical audio features were evaluated in section 5.1 based on the findings in previous research as explored in the literature review conducted in Chapter 2. A wide range of audio features used in previous research was identified as well as the mathematical methods required to extract these features from an audio signal. Only two previous studies attempted to compare audio features to determine suitability to the task at hand (Scheirer and Slaney, 1997), (Carey, Parris and Lloyd-Thomas, 1999). These two studies compared the performance of the individual features and not combinations of features. It was found that that many previous studies used audio features without elaborating or substantiating the rationale for using them other than the simplicity of usage or using mathematically similar features.

This emphasised the importance of the second research objective, namely to determine the suitability of the selected features to accurately and efficiently discriminate speech and non-speech components of broadcast audio streams. Several feature selection methods were employed and discussed in Chapter 5 to evaluate the list of features selected in the first objective individually and in combinations to determine suitability and efficiency in discriminating different audio classes. An optimal subset of features was selected to be selected and used in a classifier model. The first four standardised statistical moments of twelve audio features were evaluated, namely the mean, variance, skewness and kurtosis of the Root Mean Square value, Short Time Energy Ratio, Zero Crossing Rate, Spectral Rolloff, Spectral Flux, Spectral Centroid, Energy Entropy, Spectral Entropy, the first 13 Mel Frequency Cepstral Coefficients (MFCC), Percentage Low Energy Frames, Modified Low Energy Ratio and 4Hz Modulation Energy. The 4 Hz modulation energy feature was computed by two different methods, firstly as a by-product of the MFCC feature and secondly utilising the Hilbert transform for envelope detection. This resulted in an 88-dimensional feature space.

Principle component analysis was used to analyse the contribution of each feature and from the cumulative explained variance ratio (Figure 5.1) it was ascertained that all of the features contributed meaningfully to the overall result and that the process was not dominated by only a small number of significant features. It was also indicated that gains of cumulative explained

variance ratio diminished above 50% for the total feature space, again emphasising the need for a thorough selection process. Combinations of individual features (Table 5.1), along with the features contained in the loadings elements of the first five principle components were compared with results of the Univariate selection process, Random Forrest Regression (RFR), forward and backward Sequential Feature Selection (SFS) processes. The results are summarised in Table 5.2. The Forward Sequential Feature Selection subset (n=44) was selected as it delivered excellent mean accuracy (99.94%) with a 50% reduction in feature dimensionality.

The development of a multilayer perceptron neural network classifier model was described in Chapter 4 as proposed in the third research objective. The classification model had an overall accuracy of 99.86 % with an equally impressive precision (99.93%) and recall (99.79%). This compared very well the Support Vector Machine classifier model which was primarily used in the feature selection process. The Support Vector Machine classifier was faster and slightly more accurate, yet the MLP neural network is more robust and scalable. By optimising feature set selection, excellent results were achieved and the difference between classifier algorithms became negligible.

These chapters described the reseach process, analysis and outcomes and confirm that the objectives set out in Section 1.3 was achieved and thus to the overall aim of this study, were accomplished.

6.2 Limitations and recommendations

Despite the good results obtained in this study, some limitations need to be considered. Firstly, only two audio classes, pure speech and music were evaluated. Although this is pertinent to curated audio databases, it does not represent the nature of sound that we normally encounter in real life. Audio signals are typically a mixture of speech, music and background ambient noise. Adapting the classification algorithm to a multi-class model is completely within the potential of the MLP neural network and the biggest change required, is the use of a multi-class activation function like the

softmax function. Secondly, the number of layers and perceptions in the MLP neural network was not specifically optimised but rather chosen based on limited empirical experiments. The MLP neural network performs very well with the parameters selected, yet it could be beneficial to conduct optimisation of these parameters as this may well increase performance and/or decrease complexity. Another limitation is that the classification algorithm operates on off-line audio data. The classification algorithm would ideally be used for real-time classification. This could be achieved by integrating the part of the algorithm that extracts the audio features and the MLP neural network part with asynchronous multi-processing queues in Python.

6.3 Future research

Besides the recommendations discussed in the previous section, some other research topics opportunities emanated during this study. The MLP neural network algorithm described in this study was developed and tested on standard computer hardware. It would be a natural progression to apply this algorithm to miniaturised portable hardware and specifically Internet of Things (IoT) devices for applications such as security surveillance and real-time speech recognition and Natural Language Processing (NLP).

6.4 Conclusion

This research described the development of a speech and music discrimination algorithm using a simple feed-forward neural network classifier that can be used as a building block, enabling an automatic process of discriminating between speech and non-speech data for building speech libraries and ultimately as a tool that can assist research into the processing of local languages. It was demonstrated that with thorough feature selection processes, a mean accuracy of 99.86% and 50% reduction in dimensionality was achieved, which results in a significant reduction in processing effort required. This is particularly important in embedded processor applications where processing power is often minimal or inadequate and energy must be conserved. The salient conclusion from this study is that careful evaluation and selection of features yield significant accuracy and performance gains.

References

Ajmera, J., McCowan, I. and Bourlard, H. (2003) 'Speech/music segmentation using entropy and dynamism features in a HMM classification framework', *Speech Communication*, 40(3), pp. 351–363. doi: 10.1016/S0167-6393(02)00087-0.

Alexandre, E. *et al.* (2006) 'Automatic sound classification for improving speech intelligibility in hearing aids using a layered structure', in *International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 2006*. Springer, Berlin, Heidelberg, pp. 306–313. doi: 10.1007/11875581_37.

Alexandre, E. *et al.* (2008) 'Speech/music/noise classification in hearing aids using a two-layer classification system with MSE linear discriminants', in *16th European Signal Processing Conference (EUSIPCO 2008)*. Luasanne, pp. 1–5.

Bachu, R. G. *et al.* (2010) 'Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal', in *Advanced Techniques in Computing Sciences and Software Engineering*. Springer, pp. 279–282. doi: 10.1007/978-90-481-3660-5_47.

Balabko, P. (1999) *Speech and music discrimination based on signal modulation spectrum*, *Research Report*. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Speech+and+Music+Discrimination+based+on+Signal+Modulation+Spectrum.#0.

Barbedo, J. G. A. and Lopes, A. (2006) 'Automatic Genre Classification of Musical Signals', EURASIP J. Adv. Signal Process. 2007, 064960. https://doi.org/10.1155/2007/64960

Behnel, S. *et al.* (2011) 'Cython: The Best of Both Worlds', *Computing in Science Engineering*, 13(2), pp. 31–39. doi: 10.1109/MCSE.2010.118.

Beierholm, T. and Baggenstoss, P. M. (2004) 'Speech music discrimination using class-specific features', in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pp. 379-382 Vol.2. doi: 10.1109/ICPR.2004.1334226.

Bishop, C. M. (2006) *Pattern Recognition and Machine Learning*, *Pattern Recognition and Machine Learning*. Springer. doi: 10.1117/1.2819119.

Bugatti, A., Flammini, A. and Migliorati, P. (2002) 'Audio Classification in Speech and Music: A Comparison between a Statistical and a Neural Approach', *EURASIP Journal on Advances in Signal Processing*, 2002, pp. 372–378. doi: 10.1155/S1110865702000720.

Bugatti, A., Leonardi, R. and Rossi, L. (1999) 'A Video Indexing Approach Based on Audio Classification.', in *International Workshop on Very Low Bit Rate Video (VLBV)*, pp. 75–78.

Burred, J. J. and Lerch, A. (2004) 'Hierarchical automatic audio signal classification', *AES: Journal of*

the Audio Engineering Society, 52(7–8), pp. 724–739.

Carey, M. J., Parris, E. S. and Lloyd-Thomas, H. (1999) 'A comparison of features for speech, music discrimination, IEEE Internat', *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 2445–2448. doi: 10.1109/ICASSP.1999.758084.

Cass, S. (2018) 'The 2018 top programming languages', *IEEE Spectrum*, 31, p. 1.

Dudley, H. (1940) 'The Carrier Nature of Speech', *Bell System Technical Journal*, 19, pp. 495–515. doi: 10.1002/j.1538-7305.1940.tb00843.x.

Edwards, E. and Chang, E. F. (2013) 'Syllabic (∼2-5 Hz) and fluctuation (∼1-10 Hz) ranges in speech and auditory processing', *Hearing Research*, 305, pp. 113–134. doi: 10.1016/j.heares.2013.08.017.

El-Maleh, K. *et al.* (2000) 'Speech/music discrimination for multimedia applications', in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. doi: 10.1109/ICASSP.2000.859336.

Ericsson, L. (2009) *Automatic speech / music discrimination in audio files*. Royal Institute of Technology, Sweden.

Foote, J. T. (1997) 'Content-based retrieval of music and audio', in *Proceedings of Photo-Optical Instrumentation Engineers (SPIE)*, pp. 138–147. doi: 10.1117/12.290336.

Gallardo-Antolin, A. and Montero, J. M. (2010) 'Histogram equalization-based features for speech, music, and song discrimination', *IEEE Signal Processing Letters*, 17(7), pp. 659–662. doi: 10.1109/LSP.2010.2049877.

Giannakopoulos, T. (2015) 'PyAudioAnalysis: An open-source python library for audio signal analysis', *PLoS ONE*, 10(12). doi: 10.1371/journal.pone.0144610.

Giannakopoulos, T., Pikrakis, A. and Theodoridis, S. (2006) 'A speech/music discriminator for radio recordings using bayesian networks', in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. doi: 10.1109/ICASSP.2006.1661399.

Graves, A., Mohamed, A. and Hinton, G. (2013) 'Speech recognition with deep recurrent neural networks', *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, (3), pp. 6645–6649. doi: 10.1109/ICASSP.2013.6638947.

Hepper, P. G. and Shahidullah, B. S. (1994) 'Development of fetal hearing', *Archives of Disease in Childhood*, 71, pp. F81–F87. doi: 10.1136/fn.71.2.f81.

Khan, M. K. S. and Al-Khatib, W. G. (2006) 'Machine-learning based classification of speech and music', *Multimedia Systems*, 12(1), pp. 55–67. doi: 10.1007/s00530-006-0034-0.

Korvas, M. *et al.* (2014) 'Free English and Czech telephone speech corpus shared under the CC-BY-SA 3.0 license', in *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*. Available at: http://www.openslr.org/resources.php.

Lavner, Y. and Ruinskiy, D. (2009) 'A Decision-Tree-Based Algorithm for Speech/Music Classification and Segmentation', *EURASIP Journal on Advances in Signal Processing*, (239892), pp. 1–14. doi: 10.1155/2009/239892.

Lerch, A. (2012) *An introduction to audio content analysis: Applications in signal processing and music informatics*, *An Introduction to Audio Content Analysis: Applications in Signal Processing and Music Informatics*. John Wiley & Sons, Inc. doi: 10.1002/9781118393550.

Liu, Z., Wang, Y. and Chen, T. (1998) 'Audio Feature Extraction and Analysis for Scene Segmentation and Classification', *The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology*, 20, pp. 61–79. doi: 10.1023/A:1008066223044.

Lobo, A. P. and Loizou, P. C. (2003) 'Voiced / Unvoiced Speech Discrimination In Noise Using Gabor Atomic Decomposition', in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Hong Kong. doi: 10.1109/ICASSP.2003.1198907.

Lu, L., Jiang, H. and Zhang, H. (2001) 'A robust audio classification and segmentation method', in *Proceedings of the ninth ACM international conference on Multimedia - MULTIMEDIA '01*, pp. 203–211. doi: 10.1145/500141.500173.

Lu, L., Zhang, H.-J. and Li, S. Z. (2003) 'Content-based audio classification and segmentation by using support vector machines', *Multimedia Systems*, 8(6), pp. 482–492. doi: 10.1007/s00530-002-0065-0.

Lu, L., Zhang, H. J. and Jiang, H. (2002) 'Content analysis for audio classification and segmentation', *IEEE Transactions on Speech and Audio Processing*, 10(7), pp. 504–516. doi: 10.1109/TSA.2002.804546.

Markaki, M. and Stylianou, Y. (2011) 'Discrimination of speech from nonspeeech in broadcast news based on modulation frequency features', *Speech Communication*. Elsevier B.V., 53(5), pp. 726–735. doi: 10.1016/j.specom.2010.08.007.

Matsuzawa, T. (2007) 'Comparative cognitive development', *Developmental Science*, 10(1), pp. 97–103. doi: 10.1111/j.1467-7687.2007.00570.x.

Van der Merwe, H. (1996) *The research process: Problem statement and research design*, *Effective research in the human sciences – Research management for researchers, supervisors and master's and doctoral candidates*. Edited by J. G. Garbers. J.L.Van Schaik.

Muñoz-Expósito, J. E. *et al.* (2009) 'Speech/music discrimination based on warping transformation and fuzzy logic for intelligent audio coding', *Applied Artificial Intelligence*, 23(5), pp. 427–442. doi: 10.1080/08839510902872306.

Paliwal, K. and Wojcicki, K. (2008) 'Effect of analysis window duration on speech intelligibility', *IEEE Signal Processing Letters*, 15, pp. 785–788. doi: 10.1109/LSP.2008.2005755.

Panagiotakis, C. and Tziritas, G. (2005) 'A speech/music discriminator based on RMS and zero-crossings', *IEEE Transactions on Multimedia*, 7(1), pp. 155–166. doi: 10.1109/TMM.2004.840604.

Pedregosa, F., Weiss, R. and Brucher, M. (2011) 'Scikit-learn : Machine Learning in Python', *Journal of Machine Learning Research*, 12, pp. 2825–2830.

Pfeiffer, S., Fischer, S. and Effelsberg, W. (1997) 'Automatic audio content analysis', *Proceedings of the 4th ACM International Conference on Multimedia, MULTIMEDIA 1996*, pp. 21–30. doi: 10.1145/244130.244139.

Pikrakis, A., Giannakopoulos, T. and Theodoridis, S. (2006) 'Speech / Music Discrimination For Radio Broadcasts Using A Hybrid HMM-Bayesian Network Architecture', in *14th European Signal Processing Conference (EUSIPCO)*. Florence, pp. 2–6.

Pinquier, J., Rouas, J. L. and E-Obrecht, R. a. (2002) 'Robust speech/music classification in audio documents', in *7th International Conference on Spoken Language Processing, ICSLP2002 - INTERSPEECH 2002*.

Reeder, H. (2001) *Speech Music Discrimination*, *University of Stellenbosch, Stellenbosch*.

Rothmann, D. (2018) *Human-Like Machine Hearing With AI*, *towardsdatascience.com*. Available at: https://towardsdatascience.com/human-like-machine-hearing-with-ai-1-3-a5713af6e2f8.

Saad, E. M. *et al.* (2002) 'A multifeature speech/music discrimination system', in *Proceedings of the Nineteenth National Radio Science Conference*. Alexandria, Egypt, pp. 208–213. doi: 10.1109/NRSC.2002.1022623.

Saunders, J. (1996) 'Real-time discrimination of broadcast speech/music', *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, 2, pp. 993–996 vol. 2. doi: 10.1109/ICASSP.1996.543290.

Scheirer, E. D. (2000) *Music-Listening Systems*. Massachusetts Institute of Technology.

Scheirer, E. and Slaney, M. (1997) 'Construction and Evaluation of a Robust Multi-feature Speech Music Discriminator', in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Munich, Germany: IEEE, pp. 1331–1334. doi: 10.1109/ICASSP.1997.596192.

Sigtia, S. *et al.* (2016) 'Automatic Environmental Sound Recognition: Performance Versus Computational Cost', *IEEE/ACM Transactions on Audio Speech and Language Processing*, 24(11), pp. 2096–2107. doi: 10.1109/TASLP.2016.2592698.

Taniguchi, T., Tohyama, M. and Shirai, K. (2008) 'Detection of speech and music based on spectral tracking', *Speech Communication*, 50(7), pp. 547–563. doi: 10.1016/j.specom.2008.03.007.

Tardón, L. J., Sammartino, S. and Barbancho, I. (2010) 'Design of an efficient music-speech discriminator.', *The Journal of the Acoustical Society of America*, 127(1), pp. 271–9. doi: 10.1121/1.3257204.

Traunmüller, H. and Eriksson, A. (1994) *The frequency range of the voice fundamental in the speech of male and female adults*, *Research Report*. Stockholm. Available at:

https://www2.ling.su.se/staff/hartmut/f0_m&f.pdf.

Tzanetakis, G. and Cook, P. (2002) 'Musical genre classification of audio signals', *IEEE Transactions on Speech and Audio Processing*, 10(5), pp. 293–302. doi: 10.1109/TSA.2002.800560.

Vandenberghe, L. (2019) *Optimization Methods for Large-Scale Systems*, *Lecture Notes ECE236C UCLA*. Available at: http://www.seas.ucla.edu/~vandenbe/ee236c.html.

Varnet, L. *et al.* (2017) 'A cross-linguistic study of speech modulation spectra', *The Journal of the Acoustical Society of America*, 142(4), pp. 1976–1989. doi: 10.1121/1.5006179.

De Villiers, M. R. (2012) 'Models for interpretive information systems research, part 1: Is research, action research, grounded theory - a meta-study and examples', in *Research Methodologies, Innovations and Philosophies in Software Systems Engineering and Information Systems*. IGI Global, pp. 222–237. doi: 10.4018/978-1-4666-0179-6.ch011.

Voskoglou, C. (2017) *What is the best programming language for Machine Learning?*, *towardsdatascience.com*. Available at: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7 (Accessed: 26 February 2022).

Wang, W. Q., Gao, W. and Ying, D. W. (2003) 'A fast and robust speech/music discrimination approach', in *Fourth International Conference on Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia Processing and 4th Pacific-Rim Conference on Multimedia*. Singapore, pp. 1325–1329. doi: 10.1109/ICICS.2003.1292679.

Williams, G. and Ellis, D. P. W. (1999) 'Speech/music discrimination based on posterior probability features', in *Sixth European Conference on Speech Communication and Technology (Eurospeech '99)*. Budapest, pp. 687–690. doi: 10.2307/202051.

Wu Chou and Liang Gu (2001) 'Robust singing detection in speech/music discriminator design', in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. IEEE International Conference - Volume 02 (ICASSP '01), pp. 865–868. doi: 10.1109/ICASSP.2001.941052.

Zhang, T. and Jay Kuo, C. C. (2001) 'Audio content analysis for online audiovisual data segmentation and classification', *IEEE Transactions on Speech and Audio Processing*, 9(4), pp. 441–457. doi: 10.1109/89.917689.

Appendix A – Summary of previous studies

Paper	**Main Applications**	**Features**	**Classification method**	**Audio material**	**Results**	**Notes:**
(Ajmera, McCowan and Bourlard, 2003) *Speech/music segmentation using entropy and dynamism features in an HMM classification framework.*	Automatic transcription of broadcast news	Entropy measure and "dynamism" estimated at the output of a multilayer perceptron (MLP) trained to emit posterior probabilities of phones. MLP input: 13 first cepstra of a 12th-order perceptual linear prediction filter.	2-state HMM with minimum duration constraints (threshold-free, unsupervised, no training).	4 files (10 min. each): alternate segments of speech and music, speech/music interleaved	GMM: Speech 98.8%, Music 93.9%. Alternating, variable-length segments (MLP): Speech 98.6%, Music 94.6%.	
(Alexandre et al., 2006) *Speech/music segmentation using entropy and dynamism features in an HMM classification framework.*	Speech/music classification algorithm for hearing aids.	Spectral centroid, spectral roll-off, spectral flux, ZCR, High Zero Crossing Rate Ratio, short-time energy, low short time energy ratio (LSTER), MFCC, voice-to- white, activity level, loudness, Spectral Flatness Measure (SFM)	Fisher linear discriminant, three-layer MLP neural network	Total of 2936 files of 2.5 seconds each (122min 33sec), Sampled at 22050 Hz with 16 bits per sample. Classes: speech, speech in stationary noise, speech in non-stationary noise, stationary noise and non-stationary noise (training: 45 min, validation: 17 min and testing: 60min)	Music 99.1%, speech 96.6%. Individual features: 95.9% (MFCC), 95.1% (voice to white).	

(Alexandre et al., 2008) *Speech/music/noise classification in hearing aids using a two-layer classification system with MSE linear discriminants.*	Development of an automatic sound classifier for digital hearing aids with automatic self-adaptation.	14 Features: Spectral Centroid, Spectral Roll-Off, Voice2White, Spectral Flux, Zero Crossing Rate, Short-Time Energy, Percentage of Low Energy Frames, High Zero Crossing Rate Ratio, Low Short-Time Energy Rate, Spectral Flatness Measure, Mel Frequency Cepstral Coefficients, Loudness, Spectral Crest Factor, Bandwidth	Two-stage Mean Squared Error (MSE) linear discriminant classifier, first one discriminates the input sound into either speech or non-speech, and the second layer classifies it into either speech in quiet or speech in noise.	Total of 2936 files, with a length of 2.5 seconds each. The sampling frequency was 22050 Hz with 16 bits per sample	For a similar computational complexity, the single-layer system obtains an error probability equal 11.53%, while the dual-layer system reduces the error probability down to 8.83%.	The experiments prove that the two-layer approach presents a lower computational complexity in terms of the number of sums and multiplications required to obtain an output.
(Bachu et al., 2010) *Separation of Voiced and Unvoiced using Zero crossing rate and Energy of the Speech Signal.*	Separation of voiced and unvoiced speech	ZCR and Energy	Not stated	Not stated	Not stated	
(Balabko, 1999) *Speech and music discrimination based on signal modulation spectrum.*	Speech/music discrimination for automatic speech recognition systems	MFCC. Used 6th band (466-600 Hz) and 20th band (1510-1732 Hz) out of 40 bands	Gaussian Classifier	News broadcast of 24 minutes and studio music, various genres	Between 62% of music accuracy and 98% speech accuracy	

(Barbedo and Lopes, 2006) *Automatic genre classification of musical signals.*	Automatic genre classification of musical signals.	Bandwidth, spectral roll-off, spectral flux, and loudness. Three summary features are extracted: mean, variance, and main peak prevalence	Pairwise Euclidean distance four-layer hierarchical structure search tree and linear discriminant analysis (LDA)	More than 20 hours of audio data. Each genre is represented by at least 40 samples. The signals were sampled at 48 kHz and quantized with 16 bits.	The accuracy achieved for the first layer is higher than 87% and accuracy of 61% for the lower level genres	
(Beierholm and Baggenstoss, 2004) *Speech music discrimination using class-specific features.*	Class-specific features discriminating between speech and music.	Auto-correlation function (ACF)	Class-specific density functions were estimated using Gaussian mixture HMMs. 130 speech samples and 75 music samples were used in the density estimation	Speech: 560 seconds / 11 speakers Music: 12 minutes classical, 20 minutes Pop 2-second clips, 16 kHz sample rate	Speech: untrained 100% / trained 100% Music: untrained 80 / trained 100%	Used class-specific features approach. Given the small training corpus, the result does not seem plausible. No mention of training vs testing data ratio. Gross overfitting?
(Bugatti, Flammini and Migliorati, 2002) *Speech music discrimination using class-specific features.*	"Table of Content description" of a multimedia document	ZCR-based features, spectral flux, short-time energy, cepstrum coefficients, spectral centroids, the ratio of the high-frequency power spectrum, a measure based on the syllabic frequency	Multivariate Gaussian classifier, neural network (MLP)	30 minutes of alternating sections of music and speech (5min each)	95%–96% (NN). Total error rate: 17.7% (Bayesian classifier), 6.0% (NN).	As a first stage, a silence detector is used, which divides the silence frames from the others with a measure of the short-time energy

(Bugatti, Leonardi and Rossi, 1999) *A Video Indexing Approach Based on Audio Classification.*	Video indexing approach based only on audio classification.	ZCR variance, skewness (3rd moment), Difference between the number of ZCR samples which are above and below the mean and Short-time energy	Multivariate Gaussian classifier	Not stated	Not stated	
(Burred and Lerch, 2004) *A Video Indexing Approach Based on Audio Classification.*	Audio classification (speech/ music/backgroun d noise), music classification into genres	Statistical measures of the short-time frame features: ZCR, spectral centroid/roll-off/flux, first 5 MFCCs, audio spectrum centroid/flatness, harmonic ratio, beat strength, rhythmic regularity, RMS energy, time envelope, low energy rate, loudness, others	KNN classifier, 3-component GMM classifier	3 classes of speech, 13 genres of music and background noise: 50 examples for each class (30 sec each), from CDs, MP3, and radio.	94.6% /96.3% (hierarchical approach and direct approach, resp.)	
(Carey, Parris and Lloyd-Thomas, 1999) *A comparison of features for speech, music discrimination*	Proof of concept for pitch and amplitude features	Mel Frequency Cepstral Coefficients, Delta Cepstral Coefficients, Amplitude, Delta Amplitude, Pitch, Delta Pitch, Zero-Crossing Rate and Delta Zero-Crossing Rate	Gaussian Mixture Model using the difference in the log-likelihood	12 hours of multilingual speech and 8 hours of music (classical, pop, jazz and some world music) all in 10-second files	Results reported as equal error rate (EER): ZCR features 6% EER, Pitch features 4% EER and amplitude features 1.2-1.7% EER	

(Wu Chou and Liang Gu, 2001) *Robust singing detection in speech/music discriminator design.*	Signing signal detection in speech/music discrimination applied to applications of audio indexing.	4 Hz modulation energy, harmonic coefficient and 4 Hz modulation value of the harmonic coefficient	Gaussian Mixture Model	PBS skills database. 26 minutes of speech, music and singing signals. 16-bit monophonic samples at 16 kHz sampling rate	62% error rate reduction.	Two-stage classification algorithm - classified into singing and non-singing segments in the first stage, followed by conventional speech/music discrimination in the second stage. Rule-based Post-Filtering
(El-Maleh et al., 2000) *Speech/music discrimination for multimedia applications.*	Automatic coding and content-based audio/video retrieval	Line spectral frequencies (LSF), differential LSF, measures based on the ZCR of high-pass filtered signal and linear prediction zero-crossing ratio (LP-ZCR)	KNN classifier and quadratic Gaussian classifier (QCG)	Music and speech audio recordings with 8 kHz sampling frequency. Several speakers, different genres of music (training: 9.3 min. and 10.7min., resp.)	Frame level (20ms): music 72.7% (QGC), 79.2% (KNN). Speech 74.3% (QGC), 82.5% (KNN). Segment level (1 sec.), music 94%–100%, speech 80%–94%.	
(Ericsson, 2009) *Automatic speech/music discrimination in audio files.*	Thesis – Discrimination and annotation of speech and music for radio broadcasts.	Variance, standard deviation and the derivative and the standard deviation of the RMS amplitude, Zero Crossing-Rate (ZCR), Mel Frequency Cepstrum Coefficients (MFCC), Spectral Centroid (SC), Pulse Clarity (PC) and Modified Low	Compare feature histograms per class. Final classification algorithm not stated.	Swedish Radio's digital archive (Digas).	Accuracy of over 97% stated.	Only the MLER feature was used for classification

		Energy Ratio (MLER)				
(Foote, 1997) *Content-based retrieval of music and audio.*	Audio search engine. Retrieving audio documents by acoustic similarity	12 MFCC, Short-time energy	Template matching of histograms, created using a tree-based, vector quantizer, trained to maximize mutual information. Create histogram templates of classes to compare unknown samples based on Euclidian and Cosine distance measures.	409 sounds and 255 (7 sec long) clips of music. 16 kHz / 16Bit sampled.	No specific accuracy rates are provided. A high rate of success in retrieving simple sounds.	Rejected the use of HMM due to computational complexity. Cosine distance outperformed Euclidian distance measures comprehensively
(Gallardo-Antolin and Montero, 2010) *Content-based retrieval of music and audio.*	Automatically classifying collections of audio files in three acoustic classes: speech, instrumental music and song (music with singing voice).	Mean, variance, kurtosis and skewness of 12 MFCC. Polynomial-Fit Histogram Equalization (PHEQ) of the MFCC	GMM-based classifier	2440 audio files covering a wide variety of speakers and musical genres. 800 excerpts of speech, 901 of instrumental music, and 739 of songs (singing voice). 22.05 kHz sampling rate. 6-fold cross-validation	Frame level (25ms): MFCC only – 72.34% accuracy, Normalised MFCC 73.36% Segment level (1.5s): PHEQ 2nd order – 75.29%, Normalised to mean MFCC + PHEQ 2nd order – 81.52% and normalized MFCC augmented with their corresponding first derivatives +	MFCC mean and variance are not discriminative enough for distinguishing between the different audio classes.

					PHEQ 2nd order: 83.81%	
(Giannakopoulos, Pikrakis and Theodoridis, 2006) *A speech/music discriminator for radio recordings using Bayesian networks.*	Speech/music discriminator for radio recordings.	The first and second moment of Spectral Centroid, Spectral Flux, Spectral Rolloff, Zero Crossing Rate, Frame Energy and four Mel-frequency cepstral coefficients (MFCCs)	Bayesian Network (BN) that combines the outputs of nine individual k-Nearest Neighbour classifiers.	Distinct radio broadcasts (3 hours of total recording duration). 16 kHz sampling.	Average classification accuracy of 94.5%.	Use the region growth filtering technique to aggregate classifier output
(Giannakopoulos, 2015) *PyAudioAnalysis: An open-source python library for audio signal analysis.*	Python Audio Analysis library is to provide a wide range of audio analysis functionalities	Zero-Crossing Rate, Energy, Entropy of Normalized Energy, Spectral Centroid, Spectral Spread (The second central moment of the spectrum.), Spectral Entropy, Spectral Flux, Spectral Rolloff, 12 MFCCs, Chroma Vector (12-element) and Chroma Deviation	Support vector machines and the k-Nearest Neighbour classifier	N/A	N/A	

(Graves, Mohamed and Hinton, 2013) *Speech recognition with deep recurrent neural networks.*	Speech recognition.	Map directly from acoustic to phonetic sequences. No feature extraction required.	Recurrent neural networks (RNNs). All networks were trained using stochastic- tic gradient descent, with learning rate 10−4, momentum 0.9 and random initial weights were drawn uniformly from [−0.1, 0.1].	TIMIT Acoustic-Phonetic Continuous Speech Corpus. 462 Speakers of eight major dialects of American English, each reading ten phonetically rich sentences. The TIMIT corpus includes time-aligned orthographic, phonetic and word transcriptions as well as a 16-bit, 16kHz speech waveform file for each utterance.	Best result of 17.7% phoneme error rate on the core test set.	Instead of combining RNNs with HMMs, it is possible to train RNNs 'end-to-end' for speech recognition. A good discussion of Deep RNN's. Network depth is more important than layer size.
(Khan and Al-Khatib, 2006) *Machine-learning based classification of speech and music.*	Evaluation of features and classifiers for speech and music classification.	Haar Discrete wavelet transform, the variance of MFCC, RMS of lowpass filtered signal, delta ZCR. Used fuzzy C-means clustering to select viable feature.	Multi-Layer Perceptron (MLP) Neural Networks, radial basis functions (RBF) Neural Networks, and Hidden Markov Model (HMM)	Music, speech, and speech+music data. Speech: both genders, American English, Urdu, Japanese, Spanish, and Hebrew. The audio samples were extracted from documentaries and from different movies. There were approximately 2.25 h of speech, 2.72 h of music and 0.62 h of speech/music	MLP: Music 83.11%, Speech 39.86% Music+Speech 64.2% Overall 62.39% Accuracy RBF: Music 70.27%, Speech 20.27% Music+Speech 0% Overall 30.18% Accuracy HMM: Music 95.32%, Speech 95.32% Music+Speech 7.30% Overall 65.98% Accuracy	RBF networks give satisfactory results only for the English language. MLP networks and HMMs have given good results. A disadvantage of using HMMs is that it requires long training and testing time as compared to MLP.

				data distributed over 3-s audio files, 16-bit, 44.1 kHz, mono PCM wave files.		
(Lavner and Ruinskiy, 2009) *A Decision-Tree-Based Algorithm for Speech/Music Classification and Segmentation.*	Consumer audio application of segmentation of audio signals into speech or music	Mean and Standard deviation of Short-time energy, Zero Crossing rate, Band Energy Ratio, Autocorrelation Coefficient, MFCC, Spectral Rolloff, Spectral Centroid. Spectral Flux and Spectrum Spread. The skewness of ZCR and the difference magnitude between consecutive analysis frames. Low Short Time Energy	Three-stage sieve-like decision tree approach, applying both Bayesian and rule-based methods.	More than 12 hours of speech material was collected from free internet speech databases and more than 22 hours of music extracted from CD media and some databases covering multiple genres.	Correct identification rates of 99.4% (Speech) and 97.8% (Music)	Five thresholds are computed for each feature, based on the estimated PDFs. Elaborate rule-based final classifier and result smoothing scheme yield good results
(Lobo and Loizou, 2003) *Voiced/unvoiced speech discrimination in noise using Gabor*	Algorithm for voiced-unvoiced speech discrimination in noise.	Gabor coefficient atomic decomposition	Radial Basis function MLP neural network	62 sentences taken from the HINT database sampled at 20,161 Hz	84% correct classification accuracy	

atomic decomposition						
(Liu, Wang and Chen, 1998) *Audio Feature Extraction and Analysis for Scene Segmentation and Classification.*	Analysis of audio for the scene classification of TV programs	The mean and standard deviation of RMS value, Silence ratio, volume std, volume dynamic range, mean and std of pitch difference, speech, noise ratios, frequency centroid, bandwidth, energy in 4 sub-bands	A feedforward neural network using the one-class-in-one-network (OCON) structure	70 audio clips from TV programs (1 sec. long) for each scene class (training: 50, testing: 20) sampled at 22 kHz	Average 88% class separation accuracy	One-class-in-one-network (OCON) structure, where one subnet is designated for recognizing one class only
(Lu, Jiang and Zhang, 2001) *A robust audio classification and segmentation method.*	Audio content analysis in video parsing	High zero-crossing rate ratio (HZCRR), low short-time energy ratio (LSTER), linear spectral pairs, band periodicity, noise-frame ratio (NFR)	3-step classification: 1. KNN and linear spectral pairs-vector quantization (LSP-VQ) for speech/non-speech discrimination. 2. Heuristic rules for non-speech classification into music/background noise/silence. 3. Speaker segmentation	MPEG-7 test data set, TV news, movie/audio clips. Speech: studio recordings, 4 kHz and 8 kHz bandwidths, music: songs, pop (training: 2 hours, testing: 4 hours).	Speech 97.5%, music 93.0%, env. sound 84.4%. Results of only speech/music discrimination: 98.0%	

(Lu, Zhang and Li, 2003) *Content-based audio classification and segmentation by using support vector machines.*	Content-based audio classification and segmentation using support vector machines (SVM)	8 MFCC coefficients, zero-crossing rate (ZCR), short-time energy (STE), sub-band power distribution, brightness, bandwidth, spectrum flux (SF), band periodicity (BP), and noise frame ratio (NFR).	Compared the performance of SVM, K-Nearest Neighbor (KNN), and Gaussian Mixture Model (GMM). Audio is classified into five classes. They are silence, music, background sound, pure speech, and non-pure speech which include speech with music and speech with noise. Used rule-based pre- and post-classification	2600 audio clips, +- 4 hours in total length, collected from TVprograms, the Internet, audio and music CDs. 8 kHz sample rate at 16-bit per sample.	Average accuracy: Speech/non-speech 96.36% Music/background sound 94.67% Pure speech/non-pure speech 89.64%	SVM is more accurate and efficient than the KNN and GMM algorithms.
(Markaki and Stylianou, 2011) *Discrimination of speech from non-speech in broadcast news based on modulation frequency features.*	Discrimination of speech and non-speech for speaker segmentation/recognition and speech transcription	MFCC and Modulation spectrum envelope for 65 frequency sub-bands was detected by a magnitude square operator. Higher-order generalization of singular value decomposition (HOSVD) for dimensionality reduction.	SVM classifier with median filter smoothing	6 hours of broadcasts of Greek TV programs (ERT3). mono channel and 16 bit per sample, with 16 kHz sampling	Equal Error Rate: 3.78%	

(Muñoz-Expósito et al., 2009) *Speech/music discrimination based on warping transformation and fuzzy logic for intelligent audio coding.*	Intelligent audio coding system.	Warped LPC-based spectral centroid. Used the mean, variance, and skewness of the feature vector.	3-component GMM, with or without fuzzy rules-based system. SVM with a fuzzy rules-based system.	Speech (radio and TV news, movie dialogs, different conditions); music (various genres, different instruments/singers) -1 hour for each class.	GMM with LDA: speech 93.47%, music 86.95% overall 90.21%. RBF-SVM: speech 95.56%, music 88.95% overall 92.25%. RBF-SVM with a fuzzy system: speech 97.46%, music 98.85% overall 98.15%.	
(Panagiotakis and Tziritas, 2005) *A speech/music discriminator based on RMS and zero-crossings.*	Web-based streaming audio content characterisation.	Mean and variance of RMS value and ZCR as an indication of mean frequency.	Rule-based Gaussian likelihood ratio test.	3h speech and 52 min. music.	Overall classification accuracy about 95%. 97% for speech and 92% for music.	
(Pikrakis, Giannakopoulos and Theodoridis, 2006) *Speech/music discrimination for radio broadcasts using a hybrid HMM-Bayesian Network architecture.*	Discrimination scheme for radio recordings	Energy, zero-crossing rate, spectral entropy and the first two Mel- Frequency Cepstrum Coefficients (MFCCs).	Variable Duration Hidden Markov Model (VDHMM) and a Bayesian Network (BN).	340 minutes of BBC on-line radio recordings.	Overall performance: 94.95%	A joint segmentation/classification scheme is employed for speech/music discrimination using a hybrid architecture consisting of a Variable Duration Hidden Markov Model (VDHMM) and a Bayesian Network (BN).

(Pinquier, Rouas and E-Obrecht, 2002) *Robust speech/music classification in audio documents.*	To describe and index an audio document. Keywords and speakers detection.	4 Hz modulation energy, entropy modulation, number of "stationary" segments and segment duration.	Bayesian maximum likelihood classifier	MULTEX Corpus based on reading speech excerpts (20 kHz sampling rate) and a corpus-based on various musical excerpts. (16 kHz sampling rate). The total duration for each corpus (music and speech) was about 2000 seconds. An experimental corpus is an audio part sampled at 16 kHz of a 20 min TV movie	Overall performance of this algorithm is 90.1 % correct classification. Speech with an accuracy of 99.5% and music 93%	
(Saad et al., 2002) *A multi-feature speech/music discrimination system.*	Automatic classification of audio signals	Percentage of low energy frames, spectral roll-off, spectral centroid, spectral flux, ZCR	Rule-based classification	20 speech files and 20 music files. Duration not stated.	Speech: 90% Music 98.5% Overall: 94.25%	
(Saunders, 1996) *Real-time discrimination of broadcast speech/music.*	Automatic monitoring of FM radio channels in real-time	Short-time energy, statistical parameters of the ZCR	Multivariate Gaussian classifier	Talk, commercials, music (different types) 2h	95%–96%	
(Scheirer and Slaney, 1997) *Construction and Evaluation of a Robust Multi-feature Speech Music Discriminator.*	Speech/music discrimination for automatic speech recognition	13 temporal, spectral and cepstral features (e.g., 4 Hz modulation energy, % of low energy frames, spectral roll-off, spectral centroid,	Gaussian mixture model (GMM), K nearest neighbour (KNN), K-D trees, multidimensional Gaussian MAP estimator	FM radio (40min): male and female speech, various conditions, different genres of music (training: 36min, testing: 4 min)	94.2% (frame-by-frame), 98.6% (2.4 sec segments)	Most efficient: Low Energy & Zero-cross Rate Lowest error %: Var Spec Flux, 4 Hz Mod Energy & Low Energy. Best: 4 Hz energy, the variance of

		spectral flux, ZCR, cepstrum-based feature, "rhythmicness"), a variance of features across 1 sec. Pulse Metric				spectral flux, and pulse metric.
(Sigtia et al., 2016) *Automatic Environmental Sound Recognition: Performance Versus Computational Cost.*	Automatic Environmental Sound Recognition (AESR) algorithm for low power IoT devices.	MFCCs, Spectral centroid, Spectral flatness, Spectral roll-off, Spectral kurtosis and Zero crossing rate	Compared Gaussian mixture models (GMMs), support vector machines (SVMs) and deep neural networks (DNNs) in terms of their performance and their computational cost.	Three private data sets made available by Audio Analytic Ltd. Dataset 1: 4 822 seconds of training data and 3 669 seconds of test data, Dataset 2: 15 271 seconds of training data and 5 043 seconds of testing data and Dataset 3 consists of 5 000 seconds of training data and 4 089 seconds of test data. Recordings were converted to 16 kHz, 16 bits	Dataset 1 EER: GMM 14.0%, SVM 12.9% and DNN 10.8% Dataset 2 EER: GMM 2.9%, SVM 3.0% and DNN 1.7% Dataset 3 EER: GMM 14.0%, SVM 12.9% and DNN 10.8%	Neural Networks outperforms all the other models. Network performance is not particularly sensitive to the specific number of hidden units in each layer. However, they did observe that deeper networks (> 1 hidden layer) yielded better performance. Recurrent Neural Network architectures yield slightly worse performance and are computationally more expensive.
(Taniguchi, Tohyama and Shirai, 2008) *Detection of speech and music based on spectral tracking.*	Speech, music and mixed sound classification method based on sinusoidal trajectories.	Sinusoidal trajectories and temporal features extracted thereof.	Gaussian mixture models	Three categories: speech, singing voice, and instrument. 400-sample dataset, single channel at a 16-kHz sampling rate.	F1 score was 0.711 when using the 16-mixture GMMs and the 20 features	

				Duration not stated.		
(Tardón, Sammartino and Barbancho, 2010) *Design of an efficient music-speech discriminator.*	Efficient music-speech discriminator for large audio data sets	The mean and standard deviation of RMS, ZCR, Cepstrum Residuals, Spectral Flux, DFT Magnitude, 5 MFCC's, and Volume dynamic ratio were used in the final classification. Silence ratio, Spectral Centroid, Spectral Rolloff, Bandwidth, frame and segment energy, fundamental frequency and Salience of pitch was also evaluated.	Simple Gaussian model	Corpus supplied by Fundación Albéniz (Albeniz Foundation), "containing a large variety of classical music pieces played with a different instrument, which include comments and speeches of famous performers." No further information specified	Evaluated multiple combinations of features to compile a subset that attains a mean error rate of 0.3% over the test data set.	Fisher linear discriminator as a technique to reduce the dimensionality
(Tzanetakis and Cook, 2002) *Musical genre classification of audio signals.*	Genre categorisation for audio	Mean and Std Deviation of Spectral Centroid, Spectral-Rolloff, Spectral Flux, Zero Crossing ratio as well as Percentage Low Energy frames, MFCC and rhythm features derived from Discrete Wavelet Transform	Gaussian classifier	The database collected from radio, compact disks and the Web. Total of 6.25 hours of audio, fifteen genres including male and female speech	86% discrimination accuracy between music and speech	

(Wang, Gao and Ying, 2003) *A fast and robust speech/music discrimination approach.*	An experimental approach to discriminate speech and music	Modified Low Energy ratio	Bayes MAP classifier. Rule-based post-decision filter	5 hours of audio data, which involves clean and noisy speech from various speakers (English, French and Chinese), as well as a wide range of musical content.	Classification accuracy of 97% for music and 98.4% for speech	
(Williams and Ellis, 1999) *Speech/music discrimination based on the posterior probability*	Segmentation of speech versus non-speech in automatic speech recognition tasks	Mean per-frame entropy and average probability "dynamism", background-label energy ratio, phone distribution match—all derived from posterior probabilities of phones in hybrid connectionist-HMM framework	Gaussian likelihood ratio test	Radio recordings, speech (80 segments, 15 sec. each) and music (80, 15), respectively. Training: 75%, testing: 25%.	100% accuracy with 15 seconds long segments 98.7% accuracy with 2.5- seconds long segments	
(Zhang and Jay Kuo, 2001) *Audio content analysis for online audio-visual data segmentation and classification*	Audio segmentation/ retrieval for video scene classification, indexing of raw audiovisual recordings, database browsing	Features based on short-time energy, average ZCR, short-time fundamental frequency	A rule-based heuristic procedure for the coarse stage, HMM for the second stage	Coarse stage: speech, music, env. sounds and silence. Second stage: fine-class classification of env. sounds.	>90% (coarse stage)	

Appendix B – Summary of classifiers used in previous research

Support Vector Machine (SVM)	**Bayesian Classifier**	**Gaussian Classifier**	**Gaussian Mixture Model (**
Giannakopoulos, T. (2015)	Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006).	Balabko, P. (1999	Ajmera, J., McCowan, I. & B (2003).
Lu, L., Zhang, H.-J., & Li, S. Z. (2003)	Lavner, Y., & Ruinskiy, D. (2009)	Bugatti, A., Flammini, A., & Migliorati, P. (2002).	Burred, J. J., & Lerch, A. (2004
Markaki, M., & Stylianou, Y. (2011)	Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2006)	Bugatti, A., Leonardi, R., & Rossi, L. (1999)	Carey, M. J., Parris, E. S., & Llo H. (1999)
Munoz-Exposito, J. E., Galan, S. G., Reyes, N. R., & Candeas, P. V. (2009)	Pinquier, J., Rouas, J. L., & E-Obrecht, R. a. (2002)	Panagiotakis, C., & Tziritas, G. (2005)	Wu Chou, & Liang Gu. (2001)
Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)	Saunders, J. (1996)	Scheirer, E. & Slaney, M., (1997)	Gallardo-Antolin, A., & Mor (2010)
	Wang, W. Q., Gao, W., & Ying, D. W. (2003)	Tardón, L. J., Sammartino, S., & Barbancho, I. (2010)	Lu, L., Zhang, H.-J., & Li, S. Z. (
	Williams, G., & Ellis, D. P. W. (1999)	Tzanetakis, G., & Cook, P. (2002)	Munoz-Exposito, J. E., Galan, N. R., & Candeas, P. V. (2009)
			Scheirer, E., Slaney, M., & Alt
			Sigtia, S., Stark, A. M., Krstu Plumbley, M. D. (2016)
			Taniguchi, T., Tohyama, M., (2008)
Neural Networks (MLP, RNN, CNN)	**Discriminant Analysis**	**k-Nearest Neighbor (kNN)**	**Feature Histograms**
Multi-layer perceptron			
Ajmera, J., McCowan, I. & Bourlard, H., (2003).	Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Burred, J. J., & Lerch, A. (2004)	Foote, J. T. (1997)
Alexandre, E., Cuadra, L., Álvarez, L., Rosa-Zurera, M., López-Ferreras, F. (2006)	Alexandre, E., Gil-Pita, R., Cuadra, L., Álvarez, L., & Rosa-Zurera, M. (2008)	Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2006).	Ericsson, L. (2009)
Bugatti, A., Flammini, A., & Migliorati, P. (2002).		Giannakopoulos, T. (2015)	
Khan, M. K. S., & Al-Khatib, W. G. (2006)		Lu, L., Jiang, H., Zhang, H.-J. (2001)	

A. P. Lobo and P. C. Loizou (2003)		Lu, L., Zhang, H.-J., & Li, S. Z. (2003)	
Liu, Z., Wang, Y., & Chen, T. (1998)		Scheirer, E. & Slaney, M., (1997)	
Sigtia, S., Stark, A. M., Krstulović, S., & Plumbley, M. D. (2016)			
Recurrent Neural Network			
Graves, A., Mohamed, A., & Hinton, G. (2013)			

Appendix C – Approved Ethical Clearance

UNISA university of south africa

UNISA SOE ETHICS REVIEW COMMITTEE

DATE: 11/01/2021

Dear Mr Herman Redelinghuys

ERC Reference #: 2021/CSET/SOE/001
Name: Mr Herman Redelinghuys
Student #: 40531988
Staff #: N/A

Decision:
Research Ethics Notification Letter

Researcher(s): Name: Mr Herman Redelinghuys
Email: 40531988@mylife.unisa.ac.za
Telephone: 021 889 8608/083 415 2524

Supervisor(s): Name: Prof Zenghui Wang
Email: wangz@unisa.ac.za
Telephone: 011 471 3513

Working title of research:

Development of neural network based speech/non-speech discrimination algorithm for audio files

Qualification: MTech

This letter serves as notification that the research undertaken in the dissertation entitled: "Development of neural network based speech/non-speech discrimination algorithm for audio files" was conducted in the absence of an approved research ethics certificate. The research study was conducted from 2017 to 2020. The research study did not involve fieldwork where humans were involved, and no secondary data, animals or plants were involved in the study. The research study is regarded as a negligible risk (risk 1) study.

According to policy, the university cannot grant research ethical clearance retrospectively. The researcher submitted his dissertation together with a completed ethics application form

University of South Africa
Preller Street, Muckleneuk Ridge, City of Tshwane
PO Box 392 UNISA 0003 South Africa
Telephone: +27 12 429 3111 Facsimile: +27 12 429 4150
www.unisa.ac.za